Leveraging lyrics from audio for MIR

Andrea Vaglio

Leveraging lyrics from audio for MIR

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat : Signal, Images, Automatique et robotique

Thèse présentée et soutenue à Palaiseau, le 19 Novembre 2021, par

ANDREA VAGLIO

Composition du Jury :

Emmanuel Vincent Directeur de recherche, INRIA	Président
Isabel Barbancho Professeure, Universidad de Málaga	Rapporteur
Simon Dixon Professeur, Queen Mary University of London	Rapporteur
Anna Kruspe Docteure, Technical University of Munich	Examinateur
Romain Hennequin Docteur, Deezer	Examinateur
Gaël Richard Professeur, Télécom Paris	Directeur de thèse
Manuel Moussallam Docteur, Deezer	Invité
Florence D'Alché-Buc Professeure, Télécom Paris	Invité

Preface

This work has been done between June 2018 and May 2021. It is a collaboration between Deezer and Telecom Paris. The thesis has been carried out in both Deezer Research team, under the supervision of Romain Hennequin and the *Audio Data Analysis & Signal Processing* (ADASP) team of the *Laboratoire de Traitement et Communication de l'Information* (LTCI) laboratory, under the supervision of Gaël Richard. This work has been supported jointly by Deezer and *Association Nationale de la Recherche et de la Technologie* (ANRT) through a *Convention Industrielle de Formation par la REcherche* (CIFRE) contract.

Acknowledgements

This work was made possible thanks to the help of many mentors, colleagues and friends. First of all, I would like to particularly thank all my supervisors for their kindness, their patience and their always great advice. I really felt that none of this would have been possible without the constant help that you gave me. So, my heartfelt thanks to, in no particular order: Romain, Manu, Gaël and Florence. I would also express my gratitude to all my colleagues, being from Deezer or Telecom Paris, making the workplace a friendly, comfortable and warm place to enjoy growth and experience my project. Sadly, most of the time has been stolen by the pandemic situation, I would have liked to deepen my bonds with you more.

I also thank Isabel Barbancho and Simon Dixon for the time they dedicated to review my manuscript. I would especially like to thank all my relatives, friends or family, for supporting me despite all the particularly frequent mood changes that I have shown during these three years. I am especially grateful to my parents that support me emotionally during my best but also worst moments. I do not think that I would have to go through the last part of my thesis without you. To all those that I forget, I am thankful that I met you during this journey.

Abstract

Lyrics provide a lot of information about music since they encapsulate a lot of the semantics of songs. Such information could help users navigate easily through a large collection of songs and to recommend new music to them. However, this information is often unavailable in its textual form. To get around this problem, singing voice recognition systems could be used to obtain transcripts directly from the audio.

These approaches are generally adapted from the speech recognition ones. Speech transcription is a decades-old domain that has lately seen significant advancements due to developments in machine learning techniques. When applied to the singing voice, however, these algorithms provide poor results. For a number of reasons, the process of lyrics transcription remains difficult.

To begin, music (*i.e.* accompaniment) might be thought of as significant background noise for the signal of interest (*i.e.* singing voice). It is usually mixed at a level comparable to the vocals and is significantly correlated with it. Secondly, a song is an object of art with variable form and intelligibility of lyrics with potential large phoneme pronunciation differences from one song to another. Nonetheless, lyrics information collected from the singing voice recognition systems might be utilized to execute a variety of lyrics-related tasks.

In this thesis, we investigate several scientifically and industrially difficult 'Music Information Retrieval' problems by utilizing lyrics information generated straight from audio. The emphasis is on making approaches as relevant in real-world settings as possible. This entails testing them on vast and diverse datasets and investigating their scalability. To do so, a huge publicly available annotated lyrics dataset is used, and several state-of-the-art lyrics recognition algorithms are successfully adapted.

We notably present, for the first time, a system that detects explicit content directly from audio, yielding promising results on an industrial size explicit content dataset. The first research on the creation of a multilingual lyrics-to-audio system are as well described. The use of phonemes as intermediate representation and the design of a multilingual training data are both shown to be salient factors to improve multilingual generalization of the considered architecture for the task.

The lyrics-to-audio alignment task is further studied in two experiments quantifying the perception of audio and lyrics synchronization. The obtained results allow notably for a discussion of current lyrics-to-audio metrics. A novel phonotactic method for language identification is also presented, outperforming the state-of-the-art performance for the task. Finally, we provide the first cover song detection algorithm that makes explicit use of lyrics information extracted from audio. Extensive studies on the biggest publicly accessible cover detection dataset indicate the utility of employing lyrics information for this task.

***Keywords* -** Singing voice recognition, Explicit content detection, Lyrics-to-audio alignment, Language identification, Cover song detection

Résumé

Les paroles de chansons fournissent un grand nombre d'informations sur la musique car elles contiennent une grande partie de la sémantique des chansons. Ces informations pourraient aider les utilisateurs à naviguer facilement dans une large collection de chansons et permettre de leur offrir des recommandations personnalisées. Cependant, ces informations ne sont souvent pas disponibles sous leur forme textuelle. Les systèmes de reconnaissance de la voix chantée pourraient être utilisés pour obtenir des transcriptions directement à partir de la source audio.

Ces approches sont usuellement adaptées de celles de la reconnaissance vocale. La transcription de la parole est un domaine vieux de plusieurs décennies qui a récemment connu des avancées significatives en raison des derniers développements des techniques d'apprentissage automatique. Cependant, appliqués au chant, ces algorithmes donnent des résultats peu satisfaisants et le processus de transcription des paroles reste difficile avec des complications particulières.

Tout d'abord, la musique (*i.e.* accompagnement) peut être considérée comme un bruit de fond important pour le signal d'intérêt (*i.e.* voix chantée). Elle est généralement mixée à un niveau comparable à celui de la voix et présente une corrélation significative avec celle-ci. De plus, une chanson est un création artistique dont la forme et l'intelligibilité des paroles sont variables, avec de grandes différences potentielles de prononciation des phonèmes d'une chanson à l'autre. Cependant, les informations sur les paroles collectées par les systèmes de reconnaissance de la voix chantée peuvent être employés pour effectuer une variété de tâches liées aux paroles.

Dans cette thèse, nous étudions plusieurs problèmes de 'Music Information Retrieval' scientifiquement et industriellement complexes en utilisant des informations sur les paroles générées directement à partir de l'audio. L'accent est mis sur la nécessité de rendre les approches aussi pertinentes que possible dans le monde réel. Cela implique par exemple de les tester sur des ensembles de données vastes et diversifiés et d'étudier leur extensibilité. À cette fin, nous utilisons un large ensemble de données publiques possédant des annotations vocales et adaptons avec succès plusieurs des algorithmes de reconnaissance de paroles les plus performants.

Nous présentons notamment, pour la première fois, un système qui détecte le contenu explicite directement à partir de l'audio. Cette approche obtient des résultats prometteurs sur un ensemble de données de taille industrielle de contenu explicite. Les premières recherches sur la création d'un système d'alignement paroles-audio multilingue sont également décrites. L'utilisation de phonèmes comme représentation intermédiaire et la conception d'une base de données d'entraînement multilingue sont démontrés être des facteurs importants pour améliorer la généralisation multilingue de l'architecture considérée pour la tâche.

L'étude de la tâche alignement paroles-audio est complétée de deux expériences quantifiant la perception de la synchronisation de l'audio et des paroles. Les résultats obtenus permettent notamment d'ouvrir une réflexion sur les métriques actuelles de l'alignement paroles-audio. Une nouvelle approche phonotactique pour l'identification de la langue est également présentée, améliorant les performances de l'état de l'art pour cette tâche. Enfin, nous proposons le premier algorithme de détection de versions employant explicitement les informations sur les paroles extraites de l'audio. Des études approfondies sur le plus grand ensemble de données de détection de version accessible publiquement démontrent l'utilité de l'utilisation des informations sur les paroles pour cette tâche.

***Mots clés* -** Reconnaissance de la voix chantée, Détection de contenu explicite, Alignement des paroles et de l'audio, Identification du langage, Détection de reprise

French summary

Deezer est un service de streaming musical en ligne disposant d'un large catalogue, en constante évolution, de données multimodales hétérogènes. Il se compose de plus de 73 millions de titres et s'enrichit de dizaines de milliers de nouveaux titres chaque jour. Une collection aussi vaste doit être organisée et qualifiée automatiquement pour aider les utilisateurs à la parcourir et pour effectuer des recommandations personnalisées et intelligentes. Il est alors nécessaire d'utiliser des descripteurs musicaux de haut niveau comme le genre ou la langue des morceaux de musique. De tels descripteurs peuvent être disponibles dans les métadonnées, mais ce type de métadonnées est souvent bruité et disponible seulement pour une petite partie du catalogue. Une alternative consisterait à extraire ces descripteurs à partir de l'audio en utilisant des approches de 'Music Information Retrieval'. Ce domaine s'intéresse notamment à la création de systèmes capables d'extraire des descripteurs musicaux à partir de l'analyse de son signal audio.

Récemment, un intérêt croissant pour l'utilisation des paroles de chansons pour l'extraction des caractéristiques musicales de haut niveau est apparu. Les paroles constituent une source essentielle d'informations musicales car elles encodent une part importante de la sémantique des chansons. Elles peuvent être utilisées pour une multitude de tâches (détection de contenu explicite, détection de langue...) et d'applications spécifiques comme le karaoké. Les paroles sont disponibles pour environ $1,5$ millions de titres dans le catalogue Deezer. Pour environ 20 % de ces titres, les marqueurs temporels de début et de fin de chaque phrase des paroles sont disponibles. Ces informations sont généralement fournies en externe par des annotateurs manuels, rendant l'acquisition lente, coûteuse, de qualité variable et ne couvrant qu'une fraction du catalogue.

Une façon d'extraire les paroles à partir de l'audio est l'utilisation d'algorithmes de reconnaissance de la voix chantée. La plupart du temps, ces algorithmes sont inspirés de la reconnaissance vocale. La transcription de la parole est un domaine relativement mature et des résultats impressionnants ont été obtenus récemment grâce aux derniers développements des réseaux de neurones profonds. Cependant, ces algorithmes donnent des résultats insatisfaisants lorsqu'ils sont appliqués à la voix chantée.

L'objectif spécifique de la transcription de paroles reste un défi complexe avec des problématique particulières qui n'apparaissent pas dans la transcription de la parole. Tout d'abord, l'accompagnement musicale peut être considéré comme un bruit de fond à niveau sonore élevé fortement corrélé avec le signal d'intérêt puisque le chanteur chante généralement en harmonie et en rythme avec l'accompagnement. Ensuite, une chanson est un création artistique dont la forme et l'intelligibilité des paroles sont variables, avec de grandes différences potentielles de prononciation des phonèmes d'une chanson à l'autre.

Pourtant, les informations sur les paroles extraites à l'aide d'un système de reconnaissance peuvent être utiles pour accomplir diverses tâches liées aux paroles.

Dans cette thèse, nous étudierons plusieurs tâches de 'Music Information Retrieval' scientifiquement et industriellement complexes liées à la transcription de paroles. Pour ce faire, les systèmes de reconnaissance des paroles les plus modernes sont adaptés avec succès aux tâches considérées. De plus, nous utilisons une large base de données publique annotée de voix chantée publiée récemment. Quatre sujets sont étudiés dans le cadre de cette thèse, chacun ayant des applications industrielles potentielles importantes pour Deezer :

Détection de contenu explicite : La détection de contenu explicite consiste à classer un enregistrement audio comme étant *explicite* ou non *explicite*. Il s'agit d'une tâche particulièrement délicate pour les services de streaming notamment dans le cadre du contrôle parental. Cette thèse présente le premier système de détection de contenu explicite directement à partir d'un signal audio. L'approche modulaire proposée utilise un modèle acoustique dont les sorties représentent les caractères de l'alphabet latin, un modèle de détection de mots-clés associé à un dictionnaire de mots-clés soigneusement choisis et un modèle de forêt d'arbres décisionnels pour la décision finale.

Alignement paroles-audio : L'alignement paroles-audio vise à synchroniser le texte des paroles, au niveau du paragraphe, de la ligne ou du mot, avec la position temporelle de l'apparition de chaque unité dans le signal audio. Les outils dédiés à cette tâche ont de nombreuses applications pratiques : ils peuvent être appliqués pour générer de nouvelles données annotées afin d'entraîner des reconnaisseurs de voix chanté plus robustes ou être utilisés pour des applications spécifiques telles que le karaoké ou la suppression des paroles explicites.

Dans cette thèse, nous initions la réalisation d'un création d'un système d'alignement paroles-audio indépendant de la langue. Pour ce faire, nous nous concentrons d'abord sur une architecture de l'état de l'art qui semble adaptée à la généralisation. Ensuite, nous étudions la pertinence de différentes représentations intermédiaires, que ce soit des caractères ou des phonèmes, ainsi que plusieurs stratégies pour concevoir un ensemble d'entraînement.

L'évaluation est menée sur de multiples jeux de données, provenant de diverses sources, langues et écritures, avec une quantité de données disponibles pour l'entraînement variant de conséquente à nulle. Nous considérons notamment le cas des chansons multilingues. Des expériences supplémentaires visant à prendre en compte la similarité des phonèmes lors de l'alignement sont également présentées. Enfin, nous étudions la perception de la synchronisation de l'audio et des paroles à travers deux expériences réalistes inspirées du karaoké, et examinons les implications pour les métriques d'évaluation de l'alignement paroles-audio.

Identification de la langue : L'identification de la langue est la tâche consistant à détecter la langue chantée pour une chanson donnée. Les applications sont nombreuses : informer la transcription des paroles, améliorer la classification de genre ou aider à quantifier la distribution des langues des chansons diffusées dans les médias (nécessaire par exemple pour la loi Toubon). Dans cette thèse, nous proposons un nouveau système phonotactique utilisant des modèles récurrents pour la reconnaissance des phonèmes et la classification de la langue.

La reconnaissance des phonèmes est réalisée à l'aide d'un modèle acoustique basé sur l'algorithme 'Connectionist Temporal Classification' entraîné avec des données multilingues, avant classification de la langue à l'aide d'un modèle récurrent basé sur l'estimation des phonèmes. Les premiers résultats de l'identification de la langue du chant avec des langues non définies dans l'ensemble d'entraînement sont également présentés.

Détection de version : La détection de version vise à déterminer si deux enregistrements proviennent de la même œuvre musicale. Une application pratique est la détection de plagiat. Dans ce travail, nous proposons le premier système de détection de version qui exploite explicitement les informations sur les paroles des chansons à partir de l'audio. Notre approche repose sur la fusion d'un outil de reconnaissance de la voix chantée et d'une approche plus classique de détection de version basée sur des descripteurs harmoniques. De plus, nous exploitons une méthode efficace de recherche floue pour quantifier la similarité de chaînes de caractères et un algorithme approché de recherche des plus proches voisins permettant la création d'un système extensible capable de maintenir des performances correctes sur de très grandes bases de données.

Table of Contents

Notations

General notations

x A scalar $x \in \mathbb{R}$.

$\mathbf{X}$ A vector $\mathrm{X} \in \mathbb{R}^{A}$.

$\mathbf{X}$ A matrix $\mathbf{X} \in \mathbb{R}^{A*B}$.

$\overline{\mathbf{X}}$ A 3D-tensor $\overline{\mathbf{X}} \in \mathbb{R}^{A*B*C}$.

Singing voice recognition pipeline

$\mathscr{H}$ An acoustic model $\mathscr{H}$. Generally, $\mathscr{H}$ takes as input an acoustic feature matrix $\mathbf{X} \in \mathbb{R}^{F*T}$ and outputs a posteriorgram $\mathbf{R} \in [0,1]^{|\mathcal{C}|*T}$. $F, T, \mathcal{C}$ are respectively the number of features, the number of frames and the set of units outputted by the model (*e.g.* phonemes).

$\mathscr{L}$ A language model $\mathscr{L}$. It is based on a vocabulary, the set of words that can be recognized by the model, and defines a probability distribution across all sequences of words possible. A probability $P(\boldsymbol{s})$ can then be obtained for any sequence of words $\boldsymbol{s}$. It can also be defined at subword levels such as phonemes or characters.

Others

$P(Y)$ A probability distribution of a random variable Y.

$\boldsymbol{s}$ A string $\boldsymbol{s}$ with s_i representing the i^{th} character of the string.

Acronyms

AAE *Average Absolute Error*. 71, 73, 82–85, 88, 152, 155, 156, 158, 159

ADASP *Audio Data Analysis & Signal Processing*. 2

ANN *Approximated Nearest Neighbor*. 115, 117, 119, 122, 125, 129

ANRT *Association Nationale de la Recherche et de la Technologie*. 2

AP *Average Precision*. 117

API *Application Programming Interface*. 90, 130

ASR *Automatic Speech Recognition*. 23, 29–31, 33, 41, 56, 71, 109, 132, 134

AUC *Area Under Curve*. 67

BERT *Bidirectional Encoder Representations from Transformers*. 133

BiLSTM *Bidirectional Long Short-Term Memory*. 42, 46, 57, 58, 76, 109, 111, 151

BPM *Beats Per Minute*. 97, 99

CER *Character Error rate*. 54, 59, 60, 151

CIFRE *Convention Industrielle de Formation par la REcherche*. 2

CRNN *Convolutional Recurrent Neural Network*. 56, 66

CTC *Connectionist Temporal Classification*. 26, 30, 32, 45–53, 55–57, 59–61, 65, 73, 76, 78, 81, 86, 106, 109, 111–113, 127, 133, 150, 151

DNN *Deep Neural Network*. 30, 41, 105, 106, 111, 112

DTW *Dynamic Time Warping*. 75, 116

E2E *End-To-End*. 30–32, 34, 35, 38, 43, 45, 56, 59, 64, 66, 73, 109, 111, 112, 132

FiLM *Feature-wise Linear Modulation*. 113

GMM *Gaussian Mixture Model*. 30, 107, 111

GTP *Grapheme-To-Phoneme*. 77, 80, 132

HMM *Hidden Markov Models*. 30, 31, 33, 47, 63, 75, 115

HNSW *Hierarchical Navigable Small World Graph*. 122, 123, 125

HPCP *Harmonic Pitch Class Profile*. 116, 121

IDF *Inverse Document Frequency*. 120

IOI *InterOnset Interval.* 93

IPA *International Phonetic Alphabet.* 39, 43–45, 74, 76, 77, 80, 81, 85–87, 106, 109, 110, 127, 130, 150

ISMIR *International Society for Music Information Retrieval.* 98

KNN *K-Nearest Neighbors.* 122

LID *Language IDentification.* 105, 107, 109, 110

LRE *Language Recognition Evaluation.* 105, 110

LSH *Locality Sensitive Hashing.* 116, 122

LSTM *Long Short-Term Memory.* 42, 56, 57

LTCI *Laboratoire de Traitement et Communication de l'Information.* 2

MAP *Mean Average Precision.* 117, 124, 125, 131

MFCC *Mel-Frequency Cepstral Coefficients.* 41, 105, 110, 116, 120

MIR *Music Information Retrieval.* 20–22, 25, 92, 129, 150

MIREX *Music Information Retrieval Evaluation eXchange.* 32, 34, 38, 59, 70–74, 81, 91, 92, 99, 120, 152, 154

MLLR *Maximum Likelihood Linear Regression.* 31

MLP *MultiLayer Perceptron.* 75

NMA *Negative Mean Asynchrony.* 92

OOV *Out Of Vocabulary.* 134

PCO *Percentage of Correct Onsets.* 73, 74, 81–85, 88, 89, 91, 99, 130, 152, 155, 156, 158, 159

PR *Precision Recall.* 67

RNN *Recurrent Neural Network.* 32, 42, 45, 47, 56, 76, 106, 109, 127

ROC *Receiver Operating Characteristic.* 67

SAI *Stabilized Auditory Images.* 105

SLID *Singing Language IDentification.* 105–113, 153

STFT *Short Time Fourier Transform.* 40, 41

SVM *Support Vector Machine.* 111

SVR *Singing Voice Recognition.* 21, 23, 25, 27, 29, 31, 34–40, 43, 46, 49, 50, 52, 55, 56, 61, 63, 105, 115, 117, 120, 122, 124, 127–129, 131, 132, 134, 150, 155

SVS *Singing Voice Separation.* 23, 24, 29, 32–35, 38–41, 56, 59, 64, 65, 76, 77, 108, 131, 132, 134, 150–154

TDNN *Time Delay Neural Network.* 32

TFIDF *Term Frequency–Inverse Document Frequency.* 114, 118–120, 122

TRAP *Temporal Patterns*. 105

UBM *Universal Background Model*. 107, 111

WER *Word Error rate*. 23, 29–32, 34, 35, 37, 54, 59, 60, 120, 127, 128, 151, 154

WPS *Words Per Second*. 99–101

Part I

Extracting lyrics information from audio

Chapter 1

Introduction

1.1 General context

In 2020, the recording industry is a major economic sector weighing more than 21.6 billions of dollars. Since the advent of the internet, the way music is consumed, discovered and distributed has drastically changed. After their peak in 2000, CD album sales in the United States have fallen by 97 % reducing the amount of millions of units sold from 938 to just 31.6 [oA20].

Streaming services such as Deezer, Spotify, and Apple Music are currently the most popular way to listen to music. According to the 'International Federation of the Phonographic Industry', streaming generates 62.1 % of the recording industry revenue worldwide [otPI20]. The importance of streaming continues to expand with paid streaming income increasing by 18.5 % over 2019. Nowadays, the paid subscription account users are about 443 millions. Streaming has enabled a vast pool of music to be easily accessible, with tens of millions of songs available on each streaming provider.

Deezer is an online music streaming service with a vast and ever evolving catalog of heterogeneous multimodal data. An example of the type of data aggregated by Deezer is displayed in Figure 1.1. The Deezer catalog consists of more than 73 million tracks, from over 17 millions of artists, and is enriched with tens of thousands of new titles every day. Such a large collection needs to be organized and qualified automatically to help users browse it and to perform smart and personalized content based recommendation, allowing a more enjoyable global user experience.

To do so, it is crucial to extract meaningful and high-level musical features from tracks such as the genre or the language of music. One of the objectives of streaming platforms is to develop systems capable of computing such features, or descriptors, using available data. These descriptors may be available in metadata, but this type of metadata is often noisy and available only for a small part of the catalog. Audio signals, on the other hand, are always available and more reliable. To extract these descriptors, the streaming services rely mostly on *Music Information Retrieval* (MIR) approaches.

The MIR domain is an interdisciplinary field at the frontier between music theory, computer science, signal processing and perception. This topic is concerned with the analysis, processing, and generation of music. It is notably interested in creating systems capable of computing accurate musical descriptors from the analysis of an audio signal.

One example of a standard MIR task is the detection of the genre of a given music. A classic pipeline to resolve this task is displayed in Figure 1.2. This pipeline is quite general and is applicable to a variety of other MIR tasks. A signal transformation, usually in the time-frequency domain, so as to highlight harmonic and rhythmic patterns, is first computed. From this representation, music features are extracted. These features are used as input of a classification architecture, typically based on machine learning algorithms, to obtain a solution for the problem of genre detection.

A large proportion of recorded music has lyrics, especially in popular music. In the Deezer catalog, songs with lyrics represent around 90 % of the five million most listened tracks. Nevertheless, the lyrics have been clearly under-exploited in MIR tasks. Recently, there has been a growing interest in the use of lyrics for the extraction of meaningful musical features. Lyrics constitute a key source of music information as they encode a significant part of the song's semantics [MMC$^+$05].

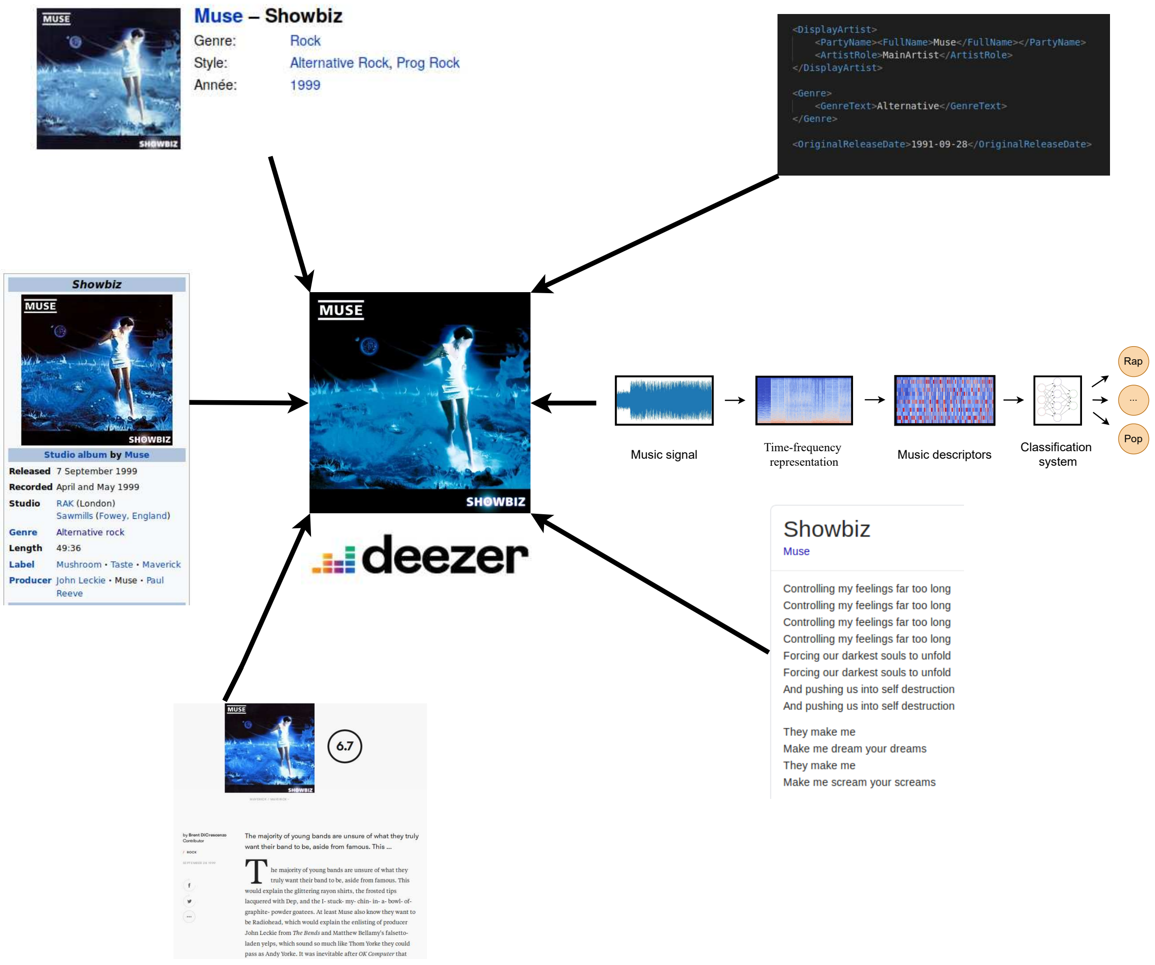
Streaming services' interest in accessing song lyrics is multifold. On the one hand, they allow specific applications such as Deezer's karaoke function. On the other hand, they also represent an important source of information for the qualification of their catalog (*e.g.* through genre and mood classification, language detection or topic modeling). Lyrics are available for about 3.7 millions of tracks in the Deezer catalog. For 20% of these tracks, songs are annotated with start and end times of each lyrical line. This type of information is generally provided externally by manual annotators, making the acquisition slow, costly, covering only a fraction of the catalog and of variable quality. One solution to this problem could be to extract lyrics directly from audio using *Singing Voice Recognition* (SVR) algorithms.

Beyond the transcription task, various subtasks fall under the scope of the SVR field. They can be seen as preliminary tasks that when solved will pave the way to a successful transcription system. Each subtask has, in itself, potential industrial applications for Deezer. The most studied of them is the lyrics-to-audio alignment one [SDE19, GYL20]. This task consists in aligning the text of lyrics of a song to the audio using various granularity for the time precision (at the level of the line, the word, the phoneme ...).

The detection of keywords or entities from audio is also a well studied topic [Kru14a, Kru16a, Kru16b]. It consists in detecting entities (*e.g.* concept, personality name) or keywords (*e.g.* offensive or sexual words). Applications are numerous: discovering songs on particular topics, genre classification, playlist generation, etc. Finally, the characterization of the music from lyrics information extracted from the audio also represents an interesting task. This task can take several forms such as, the genre or mood detection or the detection of the language of lyrics from the audio [KAD14].

1.2 Challenges

Two types of challenges are encountered in this thesis. First, considering industrial size music catalogs implies taking into account the scalability of developed systems and the



Muse – **Showbiz**

Genre:	Rock
Style:	Alternative Rock, Prog Rock
Année:	1999

```xml
<DisplayArtist>
    <PartyName><FullName>Muse</FullName></PartyName>
    <ArtistRole>MainArtist</ArtistRole>
</DisplayArtist>

<Genre>
    <GenreText>Alternative</GenreText>
</Genre>

<OriginalReleaseDate>1991-09-28</OriginalReleaseDate>
```

Showbiz

Studio album by Muse

Released	7 September 1999
Recorded	April and May 1999
Studio	RAK (London) Sawmills (Fowey, England)
Genre	Alternative rock
Length	49:36
Label	Mushroom • Taste • Maverick
Producer	John Leckie • Muse • Paul Reeve

Showbiz
Muse

Controlling my feelings far too long
Controlling my feelings far too long
Controlling my feelings far too long
Controlling my feelings far too long
Forcing our darkest souls to unfold
Forcing our darkest souls to unfold
And pushing us into self destruction
And pushing us into self destruction

They make me
Make me dream your dreams
They make me
Make me scream your screams

6.7

by Brent DiCrescenzo
Contributor

/ ROCK

The majority of young bands are unsure of what they truly want their band to be, aside from famous. This ...

The majority of young bands are unsure of what they truly want their band to be, aside from famous. This would explain the glittering rayon shirts, the frosted tips lacquered with Dep, and the I- stuck- my- chin- in- a- bowl- of- graphite- powder goatees. At least Muse also know they want to be Radiohead, which would explain the enlisting of producer John Leckie from *The Bends* and Matthew Bellamy's falsetto-laden yelps, which sound so much like Thom Yorke they could pass as Andy Yorke. It was inevitable after *OK Computer* that

Figure 1.1 – Deezer collects heterogeneous, multi-modal data.

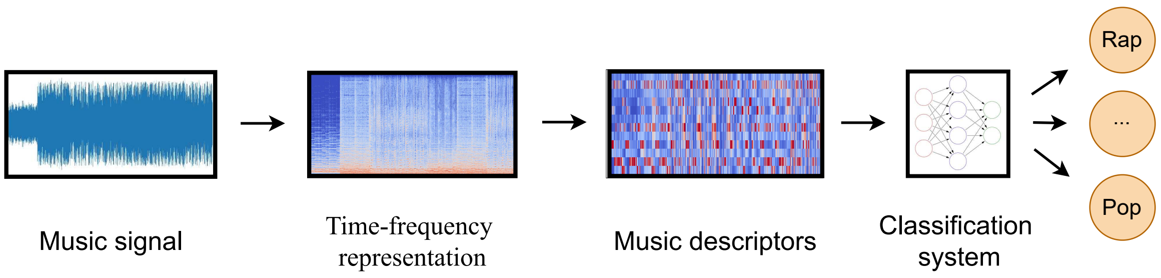


Figure 1.2 – A classic MIR pipeline for the genre detection of music.

Property	Speech	Singing
vowel content	60 %	90 %
vowel duration	speaking rate	note length, composition
pitch in a phrase	fairly constant	varies with notes, composition
pitch range	semitones	octaves

Table 1.1 – Principal distinctions between speech and singing adapted from [Mes13]. Copyright © 2013, IEEE. Permission granted for adaptation.

diversity of cases that can be encountered. For instance, such a catalog contains music from several genres and various languages, some of them being even multilingual. Second, the specific goal of lyrics transcription remains complex, the best *Word Error rate* (WER) obtained being as high as 50% (i.e. one word out of two false), with particular limitations that do not occur in speech transcription. We describe these limitations broadly in the following paragraphs.

Properties of singing voice: Properties of the singing voice greatly differ from those of the spoken voice. In speech, pitch, intensity and rhythm express emotions. Moreover, prosody (related to pitch) and punctuation (connected to rhythm) both have an essential semantic function which can change the meaning of a phrase. In the singing voice case, all these characteristics are constrained by the melodic composition. Due to its musical dimension, phonetic variability is greater in singing than in speech. For example, the duration of phonemes can vary vastly with syllables which can last for several seconds at the end of a chorus [Kru18].

Pitch range can also be extremely large. It represents another significant factor of variability and can make phonetic identification difficult (e.g. for high notes where formants do not appear clearly). Another factor of variability is the fact that voice style is largely dependent on music genre. Also, singers use special techniques, like vibrato, making singing signals even more complex. Finally, post-production effects applied on the voice, like reverberation and auto-tune, can also be considered as sources of variation. Main differences between speech and singing are described more precisely in Table 1.1.

Music is polyphonic: Music (*i.e.* accompaniment) can be considered as an important background noise for the signal of interest (*i.e.* singing voice). It is usually present with a large volume and is highly correlated with the signal of interest: singers usually sing in harmony (frequency correlations) and in rhythm (time correlations) with the music. The assumption of noise independence made in most *Automatic Speech Recognition* (ASR) systems is violated here.

Historically, the first SVR algorithms were trained and tested on *a cappella* [HSIM05]. However, their performance was shown to drop consequently when applied on polyphonic signals [MV10a]. In this case, a preprocessing step of *Singing Voice Separation* (SVS) algorithms during inference helps to partially recover the performance [SGLW19, MV10a]. Recent improvements of SVS algorithms [JHM$^+$17] have been shown to further improve results [SGLW19]. Moreover, in [MVK07], the authors demonstrate that singer identifi-

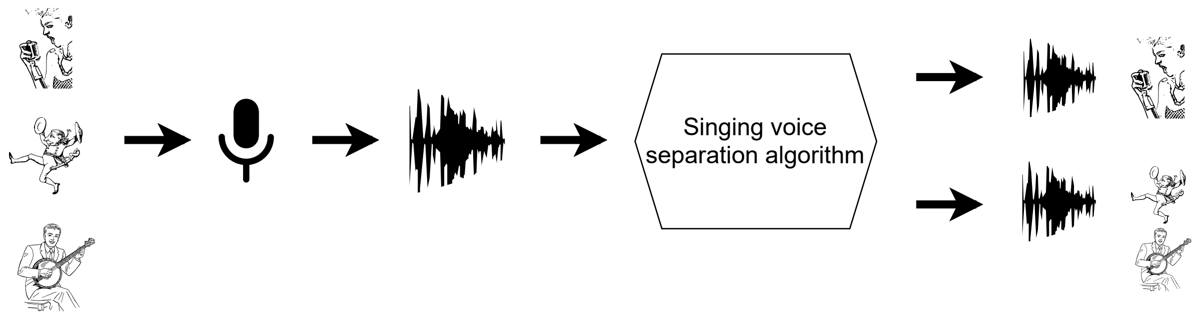
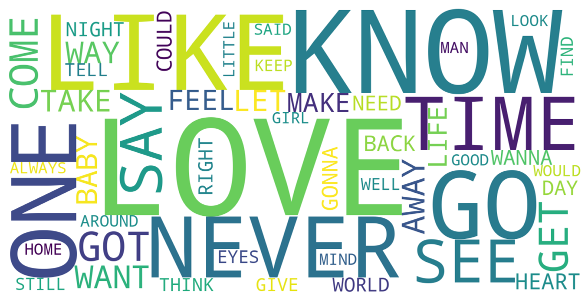


Figure 1.3 – General principle of SVS.


(a) Top words DALI [MBCHP20].

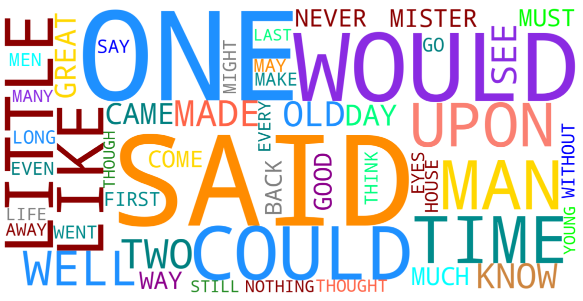

(b) Top words Librispeech-100 [PCPK15].

Figure 1.4 – Most frequent words after discarding stop words in a lyrics dataset (left) and in a prose dataset (right). Size of the representation is proportional to the word frequency.

cation is robust up to −5 dB signal-to-noise ratio using a preprocessing step of SVS. The general principle of a SVS algorithm is described in Figure 1.3.

This algorithm takes as input a recording and decomposes it into two tracks: the first one being composed of the estimated singing voice and the latter of the extracted accompaniment. In this thesis, we only focus on polyphonic data for training and evaluation. We consider this to be a reasonable decision as most of the commercially available songs are polyphonic. For our systems, we will then consider a preprocessing step of SVS for both training and inference. The impact of this preprocessing, in this case, is yet to be discussed. A complete overview of the influence of SVS in this context is given in Section 2.3.

Language model properties: Statistical properties of language models constructed from lyrics text are highly different from those constructed from speech transcripts. Actually, the vocabulary employed in singing is quite specific and often much more restricted than in speech [Mes12]. Moreover, these properties also vary according to the musical style [MNR08] (*e.g.* slang in rap). To compare both types of datasets, the most frequent words of one lyrics database and one prose database are displayed in Figure 1.4. It can be seen that the lexical fields of the lyrics text database are mostly focused around love ("love", "eyes", "heart" ...), which does not seem to tbe the case on the speech transcript database.

1.3 Research objectives and contributions

1.3.1 Main research objective

Even if the performance of SVR architectures remains far from perfect from a pure transcription metrics perspective, questions remain about the lyrical information that is nonetheless captured using these techniques and how useful it can prove for adjacent, lyrics-related applications. Thus, the primary research goal of this thesis is to investigate whether lyrical information taken directly from audio can be employed to execute a range of MIR tasks. To do so, we use various state-of-the-art lyrics recognition implementations.

We focus on a large publicly available annotated dataset for training to ensure reproducibility, each system being also evaluated on public datasets whenever possible. Moreover, to test the limit of our approaches, we also evaluate them on bigger, more diversified and more complex internal datasets. Indeed, we keep in mind the context of the thesis: categorizing industrial size catalog with large variety of data of various complexities. To that end, we will make systems, to the largest extent possible, scalable and usable in real life situations.

1.3.2 Research questions

Four topics are researched for this thesis, each having potential important industrial applications for Deezer: explicit content detection, lyrics-to-audio alignment, language identification and cover detection. Multiple research questions can be defined from these subjects:

Explicit content detection: Detecting a music recording as being explicit or not has always been performed using preexisting lyrics. The question of whether such information can be extracted directly from audio to complete the task remains unresolved. To take it a step further, one might wonder if such a system could be deployed automatically without human input.

Lyrics-to-audio alignment: The task of aligning lyrics to audio has been defined in Section 1.1. Recent systems have attained great performance on English datasets. This raises the question of their suitability for usage on a large-scale music catalog. More specifically, if these systems are still efficient on non English music and if they handle multilingual lyrics. If not, the salient elements that might improve multilingual generalization of such systems should be investigated. Finally, the perceptual validity of the metrics used to quantify the performance of these systems is unclear, which necessitates additional experiments and discussion.

Language identification: Singing language identification algorithms have traditionally been tested on languages present in the training dataset. It may be explained as the majority of these systems' goal is to be able to detect the most frequent languages. When applied to a large catalog, however, such algorithms must frequently infer labels for musics where the matching language is not present during training. In this scenario, the system should be able to identify these instances as 'out-of-set.' This begs the question of how language recognition algorithms could handle these cases.

Cover detection: Cover detection, also known as version identification, attempts to determine whether two recordings are of the same musical composition. This task is generally performed using tonal features computed from audio. The question of whether lyrics information extracted from audio could be used to accomplish the task remains unanswered. Furthermore, the complementarity of both information, lyrics and tonal ones, should be studied. One could ask if using both information sources would be preferable over utilizing only one.

1.3.3 Contributions

Following the main research objective, and the various questions that we wish to address, multiple contributions are presented in our work:

Explicit content detection: This thesis presents the first system for explicit content detection directly from audio. Our modular approach uses an audio-to-character recognition model, a keyword spotting model associated with a dictionary of carefully chosen keywords and a Random Forest classification model for the final decision. The results obtained are encouraging with a F1-score of 67% on an industrial scale explicit content dataset.

Lyrics-to-audio alignment: In this thesis, we present the first attempt to create a language-independent lyrics-to-audio alignment system. To do so, we first focus on one state-of-the-art architecture that seems fit for generalization. Then, we study the relevance of different intermediate representations, either character or phoneme, along with several strategies to design a training set. The evaluation is conducted on multiple datasets, from diverse sources, languages and scripts, with a varying amount of data available, from plenty to zero. Notably, we consider the case of multilingual songs.

The results show that the use of diverse data for training and of an universal phoneme set as an intermediate representation yield the best multilingual generalization performance. Additional experiments to take into account phoneme similarity during alignment are also presented. Finally, we investigate the perception of audio and lyrics synchrony through two realistic experimental settings inspired from karaoke, and discuss implications for the lyrics-to-audio alignment evaluation metrics. The most striking features of these results are demonstrated to be the asymmetrical perceptual thresholds of synchrony perception between lyrics and audio, as well as the influence of rhythmic factors on them.

Language identification: In this thesis, we propose a new modernized phonotactic system using recurrent models for both phoneme recognition and language classification. Phoneme recognition is performed with a *Connectionist Temporal Classification* (CTC) based acoustic model trained with multilingual data, before language classification with a recurrent model based on the phoneme estimation. The full pipeline shows unprecedented performance. First results of singing language identification with out-of-set languages are also presented.

Cover detection: In this thesis, we propose the first cover song detection system explicitly leveraging lyrics information from audio. Our approach relies on the fusion of a SVR framework and a more classic tonal-based cover detection method. Furthermore, we exploit efficient string matching and an approximated nearest neighbor search algorithm which leads to a scalable implementation which is able to operate on very large databases. Extensive experiments on the largest publicly available cover detection dataset demonstrate the validity of using lyrics information for this task.

1.4 Thesis structure

To enhance readability, the first four chapters are collected in the first part **'Extracting lyrics information from audio'**. This part basically describes the whole context surrounding our thesis, being theoretical or practical. The first chapter is the current chapter where the general context, the challenges and the objectives of the thesis are described. Following this first introduction chapter, the existing literature for the SVR domain and the datasets available for the task are given in Chapter 2. Then, all the technical background necessary to create SVR systems is detailed in Chapter 3. Given both previous chapters, a first vanilla lyrics transcription system is implemented in Chapter 4. It serves as the reference for most SVR pipelines used in the following chapters.

The last remaining chapters are collected in the second part **'Applications to MIR'** which encompasses all our contributions and the conclusion. Explicit content detection, lyrics-to-audio alignment, singing language identification and cover detection subjects are respectively studied in Chapter 5, Chapter 6 and 7, Chapter 8 and Chapter 9. The lyrics-to-audio alignment topic is researched in two consecutive chapters. The first one focuses on investigating the multilingual generalization of lyrics-to-audio alignment systems whereas the second one focuses on the perception of lyrics-to-audio synchrony. Finally, in Chapter 10, we provide an overall summary of the work and contributions discussed in this thesis. The main limitations and possible future directions of research are presented.

1.5 Publications and seminars

In this section, we present publications and seminars which occurred during the PhD thesis. For all publications where the PhD student is the first author, all code and redaction have been made by the PhD student. Participation of the PhD student to other papers is described in the publications list given below. All publications are available in Annexe B.

1.5.1 Publications

Laure Prétet, Romain Hennequin, Jimena Royo-Letelier, Andrea Vaglio. Singing voice separation: A study on training data. *In Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2019. In this paper, the PhD student mainly helps reviewing the paper written by the first author and doing statistical analysis of the results.

Andrea Vaglio, Romain Hennequin, Manuel Moussallam, Gaël Richard and Florence d'Alché-Buc. Audio-Based Detection of Explicit Content in Music. *In Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

Andrea Vaglio, Romain Hennequin, Manuel Moussallam, Gaël Richard and Florence d'Alché-Buc. Multilingual Lyrics-to-Audio Alignment. *In Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2020.

Lenny Renault, Andrea Vaglio, Romain Hennequin. Singing Language Identification Using a Deep Phonotactic Approach. *In Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2021. The first author of the paper was an intern supervised by the PhD student as a part of his thesis. Code of experiments and writing of the article were both accomplished by the intern and the PhD student.

Ninon Lize Masclef, Andrea Vaglio, Manuel Moussallam. User-Centered Evaluation of Lyrics-to-Audio Alignment. *In Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2021. The first author of the paper was an intern supervised by the PhD student as a part of his thesis. Code of experiments was done by the intern. Writing of the article was accomplished by the intern and the PhD student.

Andrea Vaglio, Romain Hennequin, Manuel Moussallam, Gaël Richard. The Words Remain the Same: Cover Detection with Lyrics Transcription. *In Proc. of Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2021.

1.5.2 Seminars

Andrea Vaglio. Detection of Explicit Content in Music. *Seminar at DeepLearn*, 2019

Andrea Vaglio. Audio-Based Detection of Explicit Content in Music. *Seminar at Data and Research Deezer seminar*, 2019

Andrea Vaglio. Automatic Lyrics Transcription. *Seminar at Kaggle Paris meetup #22*, 2019

Andrea Vaglio. Multilingual Lyrics-to-Audio Alignment. *Seminar at Data and Research Deezer seminar*, 2020

Chapter 2

Lyrics transcription methods and datasets

Singing Voice Recognition (SVR) systems are usually inspired by approaches from the *Automatic Speech Recognition* (ASR) field. This domain is a relatively mature research area where impressive results have been recently achieved thanks to the latest developments in deep neural networks [AAA+16]. However, speech recognition algorithms lead to unsatisfactory results when adapted to singing voice with the state-of-the-art *Word Error rate* (WER) being around 50% [DAD21].

In this chapter, we will present a broad overview of the ASR domain in Section 2.1. Then, an extensive literature of automatic lyrics transcription will be presented Section 2.2. We will focus on the problems encountered in the field, how the systems are transferred from ASR ones, and notably how they are adapted to the specificities of the singing voice. Next, we will discuss the impact of a *Singing Voice Separation* (SVS) preprocessing step on SVR systems Section 2.3. After presenting the annotated singing voice datasets Section 2.4, we will finally conclude Section 2.5.

2.1 A brief history of ASR systems

ASR systems are traditionally composed of multiple modules applied sequentially on the audio signal like the feature extraction one, the acoustic model, the pronunciation lexicon and the language model. When decoding a given input audio signal, all modules are used to obtain an estimated transcript. An extensive description of each block is given in the next chapter. A brief explanation will then be provided here for each module. The feature extraction module aims at computing mid-to-high level representations, or descriptors, from the input signal.

An acoustic model describes the association between an audio signal and considered linguistic elements (e.g. phonemes or words) composing the speech signal. The pronunciation lexicon is a set of words as well as their corresponding pronunciations as defined by a particular pronunciation alphabet. A language model defines a probability distribution over sequences of words or any other linguistics units (e.g. characters or words).

The acoustic model is generally the bottleneck of these systems, *i.e.* poor performance in this module drastically limits the overall pipeline performance. A large part of the literature thus focuses on it. Historically, acoustic models for speech use a combination of *Gaussian Mixture Model* (GMM) and *Hidden Markov Models* (HMM) [Rab89]. For a long time, deep neural networks were only used to obtain input features for this type of architecture (*e.g.* using tandem architecture [HES00] and bottleneck features [YHX13]).

In 2012, a paradigm shift was observed with the introduction of hybrid acoustic models [HDY+12]. Taking advantage of advances in the field of deep neural network training since the eighties (*e.g.* in initialisation and optimisation) and from the large increase in availability of computer resources and vast amount of data, the authors designed and trained large *Deep Neural Network* (DNN)-HMM.

These methods outperformed traditional GMM-HMM approaches, offering a new set of possibilities for deep neural networks in speech. Using this new paradigm, impressive results were quickly obtained by training systems on massive amounts of data, taking use of both model and data parallelism. Improvements have been steadily observed since then, achieving human parity [XDH+16] for some evaluation datasets and reaching results as low as 5% of WER on others [XWA+18], *i.e.* about five wrong words each 100 spoken words. Similarly, language models transited from simple n-gram models to neural network based language models [MKB+10].

The multiple modules of a classic ASR pipeline make the development of such systems complicated and the decoding algorithm complex. These modules are generally trained separately and hard to optimize jointly. After the deep neural network paradigm shift, new approaches trying to reduce this complexity employing *End-To-End* (E2E) acoustic models became trendy.

Such acoustic models are only based on a single DNN, taking as input low level features (*e.g.* audio or time-frequency representations) and directly outputting sequences of characters. This type of approach alleviates the need of a pronunciation lexicon or of feature engineering. When compared to ASR systems employing hybrid acoustic models outputting sequences of phonemes, these approaches have been shown to be more data greedy but yet capable of getting equivalent results, with a simpler framework, when a large volume of data is used to train both systems [GJ14].

One of the first popular E2E acoustic models is based on the *Connectionist Temporal Classification* (CTC) implementation described in [GJ14]. It has since been surpassed by the attention model that allows also, in contrast to the CTC model, to learn a language model over data. In [BCC+17], it is notably demonstrated to outperform a system using a CTC based acoustic model combined with a language model. The most recent trend in E2E approaches is transformer based acoustic models, which have been demonstrated to obtain great results in [KCH+19]. These architectures are based on multiple layers of encoder-decoder attention and self-attention.

2.2 An overview of automatic lyrics transcription literature

As described in the previous chapter the challenges encountered are two-fold. Foremost, considering an industrial context, the diversity of data that SVR systems are expected to handle is consequent with a large pool of genre, production style and language. Robust systems being able to generalize to a variety of different contexts must then be developed. Furthermore, the lyrics transcription task is challenging with specificities over speech transcription that need to be accounted for.

As for ASR systems, the acoustic model is the main bottleneck of SVR approaches. Most works thus focus on training robust acoustic models and adapting them to the characteristics of the SVR domain. In the rest of the document, we will refer as a classic system of SVR a system using all the standard modules described in the previous section (e.g. feature extraction, lexicon, acoustic model ...) by opposition to SVR systems based on an E2E acoustic model.

2.2.1 Creating a robust singing voice acoustic model

Adapting ASR trained systems: In early studies, the development of SVR approaches was hindered by the lack of publicly available annotated datasets at phoneme, word or even line level. A line is defined as a segment of lyrics bounded by a line feed. In speech recognition, acoustic models are traditionally trained on large annotated datasets of several hundred to several thousands of hours of data, segmented at the utterance level, to obtain the performing results [AAA^{+}16]. Given the more complex nature of singing voice over speech voice, we can assume SVR approaches are at least as data demanding.

First studies attempt at adapting ASR trained systems to singing to overcome this problem. More specifically, some architectures were trained on speech data and adapted to singing using speaker adaptation techniques. A small singing dataset manually annotated was generally used to perform adaptation.

For instance, in [MV10b], a monophone HMM is trained on speech data and adapted to singing using a small corpus of manually annotated *a cappella* songs, at paragraph level, by using a *Maximum Likelihood Linear Regression* (MLLR) algorithm. The transcription is then performed on polyphonic songs after extracting the singing voice using a pitch-based inference combined with a background model subtraction. This singing voice separation algorithm is extensively described in [VMR08]. A performance of 94% WER is reported for this system on a small dataset of English pop songs. More generally, SVR systems adapted from ASR ones displayed disappointing results.

Leveraging *a cappella* dataset: A first attempt to automatically generate an annotated singing voice dataset from *a cappella* is described in [Kru18]. In this paper, Kruspe et al. trained an acoustic model on 'low quality' automatic annotations generated with forced alignment using a pretrained ASR system. More precisely, a speech recognizer is used to generate a huge amount of singing annotation by aligning a large corpus of *a* *cappella* singing to their corresponding lyrics, at the word and phoneme levels. Obtained annotations are afterwards used to train a new acoustic model, showing a significant improvement over the speech recognizer when evaluated on a large *a cappella* dataset.

Following this work, multiple methods to generate this type of large scale solo-singing dataset, at line level, but with higher quality of annotation, were proposed recently [GTLW18, DB19]. These datasets are not publicly available but can be easily rebuilt from data available online using methods described in these papers. Several systems, trained on these datasets, were subsequently proposed for the latest lyrics transcription *Music Information Retrieval Evaluation eXchange* (MIREX) challenge.

The system described in [DAD21], noted Demirel21 in this thesis, is for example trained on the dataset DSING30. The architecture is based on a dilated convolutional *Time Delay Neural Network* (TDNN) with self-attention layers. A preprocessing step of SVS is performed using Spleeter [HKVM20]. A segmentation step is also applied before lyrics transcription. To do so, an algorithm spots multiple anchor words position in the audio. Lyrics transcription is carried out directly on these segmented recordings, employing a 4-gram language model trained on lyrics. Transcription hypotheses are finally rescored using a *Recurrent Neural Network* (RNN) language model to output the desired estimated transcription. This system shows great results, attaining a WER of 49.92% on the Mauch polyphonic dataset.

Training on polyphonic dataset: Recently, multiple systems have been trained on large polyphonic singing voice datasets, either private or public, annotated at word or phrase levels, leading to significant progress on lyrics transcription. Notably, the recent release of the DALI polyphonic dataset has paved the way to a massive breakthrough, allowing free access to an annotated singing voice dataset of significant size for all researchers.

These systems include the approach presented by Stoller et al. [SDE19]. This architecture will be noted as Stoller19 for the rest of the document. It is based on an E2E audio-to-character architecture, more precisely a wave-U-net. This acoustic model is displayed in Figure 2.1. A preprocessing step of SVS is performed, during training and inference, using a U-net convolutional network. The acoustic model is trained on a private English dataset of 40000 songs using a CTC algorithm. Employing a trigram word-level language model trained on lyrics, it obtains a WER of 77.8% on the Jamendo polyphonic dataset.

Another architecture demonstrating great performance is the one described in [GYL20] by Gupta et al. This approach will be referred to as Gupta20 in the rest of this thesis. It utilizes an acoustic model composed of several layers of TDNN trained using the English tracks of the DALI dataset. It uses an extended pronunciation lexicon to cope with long vowel duration in singing and genre labelling information (phoneme units are annotated with music genres) but does not rely on a preprocessing step of SVS. Finally, a trigram word language model is trained on the English portion of DALI lyrics. A version of this system obtained the best results in the MIREX 2020 lyrics transcription challenge[1] reaching a WER of 60.98% on the Jamendo dataset.

1. https://www.music-ir.org/mirex/wiki/2020:MIREX2020_Results

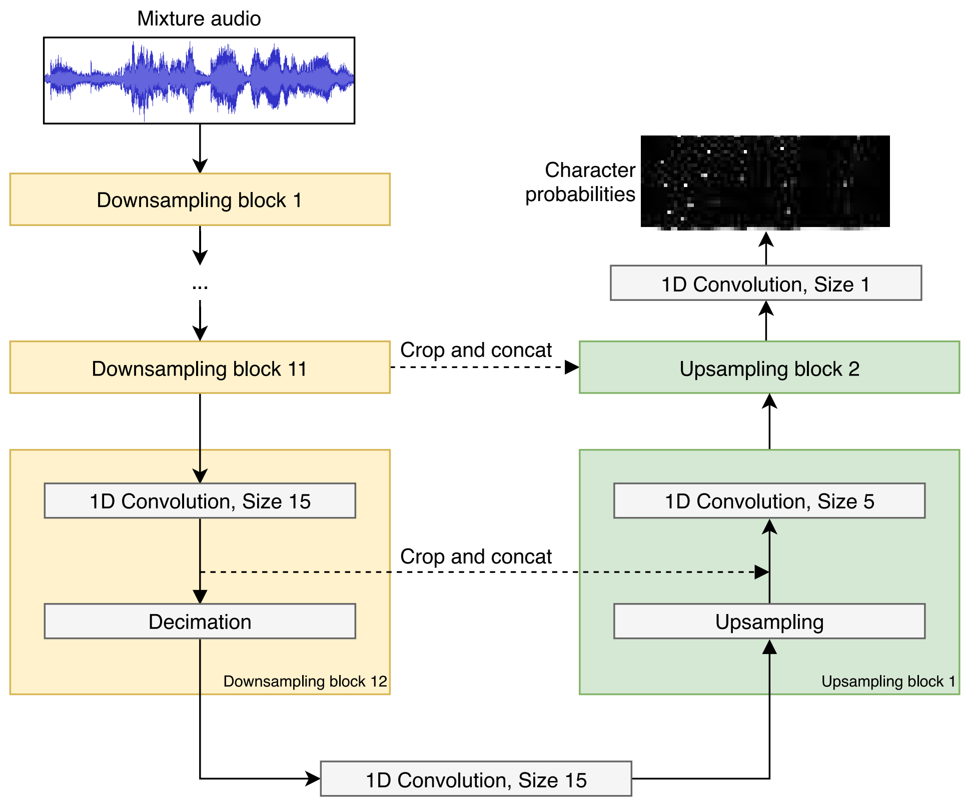


Figure 2.1 – Stoller19 acoustic model reproduced from [SDE19]. Copyright © 2019, IEEE. Permission granted for reuse.

2.2.2 Adapting acoustic model to SVR's specificities

Some studies try to adapt modules of a classic ASR pipeline to overcome the vowel duration differences between singing and speech. For example, a duration-explicit HMM is used in [Dzh17], extending a classic Markov-based acoustic model to handle long duration states. This type of adapted Markov model is notably informed by vocal sheet music in [DS15] or by metric principles of a considered music tradition in [DYRS16]. Also, the authors in [SGLW19] employ a duration-based modified pronunciation lexicon.

As described in Section 1.2, the performance of acoustic models trained using *a cappella* are known to drop when applied to polyphonic music. Attempts have been made to improve this performance regarding background music using various preprocessing steps such as a SVS algorithm [Mes13, DS15] or a voice activity detection module [FGO$^+$06]. Other methods to improve the acoustic model performance with regard to background music have included the use of aligned chords with lyrics from crowdsourcing websites [MFG12].

2.2.3 Comparing state-of-the-art systems

Performances on multiple polyphonic evaluation datasets of state-of-the-art systems for lyrics recognition are summarized in Table 2.1. Gupta20 significantly outperforms Emir21

System	Dataset	Mauch	Jamendo
Stoller19	Internal	70.9 (x)	77.8 (x)
Gupta20	DALI	**47.25 (3.67)**	**60.98 (3.04)**
Demirel21	DALI	49.92 (x)	**44.52 (x)**

Table 2.1 – State-of-the-art lyrics recognition systems performance on various evaluation datasets with their training dataset. The column named "Dataset" refers to the dataset used to train the considered system. The columns "Mauch" and "Jamendo" refer to the dataset employed to study the proposed systems performance. The WER metrics are given in percentage and the standard errors are displayed in parentheses when available. Gupta20 results are taken from the 2020 lyrics transcription challenge of MIREX. The best results for an evaluation dataset are displayed in bold.

on the Mauch dataset, attaining a WER of 47.25%, but falls behind it on the Jamendo dataset, shifting the WER from 44.52% to 60.98%. If the better performance of Demirel21 over Gupta20 is hard to conclude, both are clearly outperforming Stoller19 on all evaluation datasets.

One possible explanation is that the latter is based on an E2E acoustic model. As shown in Section 2.1, recognition architectures based on E2E acoustic model are more data demanding than classic pipelines such as Gupta20 and the Emir21. Thus the size of current available singing voice annotated datasets could be too low for systems based on an E2E acoustic model to obtain similar results to classic SVR systems. Moreover, Stoller19 is trained by directly taking as input the waveform of the training dataset. Systems trained using this sort of data are also known to be data-demanding to work properly [PNP+17].

The superiority of Gupta20 over Stoller19, the two models trained on polyphonic data, could be further explained by the difference of training annotation quality. In fact, whereas the quality of annotations of the DALI dataset used to train Gupta20 is high, Stoller19 is trained on a proprietary dataset with inferior line-level granularity, of unknown features and unknown quality.

2.3 Impact of singing voice separation

As described in Section 1.2, in this thesis, we focus on considering only polyphonic data for both the training and evaluation of our models. The influence of a SVS preprocessing during both training and evaluation in this case is yet to be studied.

In [GYL20], Gupta et al. show that the performance of a standard SVR pipeline is improved on lyrics transcription when taking polyphonic music as input directly over using a preprocessing step of SVS. Significant improvements are observed on all evaluation datasets. On the Jamendo dataset, for example, the WER is reduced from 71.83% to 66.58%. It could be explained since, even with the most recent SVS algorithms, artifacts and distortions are created in the extracted singing voice which may affect the performance of trained acoustic models.

Name	Type	Quality	Language	# Tracks	Granularity
DALI [MBCHP20]	Poly	Pro	Multiple	5358	Word
Deezer	Poly	Pro	Multiple	845k	Line
DSING30 [DB19]	Acap	Amat	English	4324	Line
Hansen [Han12]	Poly	Pro	English	9	Word
Mauch [MFG12]	Poly	Pro	English	20	Word
Jamendo [SDE19]	Poly	Pro	English	20	Word

Table 2.2 – Various state-of-the-art datasets for the lyrics transcription task are described. The term "type" relates to the nature of the recordings in the dataset under consideration, which might be either *polyphonic* or *a cappella*. The term "Quality" indicates the quality of the recordings in the associated dataset: *professional* or *amateur*. The term "granularity" refers to the lowest level of annotation accessible for the corresponding dataset.

In comparison, Basak et al. exhibit an improvement in lyrics transcription performance for their system by applying a preprocessing step of SVS in [BAGT21]. This is understandable given that the system is built on an E2E acoustic model, more precisely a transformer. This sort of system being data-demanding, it is likely that they have difficulty modeling both the voice and the accompaniment characteristics when trained with insufficient data. A SVS preprocessing step thus helps ease constraints on the acoustic model by removing part of the complexity of the input signal to model. On a subset of the DALI dataset, the WER is decreasing from 60.4% to 57.4%. The system being trained with a dataset of thousands of tracks, we assume the presence of the background music could help increase the performance of this sort of acoustic model when trained using a much larger dataset.

Using both the extracted voice track and the entire mix for training and inference of SVR algorithms is not explored in this thesis and might be an interesting direction for future work. Indeed, both signals presenting different difficulties, their combined usage could improve the efficiency of trained systems by providing them complementary information.

2.4 Datasets

In this section, the state-of-the-art datasets for the lyrics transcription task are extensively described. An overview of all datasets presented in this section are displayed Table 2.2.

2.4.1 DALI dataset

The DALI dataset is the first large publicly annotated singing voice dataset available. It contains 5358 audio tracks with time-aligned lyrics at paragraph, line and word levels. It is composed of varied western genres (*e.g.* rock, rap and electronic) in several languages. It is extensively described in [MBCHP20]. This dataset is generated automatically using a teacher-student paradigm from manual annotations found on the internet. In this section, how this dataset is generated is described in detail and the quality of the obtained annotations are discussed.

To create this dataset, the authors use a set of lyrics manually aligned by non-expert users of karaoke games from various sources. This set of annotations being however deprived of audio recording, a set of candidate recordings are retrieved from the internet. A singing voice detection system, based on a deep neural network architecture, is then applied to each candidate. The result is finally compared with the annotations to find the best candidate for each annotation file and update the annotations according to it. A correlation threshold is used to discard badly matching cases.

A student-teacher paradigm is employed to improve the results. More precisely, a first singing voice detection architecture is trained, on a small manually annotated ground truth dataset, and utilized to generate a new set of updated annotations from the ones found on the web. These new annotations are afterwards used to train a singing voice detection system student that again performs the above comparison and updates annotations. This process can be repeated iteratively. The authors demonstrate that the teacher-student paradigm improves the accuracy of the trained singing voice identification system at each iteration. Two parameters are estimated to update annotations: the offset and the warping parameters. The first defines the start of the annotations, while the second adjusts the time grid size by compressing or extending all annotations at once.

On 105 tracks, the authors manually found the two parameters producing the best global alignment for each pair of audio-lyrics. On this dataset, estimation of parameters is demonstrated to be closer to the ground truth values, each time a teacher-student iteration is performed. It therefore demonstrates the increase of the quality of audio-annotation pairs after each new iteration. However, it cannot quantify the absolute quality of annotations. It would necessitate manually reviewing each annotation, which is exactly what this method attempts to avoid. Looking over the annotations of a hundred songs manually, the resulting annotations appear to be of high quality overall, as similar to what we would manually annotate, which might explain the high performance attained by systems trained on this dataset.

Given overall quality of annotations, this dataset could be used to correctly estimate efficiency of trained systems for SVR tasks. Manually annotated evaluation datasets presented Section 2.4.3 are certainly of better quality annotation, however they are only constituted of a maximum of 20 songs. Metrics evaluated on this type of dataset are thus less representative of the diversity of cases encountered in an industrial dataset in comparison to the one computed on a DALI dataset. Furthermore, they may be extremely noisy, displaying large variations in values between different systems or even between two training of the same system. Finally, these datasets are only composed of songs in English and do not make sense for evaluation in multilingual SVR studies.

2.4.2 Deezer dataset

The Deezer catalog has line-level synced lyrics for around 845 thousand tracks. Around half of them are in English and one-third are in Chinese. The next three most popular languages are Spanish, French and Portuguese. As described in Section 1.1, these annotations are supplied by external providers and are of variable quality. Moreover, for a given track the audio used by the provider to make the annotations is unknown. They only furnish to Deezer pairing metadata for each annotated file and matching is done internally to recover the corresponding audio.

Annotated lyrics can then be matched to the wrong audio. Two cases happen. 1) The lyrics and the audio matched are totally unrelated. These cases give particularly bad results during evaluation and are therefore discarded after being manually checked. 2) The lyrics and the matched audio are partially related. It happens when two different versions of the same song are matched, *e.g.* a radio edit and an extended version, or a studio and a live version. A possible consequence is that the audio contains lyrics that do not appear in the lyrics transcript or the other way around.

Moreover, annotations appear to have been offsetted for some tracks to optimize visual presentation over exact synchronization (*e.g.* lyrics displayed a bit earlier are easier to follow in a karaoke setting). Sometimes, this offset increases with time, indicating the track used for annotation and the Deezer audio have two different speeds. All this makes the dataset particularly noisy. Extra care will be taken when evaluating systems on this data by, for example, correcting the offset. Collecting data from the Deezer catalog, various evaluation datasets will be defined in the next chapters.

2.4.3 Additional datasets

***A cappella* datasets:** As described Section 2.2.1 some large *a cappella* datasets were recently introduced [GTLW18, DB19]. All these datasets are constructed from the freely accessible DAMP Sing! dataset comprising lyrics prompts, prompt timings, and accompanying audio. Because there may be variations between the lyrics prompts and prompt timings and what is actually sung in the recordings, an additional procedure is usually described to retrieve the ground truth alignment timings and transcripts of the audio.

The authors in [DB19] notably describe how to construct the DSING30 dataset. The obtained dataset consists of 150 hours of singing voice in English at line level. A partition between train, validation and test subset is also given to create the first reproducible benchmark of lyrics transcription for *a cappella* data. As described in the previous chapter, we assume in this thesis that only polyphonic data are available. The *a cappella* datasets are thus only presented for reference.

Moreover, the DAMP database, from which these datasets are generated, are composed of audio of people recording themselves singing popular songs on their phone. The quality and diversity of the obtained database is thus far from those of an industrial size catalog composed of professionally quality recordings. Even when compared to professional *a cappella* recordings, this sort of amateur recording will not contain multiple vocalists at the same time, nor will include any processing on the voice.

The state-of-the-art system for *a cappella* recognition obtained a WER of 15.65% on the test part of the DSING30 dataset. Given the difference of performance when compared to systems evaluated on polyphonic datasets, this could suggest vocal separation (or at least the modeling of the accompaniment) is an important element of a SVR system. Indeed, DSING30 recordings could be considered as polyphonic tracks where the singing voice has been perfectly extracted. Still, it would need to assume that performance between this dataset and the polyphonic datasets used for evaluation are comparable. In practice, the recordings of DSING30 could be examples less difficult for the lyrics transcription task than the ones of the polyphonic datasets.

Standard polyphonic datasets evaluation: Some polyphonic datasets are considered as classic evaluation datasets for the SVR. Firstly, the Hansen dataset [Han12] is a small dataset of nine pop music songs in English. Secondly, we also consider the Mauch dataset [MFG12] which consists of 20 pop music songs in English. Finally, the Jamendo dataset [SDE19] is composed of 20 music songs in English of ten different western music genres. All datasets are annotated with start-of-word timestamps.

These datasets are used in the MIREX challenge for evaluation of both lyrics-to-audio alignment and lyrics transcription tasks. At the end of the thesis, it has been discovered that Mauch and Hansen datasets both present an overlap with the DALI dataset. All systems trained on the DALI dataset thus certainly present optimistic metrics on these datasets. This overlap is not noted in any of the literature paper presenting systems trained on the DALI dataset and evaluated on these datasets. Thus these might also present optimistic values on them.

2.5 Conclusion

In this part, we attempt to provide suggestions of what is currently the ideal approach to train a lyrics transcription system based on published research as of this thesis's writing date. Beforehand, it is important to note that the literature is continuously and rapidly developing, and that the state-of-art altered significantly throughout the thesis itself. Furthermore, we continue to presume that only polyphonic data is accessible for training and testing.

To begin with, it appears that, given the publicly available annotated singing voice data, traditional systems of SVR outperform systems based on E2E acoustic models. Additionally, given [GYL20], it seems that providing musical information, such as the genre, improves results for this sort of classic system. Next, a SVS preprocessing step appears to decrease the performance for traditional approaches. This type of system seems to be able to enhance transcription by leveraging the relationship between the background accompaniment and the singing voice. Inversely, when applied to systems based on E2E acoustic models, this preprocessing step appears to improve the performance. Because this type of system is data-intensive, it is likely that it may struggle to model both the voice and the background accompaniment when trained with insufficient data.

Finally, the literature suggests that training systems with moderately sized datasets of high-quality annotations are favored over bigger datasets of low-quality annotations. To summarize, the best approach to do lyrics transcription at the moment seems to be: using a conventional pipeline of SVR, without any preprocessing step of SVS, trained on a high-quality annotation dataset and informed by musical information if available.

Chapter 3

Creating a singing voice recognition pipeline

As explained in the previous chapter, the acoustic model is the bottleneck of this system. It predicts the probability of text units, *e.g.* characters or phonemes, through time. Classically, the chosen text units for the acoustic model are phonemes and a pronunciation lexicon is used to map words to phonemes before training. A pronunciation lexicon can be described as a list of words with their possible pronunciations given a considered pronunciation alphabet (*e.g.* the IPA). A classic *Singing Voice Recognition* (SVR) approach is displayed in Figure 3.1.

The input is an acoustic feature matrix $\mathbf{X}$ extracted from the audio and the outputs are the probability of each phoneme in audio through time, known as posteriorgram. More generally, a posteriogram is defined as a matrix describing the likelihood of the considered text units over time. The language model $\mathscr{L}$ provides a probability distribution over the possible combinations of text units. Usually, language models are defined at the word level. Acoustic and language models are habitually treated independently and trained separately.

The decoder provides an estimated transcript $\hat{\boldsymbol{y}}$ for an acoustic feature matrix given an acoustic model, a language model, and if needed a pronunciation lexicon. More formally, the goal of the decoder is to compute the most likely transcript $\hat{\boldsymbol{y}}$ given the input matrix $\mathbf{X}$, such as:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} P(\boldsymbol{y}|\mathbf{X}) \tag{3.1}$$

3.1 Singing voice separation

The goal of a *Singing Voice Separation* (SVS) algorithm is to separate a music recording into two tracks: the singing voice and the instrumental accompaniment. State-of-the-art methods are based on supervised deep learning [HKVM20] and are widely used in SVR systems [BAGT21] as a preprocessing step before feature extraction. An example of a SVS algorithm is given in Figure 3.2. Consider the signal $\mathrm{X} \in \mathbb{R}^N$, of N samples, composed of

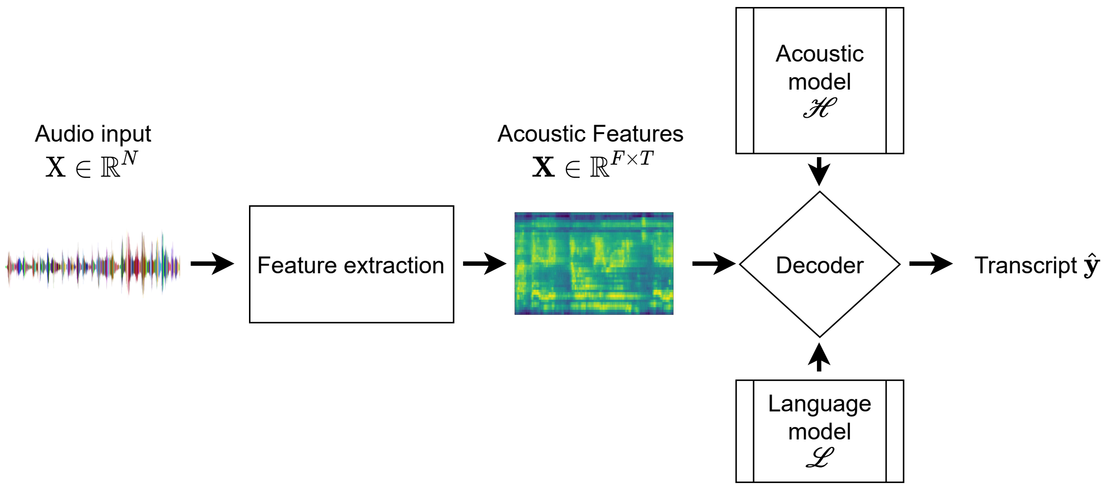


Figure 3.1 – A classic SVR pipeline.

two sources: the singing voice S_{voc} and the accompaniment S_{instr}. Assuming the linearity of the mixture, it can be simply represented as the sum of both sources such as:

$$\mathrm{X}(t) = \mathrm{S}_{voc}(t) + \mathrm{S}_{instr}(t)$$

A *Short Time Fourier Transform* (STFT) is then commonly used to map the mixture from the time-domain to the time-frequency domain. A matrix $\mathbf{X} \in \mathbb{C}^{K \times T}$ with K and T denoting the number of frequency bins and frames, respectively. Knowing the linear property of the STFT, the preceding equation become in the time-frequency domain:

$$\mathbf{X}(k,t) = \mathbf{S}_{voc}(k,t) + \mathbf{S}_{instr}(k,t) \tag{3.2}$$

where $\mathbf{S}_{voc}$ and $\mathbf{S}_{instr}$ are the STFTs of the singing voice and the instrumental signals. In a SVS system, a mask $\mathbf{M}_i(k,t) \in [0,1]^{K \times T}$ is typically computed for each source $\mathrm{S}_i(t)$, where $i \in \{voc, instr\}$, and applied to the mixture to isolate them:

$$\hat{\mathbf{S}}_i(k,t) \;=\; \mathbf{M}_i(k,t) \times \mathbf{X}(k,t) \tag{3.3}$$

where $\times$ is the element-wise multiplication. A SVS system does not generally output the desired masks directly but an estimation of the STFT of each source. Because the phase of the STFT is difficult to predict, the estimation is usually limited to the amplitude of this matrix, the so-called spectrogram. Such SVS systems thus take as input the spectrogram of the mixture and produce two estimated spectrograms, one for each source, $|\hat{\mathbf{S}}|_{instr}$ and $|\hat{\mathbf{S}}|_{voc}$. The desired masks are finally computed, for each source using both obtained spectrograms such as:

$$\mathbf{M}_i(k,t) \;=\; \frac{|\hat{\mathbf{S}}|_i(k,t)}{|\hat{\mathbf{S}}|_{voc}(k,t) + |\hat{\mathbf{S}}|_{instr}(k,t)} \tag{3.4}$$

An estimate of each isolated source's Fourier transform is then obtained by applying each corresponding mask to the mixture, recovering the phase from it, as in Equation (3.3). Finally, by doing an inverse STFT, an estimate of each isolated source $\hat{\mathrm{S}}_i$ is obtained from the corresponding Fourier transform.

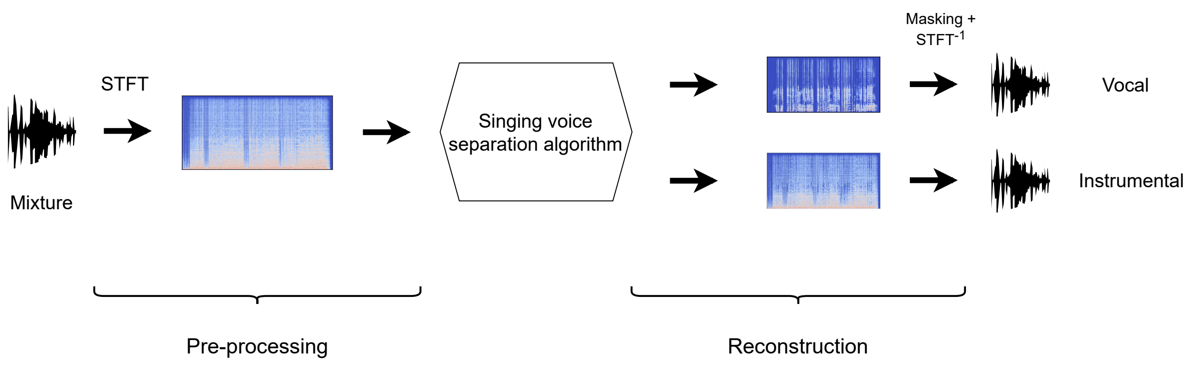


Figure 3.2 – An example of SVS in the time-frequency domain.

3.2 Feature extraction

Mel-spectrogram features are used broadly in the *Automatic Speech Recognition* (ASR) domain [PCZ+19]. They have traditionally been favoured over spectrograms, being more compact, perceptually driven, but still retaining a substantial amount of information. These features are notably more information-preserving than traditional *Mel-Frequency Cepstral Coefficients* (MFCC), another feature commonly utilized [DM80, GYL20].

More formally, consider a raw digital audio signal $\mathrm{X} \in \mathbb{R}^N$ of N samples. To obtain a more sparse representation, a STFT is applied to it. A matrix $\mathbf{X} \in \mathbb{C}^{K \times T}$ with K and T being respectively the number of frequency bins, and the number of frames is thus obtained. Only the modulus (the so-called spectrogram) is kept. The spectrogram frequencies are then mapped from linear scale to mel-scale, known to better approximate human auditory perception [SVN37]. To do so, the mel-scale filter bank $\mathbf{H}(m,k)$ is used. As displayed in Figure 3.3, these filters are triangular band pass filters uniformly distributed on the mel scale. Mel coefficients are afterwards obtained after a logarithmic scaling, such as:

$$\forall m,t \in [\![1,M]\!] \times [\![1,T]\!], \quad \mathbf{mel}(m,t) = \log(\sum_{k=1}^{K} \mathbf{H}(m,k) \cdot |\mathbf{X}(k,t)|) \tag{3.5}$$

The obtained representation is the desired mel spectrogram where M is representing the number of filters employed, generally 40.

3.3 Acoustic model

Usually, an acoustic model $\mathscr{H}$ is defined as a model taking as input an acoustic feature matrix $\mathbf{X}$ extracted from the considered audio and outputting a posteriorgram $\mathbf{R}$, *i.e.* a matrix describing the probability of each considered text unit through time (e.g. phoneme or character). The acoustic model processes the feature vector input sequence frame by frame, generating the appropriate text unit probability vector for each time frame. Classic *Deep Neural Network* (DNN)s can be used as an acoustic model [HDY+12].

However, recurrent networks are typically preferred since these models take prior time frames into consideration by utilizing a persistent hidden state value [GMH13]. It is

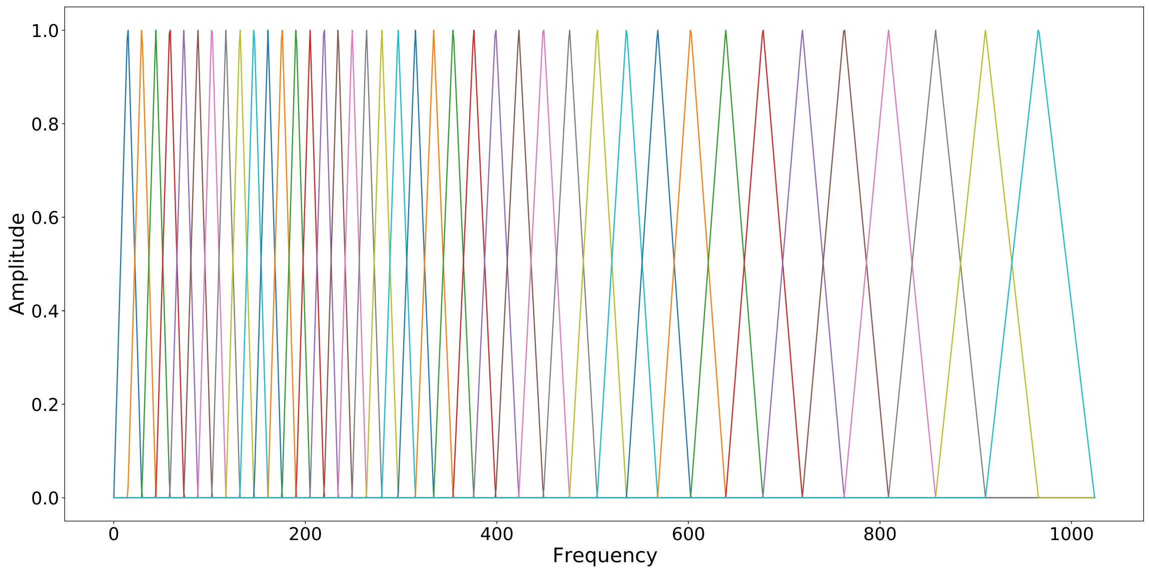


Figure 3.3 – Example of a mel filter bank.

particularly relevant for modeling properties of a sequence with temporal dependency like feature matrix. Still, vanilla *Recurrent Neural Network* (RNN)s suffer from 'vanishing gradient' issues, making long-term dependency modeling problematic. The *Long Short-Term Memory* (LSTM) network is a notable recurrent model architecture that can cope with this problem. It is made up of an input gate, a forget gate, and an output gate regulating the transmission of information passing through the input vector, the cell's hidden state and the output vector. These gates aid the network in coping with lengthy sequences and resolving vanishing problems. Exhaustive description of the network is given in [HS97].

A phrase can be thought of as a non-causal sequence where an uttered word is emitted in function to previous and subsequent words in the sentence. At a lower level, one considered text unit also depends on text units of past and future time frames. Because recurrent layers only analyze one frame given the previous ones, they do not consider data in subsequent frames. Wrapping two LSTM layers into a layer with shared inputs and concatenated outputs is an effective technique of employing LSTM layers on non-causal sequences. One LSTM layer analyzes sequences ahead, while the other analyzes sequences backward, resulting in the use of future and past information to compute the final output vector. This layer is known as *Bidirectional Long Short-Term Memory* (BiLSTM) and is extensively presented in [SP97].

3.4 Intermediate representation

3.4.1 Discussing character or phoneme

Multiple text units can be considered as outputs of the acoustic model. The most classic are characters [SDE19], *e.g.* of the Latin alphabet, and phonemes [GYL20]. The first representation alleviates the need of a pronunciation lexicon. As described in Section 2.1, this representation is used in *End-To-End* (E2E) acoustic models. Such models take as input low-level features and directly output characters. Recognition systems based on this type of acoustic model, are however known to be more data-demanding for training in comparison to a more classic phoneme-based architecture [GJ14].

Moreover, the character representation is badly fitted for multilingual systems. Indeed, character-based SVR systems cannot be employed to transcribe songs in languages that employ a different writing system than the one used by the system. Phoneme-based methods, by comparison, may transcribe any language as long as a pronunciation lexicon is provided for it. This subject is extensively studied in Section 6.1.1.

The second representation, the phonemes, can be defined as the smallest distinguishable unit that can be isolated in a speech sequence. One of the most popular phoneme sets is the *International Phonetic Alphabet* (IPA) which is a standardized representation of speech sounds in written form of all spoken languages. It is described extensively in the next section.

3.4.2 IPA Phonemes

General presentation:

The IPA is a phonetic alphabet composed of more than 160 symbols to transcribe speech or singing. 107 of these symbols are phonemes. These phonemes are characterized by distinctive features like: 'Voicing', 'Tenseness' and 'Frontness'. Features of a phoneme describe articulatory and acoustic properties of the realization of this phoneme. Such features are used to group phonemes into natural classes. Acoustic realizations of phonemes of the same classes are more likely to have comparable behaviors as they have common features. The two largest natural classes are consonants and vowels.

They are principally distinguished by a feature called 'Sonority', linked to the amount of acoustic energy of the realization of the phoneme. Vowel sounds possess a lot of acoustic energy: they are said to be sonorous. It can be explained by the fact that vowels are produced with an open vocal tract and always vibrating vocal folds. Furthermore, vowels are often maintained for a lengthy period of time. This is especially true for the singing voice, where the duration of vowels generally corresponds to the duration of notes [Mes13]. In opposition, consonant sounds are shorter and with a partially or fully closed vocal tract. The sounds of consonants are then less sonorous. Vocal chords can vibrate or not during the acoustic realization of a consonant.

IPA vowels:

All vowels can be described simply using few features. These vowels are commonly represented in the IPA vowel chart displayed in Figure 3.4. Similarity between vowels can be easily computed using the distance between vowels in this trapezium. Vertical and Horizontal positions are respectively linked to the 'Height' and the 'Backness' of the vowel. Such features are theoretically describing the location of the tongue during the phonation: from high to low for 'Height' and from front to back for 'Backness' where high is at the top

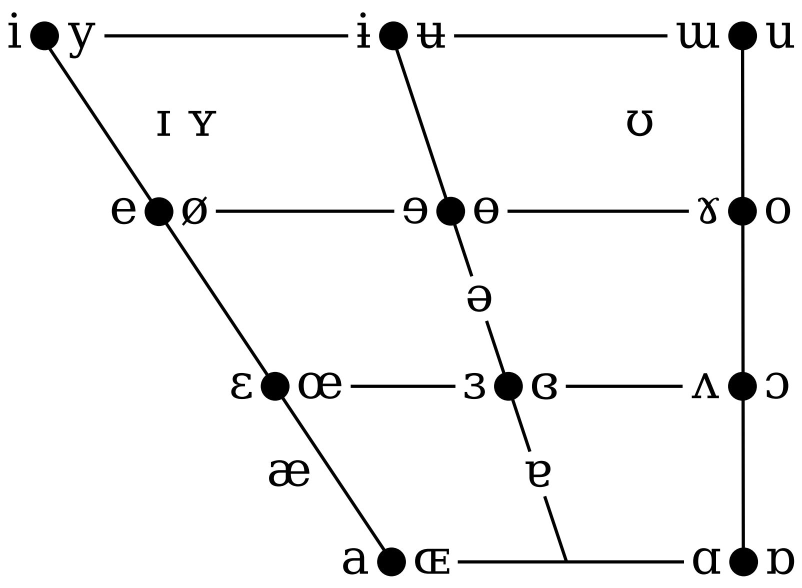


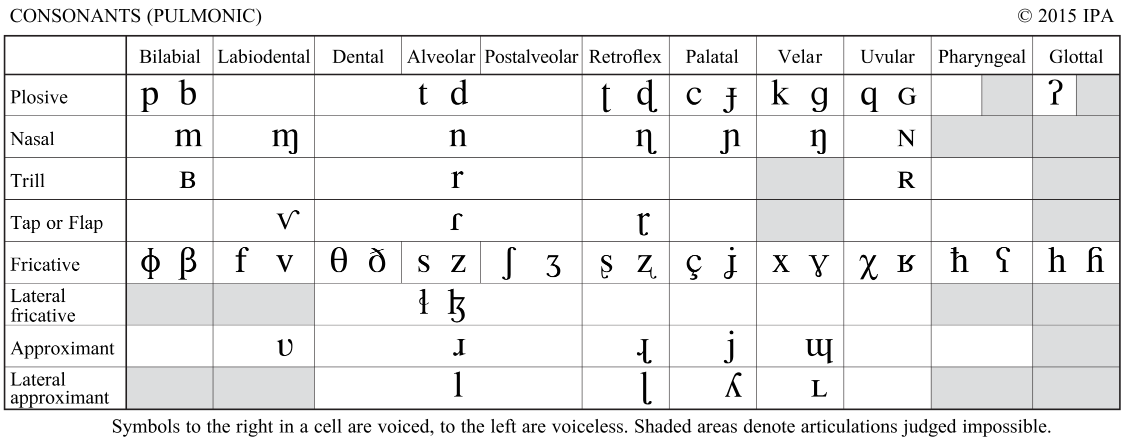
Figure 3.4 – IPA vowel chart reproduced from [vow]. Nasal vowels are not included in the figure. CC BY-SA 4.0, via Wikimedia Commons. Permission granted for reuse.

of the diagram and front at its left. In practice, each position is linked to one of the two first formants. Two more features are generally used to describe vowels: 'Tenseness' and 'Rounding'. In the diagram, vowels are grouped in pairs of unrounded and rounded vowel sounds where lips are respectively unrounded and rounded during phonation. 'Tenseness' is linked to the amount of muscular effort needed for the phonation of one particular phoneme.

IPA consonants:

Consonants can be discriminated with the following features: 'Voicing', 'Manner of articulation' and 'Place of articulation'. This type of feature can be used to further decompose consonants into various natural classes like plosive or fricative. A phoneme is called voiced if the vocal cords are vibrating during its acoustic realization and voiceless in the opposite case. The 'Place of articulation' characterizes where the vocal tract is the most obstructed during the acoustic realization of the consonant. The 'Manner of articulation' describes the way it is obstructed. Most consonants are described in the IPA pulmonic consonant table displayed in Figure 3.5.

Rows design the 'Manner of articulation' and columns the 'Place of articulation'. 'Place of articulation' begins on the left with the lips and progresses gradually to the glottis on the right. 'Manner of articulation' begins at the top with the greatest obstruction of the vocal tract (*i.e.* plosives) and progresses gradually to the bottom with the least degree of obstruction (*i.e.* approximants). Consonants can be alone in the table or arranged in pairs of voiced and voiceless phonemes. A more complete overview of characteristics of IPA phonemes is given in [And18].


CONSONANTS (PULMONIC) © 2015 IPA

	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		ʈ ɖ	c ɟ	k ɡ	q ɢ	▨	ʔ
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	ɴ	▨	▨
Trill	ʙ			r				▨	ʀ		▨
Tap or Flap		ⱱ		ɾ		ɽ		▨			▨
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative	▨	▨		ɬ ɮ						▨	▨
Approximant		ʋ		ɹ		ɻ	j	ɰ			▨
Lateral approximant	▨	▨		l		ɭ	ʎ	ʟ		▨	▨

Symbols to the right in a cell are voiced, to the left are voiceless. Shaded areas denote articulations judged impossible.

Figure 3.5 – Table of IPA pulmonic consonants [con]. CC BY-SA 3.0, via Wikimedia Commons. Permission granted for reuse.

3.5 Connectionist temporal classification

Acoustic models are usually trained using alignment annotations, at the frame level, between audio and text. However, the availability of such annotations is very limited for polyphonic music where they are traditionally generated by employing an intermediate model [Kru18], leading to suboptimal performance [SDE19].

One way to go around this problem is to use the *Connectionist Temporal Classification* (CTC) algorithm [GFGS06] that will be presented succinctly in the next paragraphs. As explained in Section 2.1, this technique is particularly used to train E2E acoustic models [SDE19], removing the requirement for character annotations at the frame level, which neither exist nor make sense in practice. A CTC system can then be trained directly using a dataset $\{(\mathbf{X}_i, \boldsymbol{u}_i)_{i=1}^{N_{data}}\}$ where $\mathbf{X}_i$ is a matrix of acoustic features, $\boldsymbol{u}_i$ the corresponding sequence of characters and N_{data} the number of examples in the dataset.

General presentation:

The CTC algorithm enables to train a model without any predefined alignment, by introducing a blank token ϵ in the set $\mathcal{C}$ of units supported by the model. Any unit, including ϵ, can be emitted at each frame by the model. Generally, this model is a RNN, but any model capable of giving a distribution over outputs for a fixed context of the input $\mathbf{X}$ could be used. The probability of the output unit sequence is maximized using the algorithm by marginalizing, over all possible alignments, for a given input $\mathbf{X}$. The objective function being differentiable, the network is trained using back-propagation through time, by providing the phonetic transcription $\boldsymbol{y}$ of the segment of lyrics. For a considered input $\mathbf{X}$, an output distribution through all possible $\boldsymbol{y}$ is finally obtained.

A comparison between outputs of a model trained with frame-level annotations and a CTC trained model are displayed in Figure 3.6. It shows that the CTC model, by contrast to the framewise one, did not learn boundaries and timings. It can be explained as the CTC algorithm considers two label sequences equivalent if they only differ in alignment.

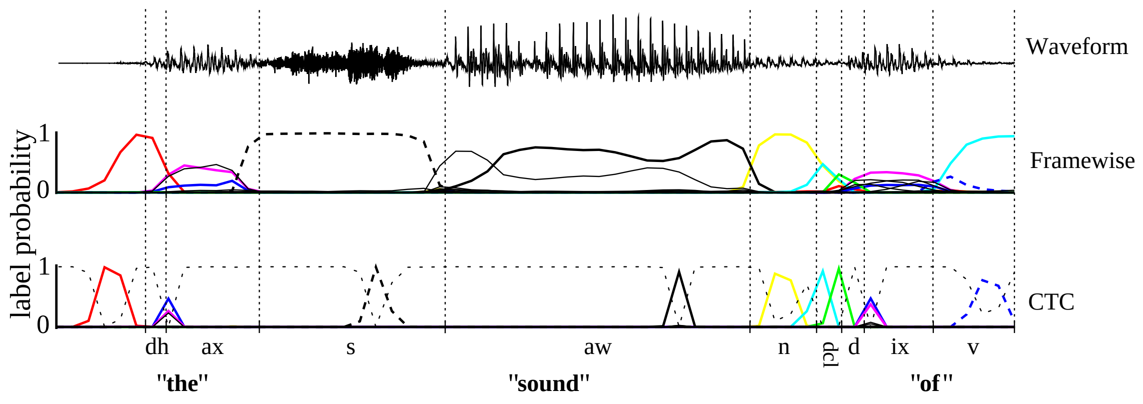


Figure 3.6 – Framewise and CTC models predicting a speech signal reproduced from [GFGS06]. The shaded lines correspond to the probabilities of observing phonemes at given times. Transparent dot lines correspond to the probability of the blank label. Copyright © 2006, ACM. Permission granted for reuse.

The CTC model prediction can be described as a sequence of predicted spike units (*e.g.* character or phoneme) separated by 'blank' units. Still, CTC models are shown to infer reliable alignments when outputs of each frame depend on the entire input sequence using e.g. BiLSTM [VHM+20]. In contrast, the uni-directional CTC acoustic model suffers from alignment delay [SdCQS+15].

CTC alignments:

When mapping $\mathbf{X} = (\mathrm{X}_1, \mathrm{X}_2, \mathrm{X}_3, \mathrm{X}_4, \mathrm{X}_5, \mathrm{X}_6)$, to $\boldsymbol{y} = [a, b, c]$, multiple alignments are possible: *aaabbc*, *aabbcc* ... However, these sorts of alignments raise two concerns: it is not always desirable to map every input frame to an output symbol, and it is necessary to be able to have two identical symbols adjacent to one another. These problems are resolved using an additional symbol: the blank symbol ϵ. The ϵ symbol is considered as a 'non-emission' symbol and is simply discarded when an alignment is mapped to its corresponding transcript. Alignments allowed by the algorithm for a given transcript $\boldsymbol{y}$ must fulfill two requirements. Firstly, they must have the same length as the input. Secondly, they must map to the given transcript after merging the repeated symbols and discarding the ϵ.

The CTC alignments have multiple properties. They are monotonic and many-to-one (many inputs can map to the same output however reciprocity is not true). In the case of lyrics transcription, both assumptions can be assumed to be valid. Indeed, the relationship between frames of singing and text units is always monotonic in time. Furthermore, the duration of a frame is typically significantly less than that of a text unit (*e.g.* phoneme or character), this last usually running through numerous frames. Inversely, it seems unlikely that a given frame is overlapping multiple text units. The second assumption also implies that the length of $\boldsymbol{y}$ cannot be longer than the length of $\mathbf{X}$. Such a constraint is realistic in the case of SVR, the size of the input being generally much longer than the output one.

Loss function:

A RNN is classically used to produce per time step probabilities for the CTC algorithm. Its output is constituted of all text units considered (*e.g.* characters or phonemes) and the additional ϵ label. Considering $\mathbf{X}$, an input sequence of length T, the obtained output vectors Y_t are normalized by a softmax function as:

$$P(s,t|\mathbf{X}) = \frac{e^{y_{s,t}}}{\sum_{s'} e^{y_{s',t}}} \tag{3.6}$$

for $1 \leq s \leq |\mathcal{C}|$, $1 \leq t \leq T$. $\mathcal{C}$ and T are respectively the set of units outputted by the model, including blank, and the number of temporal frames. Here, $y_{s,t}$ is the s^{th} element of the vector Y_t. The per-time step probability of each unit of the vocabulary $\mathcal{C}$ given the input $\mathbf{X}$ is then obtained. The probability of an alignment $\boldsymbol{a}$, given input $\mathbf{X}$, can then be easily computed as the the product of the emission probabilities at every time step such as:

$$P(\boldsymbol{a} \mid \mathbf{X}) = \prod_{t=1}^{T} P(a_t, t \mid \mathbf{X}) \tag{3.7}$$

The probability of a given transcription $\boldsymbol{y}$ can finally be obtained by marginalizing all other possible alignments mapping to it with:

$$P(\boldsymbol{y} \mid \mathbf{X}) = \sum_{\boldsymbol{a} \in \beta^{-1}(\boldsymbol{y})} P(\boldsymbol{a} \mid \mathbf{X}) \tag{3.8}$$

here, β is the function that, for a given alignment, merges all repeated characters and discards the ϵ units. A distribution of all possible $\boldsymbol{y}$ for a given input $\mathbf{X}$ is finally obtained. The CTC cost function, or CTC loss function, is usually defined as the negative log likelihood of this probability. The network is therefore optimized to minimize this cost. This loss function being composed of sums and products of the per time step probabilities, it is differentiable with respect to them. Knowing this, the gradient of this loss function can be easily computed with respect to these probabilities and used to perform backpropagation.

CTC forward-backward algorithm:

The CTC loss being computationally expensive, a forward backward algorithm is generally used to compute it. This algorithm is a dynamic programming algorithm allowing a much faster computation of the loss. It avoids the need of computing the probability of each possible alignment at each time step, which would quickly become untractable. This algorithm is similar to the one used to compute the probability of a given sequence in the context of a *Hidden Markov Models* (HMM) with some specificity due to the CTC algorithm properties. The basic idea is to merge two alignments that arrive at the same output at the same step.

An example of the computation performed by the dynamic programming algorithm is displayed in Figure 3.7. Here, consider $\boldsymbol{z}$, which is the transcript $\boldsymbol{y}$ extended by an ϵ at the beginning, the end, and in between every unit. A decoding diagram of size $|\boldsymbol{z}| \times T$ is then created. Each node $\alpha_{s,t}$ in the diagram represents the value of the CTC score of

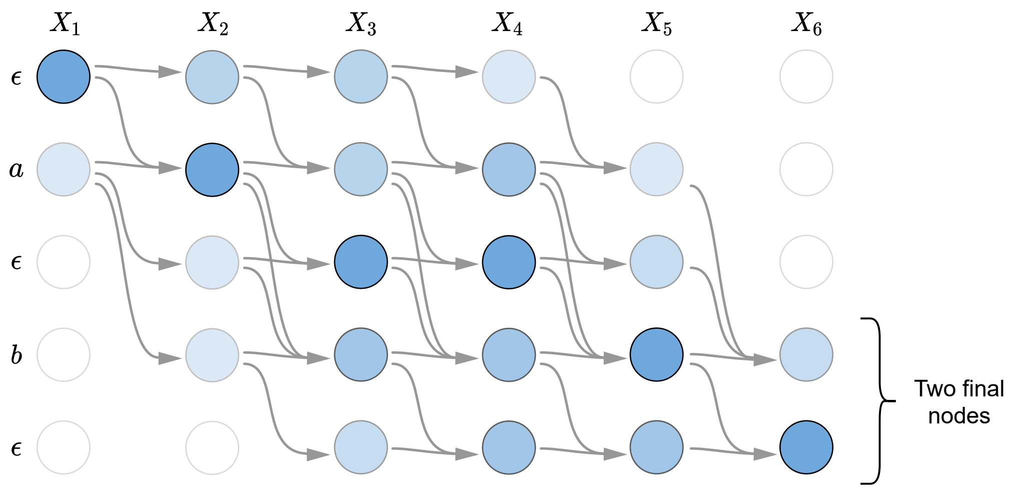


Figure 3.7 – Computing the CTC score $P(\boldsymbol{y}|\mathbf{X})$ using dynamic programming, node (s,t) in the diagram represents $\alpha_{s,t}$. The score is computed by summing the probabilities of the two final nodes. Here, an example is given for $\mathbf{X} = (\mathrm{X}_1, ..., \mathrm{X}_6)$ and $\boldsymbol{y} = [a, b]$. This figure is adapted from [Han17]. It is licensed under Creative Commons Attribution 4.0 Int. Permission is granted for adaptation

the sub-sequence $\boldsymbol{z}_{1:s}$ after t frames. Every valid alignment has a path in this graph. The $\alpha_{s,t}$ scores are afterwards computed efficiently by merging together alignments reaching this node. $\alpha_{s,t}$ is subsequently computed recursively from α's of the previous frame. Only transitions between blank and non-blank characters, and between pairs of distinct non-blank characters are allowed. Since ϵ at the beginning and the end of the sequence is optional, there are two valid starting nodes and two final nodes. The coefficients α's are initialized as follows:

$$\alpha_{s,1} = P(z_s, 1 \mid \mathbf{X}) \text{ for } s \in \{1, 2\} \text{ and } \alpha_{s,1} = 0, \forall s > 2 \tag{3.9}$$

Recursion is given by:

$$\begin{aligned} \alpha_{s,t} &= (\sum_{\tau=0}^{\tau=1} \alpha_{s-\tau,t-1}) P(z_s, t \mid \mathbf{X}), \text{if } z_s \in \{\epsilon, z_{s-2}\} \\ \alpha_{s,t} &= (\sum_{\tau=0}^{\tau=2} \alpha_{s-\tau,t-1}) P(z_s, t \mid \mathbf{X}), \quad \text{otherwise} \end{aligned} \tag{3.10}$$

Finally, the CTC score is given at the last step T by summing the two final nodes and applying negative log with:

$$P(\boldsymbol{y}|\mathbf{X}) = -\log(\alpha_{|\mathbf{z}|,T} + \alpha_{|\mathbf{z}|-1,T}) \tag{3.11}$$

Numerical stability:

As the naive CTC scoring is numerically unstable, multiple solutions are used to avoid underflow. The α's coefficients can be normalized at each time step such as:

$$c_t = \sum_s \alpha_{s,t}, \quad \hat{\alpha}_{s,t} = \frac{\alpha_{s,t}}{c_t} \tag{3.12}$$

with c_t the normalization coefficient of the t^{th} time step. However, underflow can still happen for long input. Preferably, computations are done in log-space using the log-sum-exp trick [Han17]. Inference is equally performed in log-space using the log-sum-exp trick.

CTC limitations:

An important property of the CTC algorithm is the conditional independence assumption. The algorithm considers that given the inputs, each output is independent of each other. The CTC algorithm thus cannot learn a language model over the outputs. An additional language model is then generally included to improve the transcription performance. This property, however, makes acoustic models trained using the CTC algorithm straightforward to transfer from one domain to another (*e.g.* new genres or new languages), with only the language model needing to be changed.

3.6 Language modeling

A language model $\mathscr{L}$ is traditionally described by a vocabulary, defining a collection of text units the model can identify, and a probability distribution over all potential unit sequences. In this section, we consider the text units to represent words, a popular choice for SVR systems [GYL20]. Still, language models can also be specified at the level of subword (*e.g.* phoneme or character). For any sequence of words $\boldsymbol{w}$, a language model computes the corresponding probability $P(\boldsymbol{w})$. Considering a sequence of N words, the probability can be written:

$$P(\boldsymbol{w}_1, ..., \boldsymbol{w}_N) = P(\boldsymbol{w}_1)P(\boldsymbol{w}_2|\boldsymbol{w}_1)...P(\boldsymbol{w}_N|\boldsymbol{w}_1, ..., \boldsymbol{w}_{N-1}) \tag{3.13}$$

where $\boldsymbol{w}_i$ is the i^{th} word of the sequence. Each conditional term could be estimated from a training dataset, using relative frequency counts with:

$$P(\boldsymbol{w}_N|\boldsymbol{w}_1, ..., \boldsymbol{w}_{N-1}) = \frac{C(\boldsymbol{w}_1, ..., \boldsymbol{w}_N)}{C(\boldsymbol{w}_1, ..., \boldsymbol{w}_{N-1})} \tag{3.14}$$

where C is the function counting the frequency of each sentence in the considered dataset. However, this type of method becomes infeasible when considering large strings, with most of the sentences having a count of zero in the training dataset. To simplify this problem, for a given word, the n-gram model only considers its $n-1$ preceding words are sufficient to specify the probability of a new word. It then makes the following assumption

$$P(\boldsymbol{w}_N|\boldsymbol{w}_1, ..., \boldsymbol{w}_{N-1}) \approx P(\boldsymbol{w}_N|\boldsymbol{w}_{N-n+1}, ..., \boldsymbol{w}_{N-1}) \tag{3.15}$$

thus, the probability of a sequence $\boldsymbol{w}$ is computed by:

$$P(\boldsymbol{w}_1, ..., \boldsymbol{w}_N) = P(\boldsymbol{w}_1)P(\boldsymbol{w}_2|\boldsymbol{w}_1)...P(\boldsymbol{w}_N|\boldsymbol{w}_{N-n+1}, ..., \boldsymbol{w}_{N-1}) \tag{3.16}$$

with each conditional term being given by:

$$P(\boldsymbol{w}_N|\boldsymbol{w}_{N-n+1}, ..., \boldsymbol{w}_{N-1}) = \frac{C(\boldsymbol{w}_{N-n+1}, ..., \boldsymbol{w}_N)}{C(\boldsymbol{w}_{N-n+1}, ..., \boldsymbol{w}_{N-1})} \tag{3.17}$$

A n-gram model is therefore a statistical model predicting the probability of a word given its $n-1$ previous words. More precisely, it is capable of computing, for each possible sequence of n words of the vocabulary, the probability of the last word given the previous ones. The probability of a sequence of words is finally directly given by Equation (3.16). A n-gram is simply defined as a sequence of n words. The language models are trained on text datasets by considering all n-grams present. However, even for large text databases, some n-grams are usually missing and assigned zero probability while having non-zero probability in real-world circumstances. Other n-gram occurrences are too low to provide a reliable estimate of their probability. Several probabilities of n-gram are then underestimated which can degrade the model performance.

To mitigate this problem, several methods are classically employed like smoothing. This method includes updating the likelihood of a n-gram by using probabilities of lower order n-grams (whose chance of occurrence in a text corpus is higher). By changing the way the probability mass is allocated, models generalizing better are obtained. One way to adjust probability is the so-called discount. The main idea is to reduce the probability of the seen sequences of a given amount, that are then dispersed among the probabilities of unobserved n-grams. When no instance of a n-gram is found, the algorithm thus resorts to the probabilities of n-grams of lower order.

A n-gram of size one is referred to as unigram, size two as bigram and size three as trigram, for higher order they are referred as n-gram. Bigram or trigram models are the most commonly used. In any case, the probability of n-grams of inferior rank are also computed to be able to compute Equation (3.16). Considering n-grams of higher rank provides better statistical models but are more resource demanding. Moreover, the number of n-grams to compute grows exponentially with n, most of them being assigned a probability of zero.

3.7 Decoding

This section is mainly taken from the work of Hannun [Han17]. We consider the specific case where the acoustic model is trained with the CTC algorithm. As a reminder, the decoder's objective is to compute the most likely transcript $\hat{\boldsymbol{y}}$ given the input matrix $\mathbf{X}$, such as:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} P(\boldsymbol{y}|\mathbf{X}) \tag{3.18}$$

3.7.1 Greasy decoding

An easy way to perform decoding, for a SVR approach, is the greasy decoding [GJ14]. It is at first a correct approximation to obtain a probable output for a considered input. It solely consists in taking, at each time step, the output of the CTC trained acoustic model with the highest probability and returning the corresponding sequence of symbols $\hat{\boldsymbol{a}}$. More formally:

$$\hat{\boldsymbol{a}} = \arg\max_{\boldsymbol{a}} \prod_t P(a_t|\mathbf{X}) \tag{3.19}$$

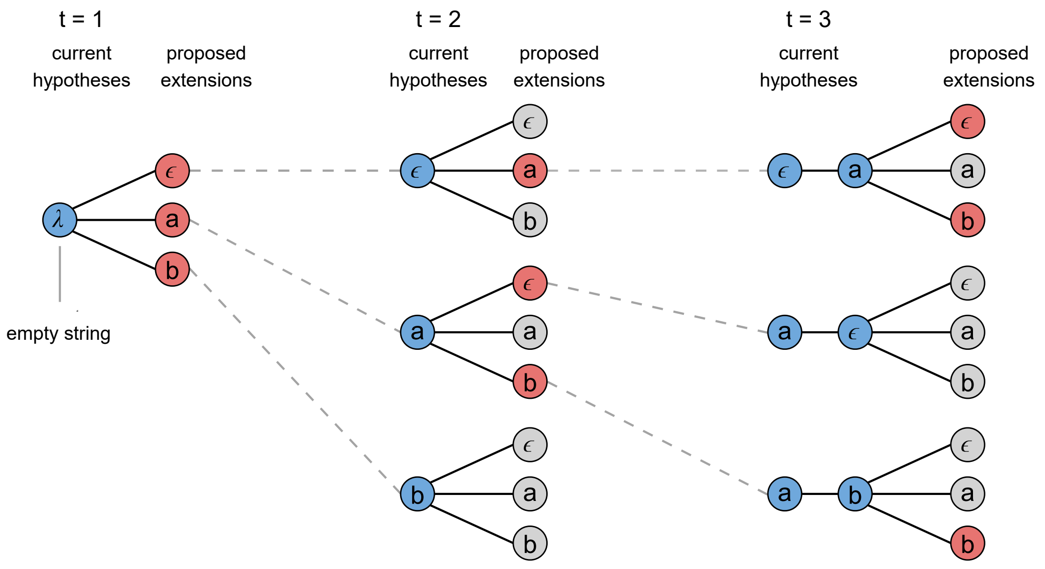


Figure 3.8 – A vanilla beam search algorithm with an alphabet of $\{\epsilon, a, b\}$ and a beam size of three. At each time step t, the set of possible alignments is extended with all possible units of the alphabet and the top three alignments are kept. This figure is reproduced from [Han17]. It is licensed under Creative Commons Attribution 4.0 Int. Permission is granted for reuse.

An estimation of the transcription $\hat{\boldsymbol{y}}$ is then simply obtained after merging repeated symbols and discarding CTC labels. This method has two drawbacks. Firstly, it does not consider the use of an additional language model. Secondly, it does not take into account that, in the CTC paradigm, multiple alignments can map to the same output. So, such a basic decoding can perform well when a single alignment possesses most of the probability mass, but can miss the correct output otherwise. As an example, consider two alignments that map to the same valid output, each with a lower probability than a third alignment mapping to an incorrect outcome but obtaining a greater value when summing both probabilities. Using the greasy decoding, the third output is returned, obtaining an incorrect transcription.

3.7.2 Prefix beam search

One decoding taking into account the many-to-one property of the CTC algorithm is the CTC prefix beam search. This approach is an extension of the vanilla beam search. A vanilla beam search is displayed in Figure 3.8. Starting from an empty string, a set of probable alignments is proposed, at each time step, by extending preceding hypotheses with all possible units of the considered alphabet. Following that, only the most likely alignments are preserved, depending on a tunable parameter N delimiting the size of the hypothesis pool, the so-called beam size. Thus, the algorithm outputs, at the last time frame, the N top alignments.

In the case of the CTC prefix beam search, to handle the many-to-one property of CTC alignments, instead of outputting the best N alignments, the N best prefixes are returned. Starting with an empty string, a set of possible prefixes is outputted, for each time frame,

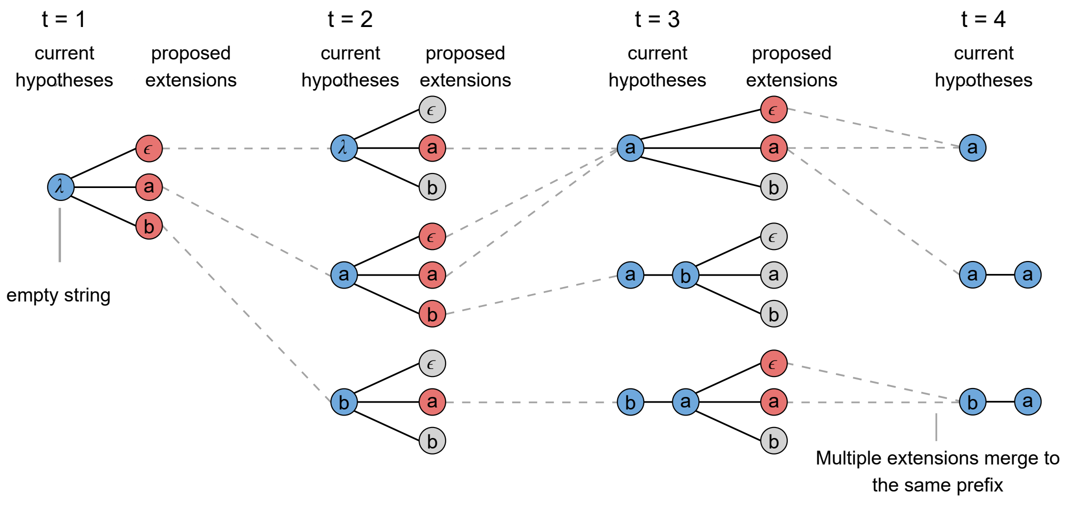


Figure 3.9 – A prefix CTC beam search algorithm with an alphabet of $\{\epsilon, a, b\}$ and a beam size of three. In this case, the top three prefixes are kept at each time step t. The probability of one prefix is obtained by summing probabilities of all alignments mapping to it. This figure is reproduced from [Han17]. It is licensed under Creative Commons Attribution 4.0 Int. Permission is granted for reuse.

by extending previous prefixes with all potential units of the considered alphabet. The score of each prefix is then computed by summing scores of each alignment mapping to it, *i.e.* after merging and discarding characters. Again, only the top N prefixes are preserved.

The top one prefix is usually chosen as the estimated transcription $\hat{\boldsymbol{y}}$, but the pool of answers can be also re-scored using another function before making the final decision. The prefix algorithm is displayed in Figure 3.9. It has been successfully applied to SVR systems [SDE19]. When using few computational resources, *i.e.* a small beam size, the method may not always discover the most likely transcription. However, when given greater processing resources (*i.e.* by increasing the beam size), this approach is known to converge asymptotically to it.

Looking in Figure 3.9, at the third temporal frame, there is a case where a proposed extension can generate two hypothetical prefixes. Such cases happen when the new proposed unit for a prefix is a repetition of the last unit of the prefix. The last unit of the first hypothetical prefix is unrepeated and its score is obtained from the probabilities of all alignments mapping the previous prefix not ending with an ϵ. Inversely, the last unit of the second hypothetical prefix is repeated and is obtained from the probabilities of all alignments mapping to the previous prefix ending with an ϵ, as it is required between repeated characters.

To compute the probability of both new prefixes, with or without a repeated last unit, knowing which alignments of the previous prefix concerned end with an ϵ or not is needed. The probability is then preserved, for each prefix in the beam, of all alignments ending with an ϵ, and the probability of all other alignments. It permits, if a split occurs during a proposed extension, to preserve the required portion of the prior considered prefix's score for both new prefixes. Their combined score is used to rank the hypothetical prefixes at each step before trimming the beam. The way this case is handled is displayed in Figure 3.10.

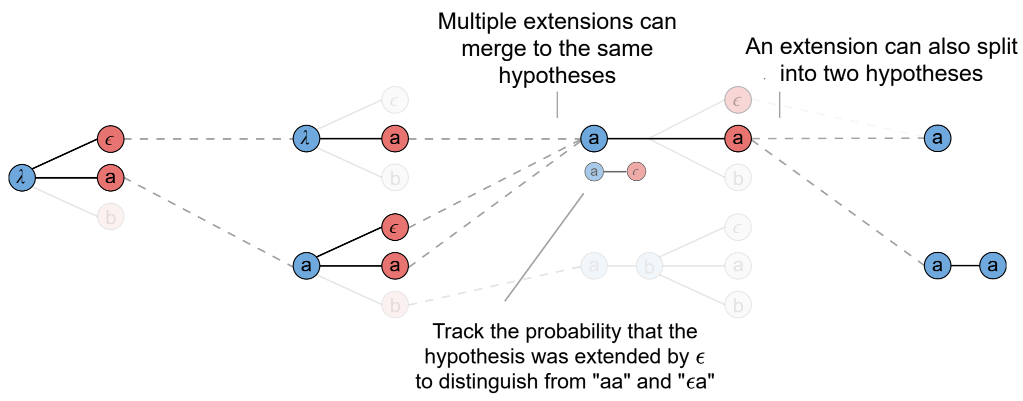

Multiple extensions can merge to the same hypotheses
An extension can also split into two hypotheses
Track the probability that the hypothesis was extended by ϵ to distinguish from "aa" and "ϵa"
λ
ϵ
a
b

Figure 3.10 – Two probabilities are kept for each hypothetical prefix: the sum of probabilities of all alignments mapping to it ending with an ϵ and the sum of probabilities of others mapping alignments. It allows to compute the probability of new prefixes when a proposed extension splits into two hypotheses: with or without a repeated character, an ϵ being required between the latter. This figure is reproduced from [Han17]. It is licensed under Creative Commons Attribution 4.0 Int. Permission is granted for reuse.

The complete CTC prefix beam algorithm is extensively described in [AAA+16]. Still, the presented algorithm does not yet incorporate any language modeling.

3.7.3 Incorporating language modeling

A language model can be directly incorporated during the inference, obtaining as a new decoding equation:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} P(\boldsymbol{y}|\mathbf{X})P(\boldsymbol{y}) \tag{3.20}$$

The probability of the transcription given the input acoustic feature matrix $P(\boldsymbol{y}|\mathbf{X})$ is given by the acoustic model and the probability of the transcription $P(\boldsymbol{y})$ is given by the language model. In the case of the prefix algorithm, the latter is only taken into account when a prefix is extended by the type of text unit considered by the language model, *i.e.* a character or a word. As a consequence, the algorithm tends to favor shorter prefixes. To mitigate this effect, and balance between both terms, two tunable parameters are also inserted, such as:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} P(\boldsymbol{y}|\mathbf{X})P(\boldsymbol{y})^a L(\boldsymbol{y})^b \tag{3.21}$$

where a is the grammar scale factor, b the insertion penalty parameter. $L(\boldsymbol{y})$ is a function outputting the length of the string, given the considered text unit of the language model. It is acting as an insertion bonus factor to avoid the decoder to always favor shorter transcripts. The grammar scale parameter simply allows to adjust acoustic and linguistic information balance during decoding. Both parameters are usually chosen on a validation dataset as in [SDE19].

3.8 Evaluation

Results are classically analysed using the *Character Error rate* (CER) and *Word Error rate* (WER) metrics. They are both based on the Levenshtein distance. This distance is used to quantify the similarity between two strings of characters. It is equal to the minimal number of character operations (*i.e.* insertion, deletion and substitution) needed to map one string to the other. The CER is simply the Levenshtein distance between a generated and an expected character sequence, where all character operations are weighted equally, normalized by the length of the expected sequence. Here, the Levenshtein distance is the smallest number of operations required to transfer the produced string to the reference one. The WER is derived from this definition, working at the word level instead of the character level. It can be computed as:

$$WER = \frac{(S + D + I)}{N} = \frac{(S + D + I)}{(S + D + C)} \tag{3.22}$$

where S is the number of word substitutions, D is the number of word deletions, I is the number of word insertions, C is the number of correct words and N is the number of words in the reference. The same equation can be directly deduced for the CER. Both metrics are better when smaller. Standard errors are given whenever possible using the bootstrapped toolkit [1].

1. Available at `https://github.com/facebookarchive/bootstrapped`

Chapter 4

A vanilla lyrics transcription system

In this chapter, we present the first lyrics transcription implementation that we developed. This approach is heavily inspired by Stoller19 [SDE19] which was the state-of-the-art system in *Singing Voice Recognition* (SVR) at the time. The majority of the SVR systems developed in the subsequent chapters are derived from this iteration. It will therefore serve as a reference system to describe these implementations. We will emphasize their differences with this reference system, each design choice being grounded on the task's specificity. An overview of the approach is given in Figure 4.1.

It is composed of a series of modules sequentially applied, the most significant of which are the acoustic model $\mathscr{H}$ and the language model $\mathscr{L}$. The system takes as audio input a monophonic audio signal $\mathrm{X} \in \mathbb{R}^N$, where N is the number of samples and outputs the estimated transcript $\hat{\boldsymbol{y}}$. An isolated singing voice $\mathrm{X}_{voc} \in \mathbb{R}^N$ is first extracted from the audio input. An acoustic feature matrix $\mathbf{X} \in \mathbb{R}^{F \times T}$ is subsequently computed from it, where F is the feature size and T is the number of frames. The acoustic model $\mathscr{H}$ is an audio-to-character module taking as input the acoustic feature matrix and outputting a posteriorgram $\mathbf{R}$. More formally, given $\mathbf{X}$, the matrix of acoustic features, each coefficient $r_{s,t}$ provides an estimation of the posterior probability of c_s, the s^{th} character being uttered at the t^{th} frame, such as:

$$r_{s,t} = \mathscr{H}_{s,t}(\mathrm{X}) = P(c_s, t \mid \mathbf{X}), \tag{4.1}$$

for $1 \leq s \leq |\mathcal{C}|$, $1 \leq t \leq T'$, where $\mathcal{C}$ is the set of units outputted by the model and T' is the number of frames obtained after the model's downsampling. It consists of 26 lowercase letters of the Latin alphabet, and the word-boundary 'space' token, the instrumental token 'I', described in Section 4.2, the apostrophe and the *Connectionist Temporal Classification* (CTC) blank token ϵ introduced in Section 3.5. The decoder finally takes as input the character posteriorgram $\mathbf{R}$ and the language model $\mathscr{L}$ to output the estimated transcript $\hat{\boldsymbol{y}}$.

The chapter is divided into four sections. First of all, the developed lyrics transcription implementation is presented in Section 4.1. After that, the dataset processing is described in Section 4.2. Then, the system's parameters are detailed in Section 4.3. Finally, the evaluation of the designed system is performed in Section 4.4. Its performances are compared to those of state-of-the-art architectures for the lyrics transcription task.

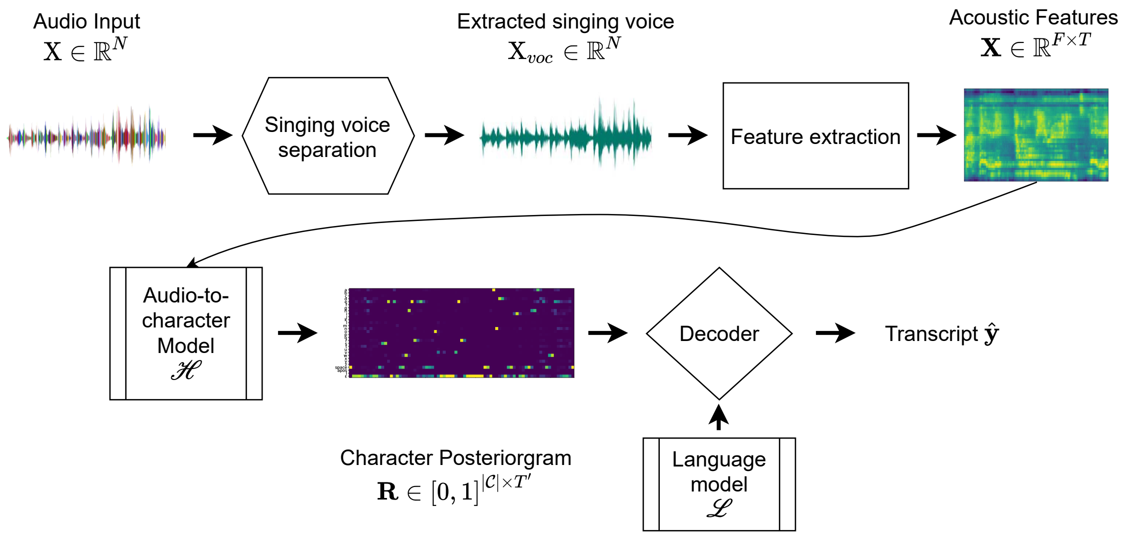


Figure 4.1 – General overview of our first lyrics transcription system.

4.1 General presentation

The acoustic model $\mathscr{H}$ is an *End-To-End* (E2E) acoustic model. It takes as input low level features and directly outputs the character probability through time. Such a model alleviates the need of a pronunciation lexicon as described in Section 3.4.1. As discussed in Section 2.3, a preprocessing of *Singing Voice Separation* (SVS) is likely to improve the performance of this type of acoustic model. Such preprocessing is then performed to the input audio before calculating features from the extracted singing voice.

As input features, we choose to use mel-spectrograms over spectrograms for the reasons described in Section 3.2. The use of the direct waveforms could be an alternative, however, systems trained on waveforms are known to need way more data than those trained on mel-spectrograms to obtain similar results [PNP$^+$17]. Moreover, in [DS14], the authors show that the first convolutional layer of their trained network on this data learns something very similar to sinusoids thus having its output comparable to a Fourier transform.

This audio-to-character module is implemented using a *Convolutional Recurrent Neural Network* (CRNN) trained with a CTC algorithm, alleviating the need of frame level synchronized annotations. As described in Section 2.1 and Section 2.2, the CTC algorithm has been applied successfully to both the *Automatic Speech Recognition* (ASR) and SVR domains. To reduce the feature dimensionality and accelerate training, we use additional convolutional layers. We use bidirectional *Long Short-Term Memory* (LSTM)s as *Recurrent Neural Network* (RNN) layers, so outputs at each frame depend on the entire input sequence as in [GJ14]. For the language model $\mathscr{L}$, we use a simple word n-gram model as is often done in state-of-the-art approaches seen in Chapter 2. $\mathscr{H}$ and $\mathscr{L}$ are trained independently.

4.2 Dataset Processing

The audio-to-character model $\mathscr{H}$ is trained with the English part of the DALI dataset. Tracks are downsampled to 16 kHz and converted to mono. Vocals from each song are then extracted using Spleeter [HKVM20]. For each song, training samples are generated by segmenting the track using a window of five seconds with a hop size of 2.5 seconds. The character sequence associated with a segment is created by concatenating all words whose start position are within the segment. In case no words start within a segment, we generate a token 'I' (for 'Instrumental'). Character sequences are transformed to fit the set of characters $\mathcal{C}$ outputted by the model: each character sequence is converted to lower-case and non-valid characters are discarded. We split the dataset into a training set consisting of 85% of the data, a validation set containing 1% of the data, and a test set containing 14% of the data. This split is done in an artist-aware fashion [Fle07]. We obtain datasets of 384, 5 and 63 hours of music.

The lyrics text corpus used to train the language model is simply obtained using the lyrics' text of the training set previously computed. Various basic text preprocessing steps are then applied, like tokenization, unicode normalization and removing non-ASCII characters, to obtain the final lyrics text corpus. It is composed of 22000 words with only 6000 words with more than five occurrences. These characteristics are comparable to those of Mesaros's lyrics corpus described in [Mes12].

4.3 Model parameters

The model $\mathscr{H}$ is composed of two convolutional layers, followed by three *Bidirectional Long Short-Term Memory* (BiLSTM) layers and a dense layer. An overview of the model is displayed Figure 4.2. 40 mel-scale log filter-bank coefficients, energy plus deltas and double-deltas are computed from the extracted vocals. To do so, a Hann window of 32 ms with a step size of 16 ms is used. The input feature sequences are downsampled by two sub-modules each composed of a 2D-convolutional layer (32 filters with kernel size 3×3), a ReLU activation function and a 2×3 max-pooling layer. The sequence length T is thus divided by four.

The recurrent part of the acoustic model is composed of 3 bidirectional LSTM layers with 256-dimensional hidden states and recurrent dropout of 0.1. An additional dropout of 0.1 is applied between each recurrent layer. Finally a time-distributed dense layer and a softmax activation function are applied to obtain the per-frame character probability vectors. The model is trained using the CTC loss implementation of [SRP18]. The loss is minimized using the ADAM algorithm with a learning rate of 10^{-4}, a batch size of 32 during 4000 training epochs with 250 steps per epoch. We use validation-based early stopping. For transcription, a prefix beam search decoding is employed, using a beam width of 100.

The language model $\mathscr{H}$ is a basic trigram word language model where the vocabulary is restricted to the 6000 most frequent words, thus reducing overfitting, generating around 370k trigrams. An approximated modified Kneser-Ney algorithm is used for smoothing

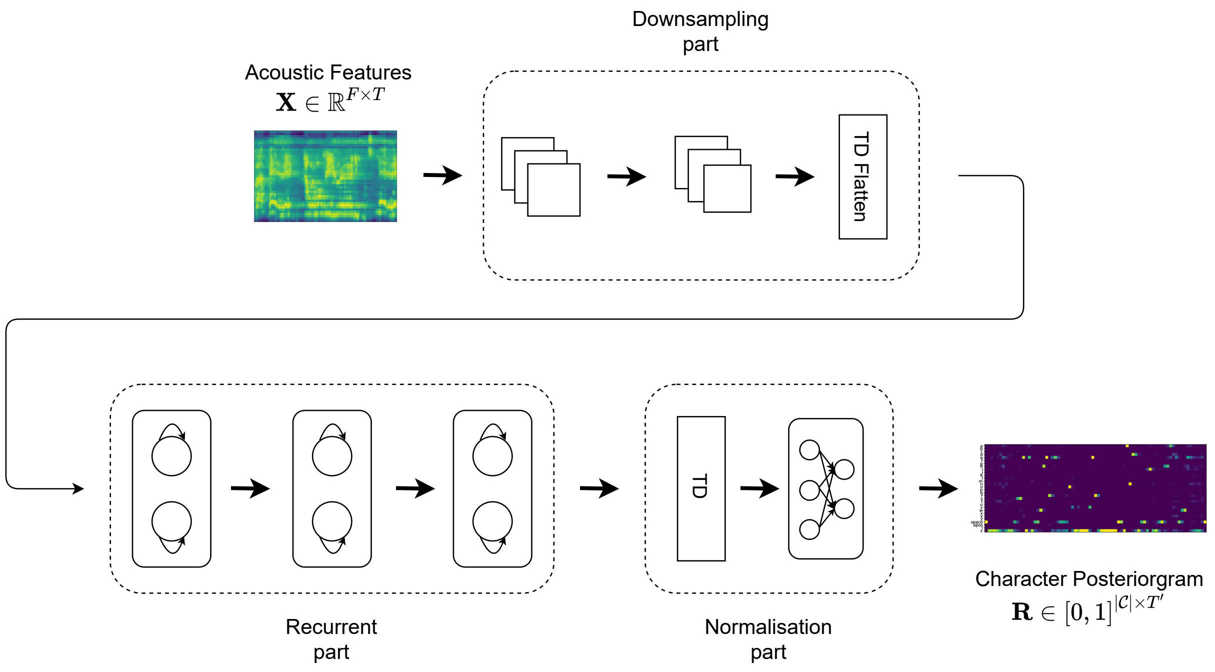


Figure 4.2 – Overview of the acoustic model $\mathscr{H}$. It is composed of three parts: the downsampling, the recurrent and the normalization. The downsampling section is made up of two convolutional layers, which are followed by a time-distributed flatten layer. The recurrent part is composed of three BiLSTM. The normalization part is constituted of a time-distributed layer and a softmax one outputting the desired characters' probability.

	Hansen	Mauch	Jamendo
Stoller19 [SDE19] (2019)	- (-)	70.9 (-)	77.8 (-)
Ours \w sep (2019)	73.93 (3.14)	65.78 (3.20)	82.29 (2.37)
Ours \wo sep (2019)	76.88 (2.22)	71.36 (2.52)	83.35 (2.05)
Gupta20 [GYL20] (2020)	**47.86 (6.30)**	**47.25 (3.67)**	60.98 (3.04)
Demirel21 [DAD21] (2021)	- (-)	49.92 (-)	**44.52 (-)**

Table 4.1 – Comparing our lyrics recognition reference system, with and without a prestep of SVS algorithm, to the state-of-the-art systems on various evaluation datasets. The WER metrics are given in percentage and the standard errors are displayed in parentheses when available. Gupta20 results are taken from the 2020 lyrics transcription challenge of *Music Information Retrieval Evaluation eXchange* (MIREX). The best results for an evaluation dataset are displayed in bold.

[NEK94]. It is trained using the kenLM toolkit [1]. This toolkit is specifically designed to efficiently create n-gram models. The models' parameters are chosen using the validation set. Pruning n-gram and various n values were tested without improvement. This language model is incorporated during decoding, after tuning language model weight and insertion penalty value on the validation set using the ctcdecode toolkit [2]. This toolkit is an implementation of the CTC beam search decoding algorithm.

4.4 Results

In order to evaluate our approach, we use a *Character Error rate* (CER) metric computed using the python Levenshtein toolkit [3] and a *Word Error rate* (WER) metric computed using the JiWER toolkit [4]. An example of the type of posteriorgram and resulting decoding obtained from our trained model is given in Figure 4.3. As expected, a sequence of spikes is observed, associated with detected characters, and separated by the ϵ token. The relatively low values of the WER and the CER seem to validate our implementation. To confirm our intuition of the SVS improving the E2E acoustic models lyrics transcription performance, we also train another model without any step of SVS.

Our systems' results are given in Table 4.1 with comparison to state-of-the-art architectures. Firstly, we can see our model trained with a preprocessing step of SVS consistently beats the one trained without it, though only providing slight improvements. This result still validates the use of this module and is another argument for the use of a SVS algorithm in the context of E2E acoustic models. Secondly, our system obtains comparable results with Stoller19 while being trained on a much smaller dataset. It is worth noting Gupta20 and our approach are both trained using the English part of DALI, while Stoller19 is trained using a private dataset of unknown quality.

1. `https://kheafield.com/code/kenlm/`
2. `https://github.com/parlance/ctcdecode`
3. `https://github.com/ztane/python-Levenshtein`
4. `https://github.com/jitsi/jiwer`

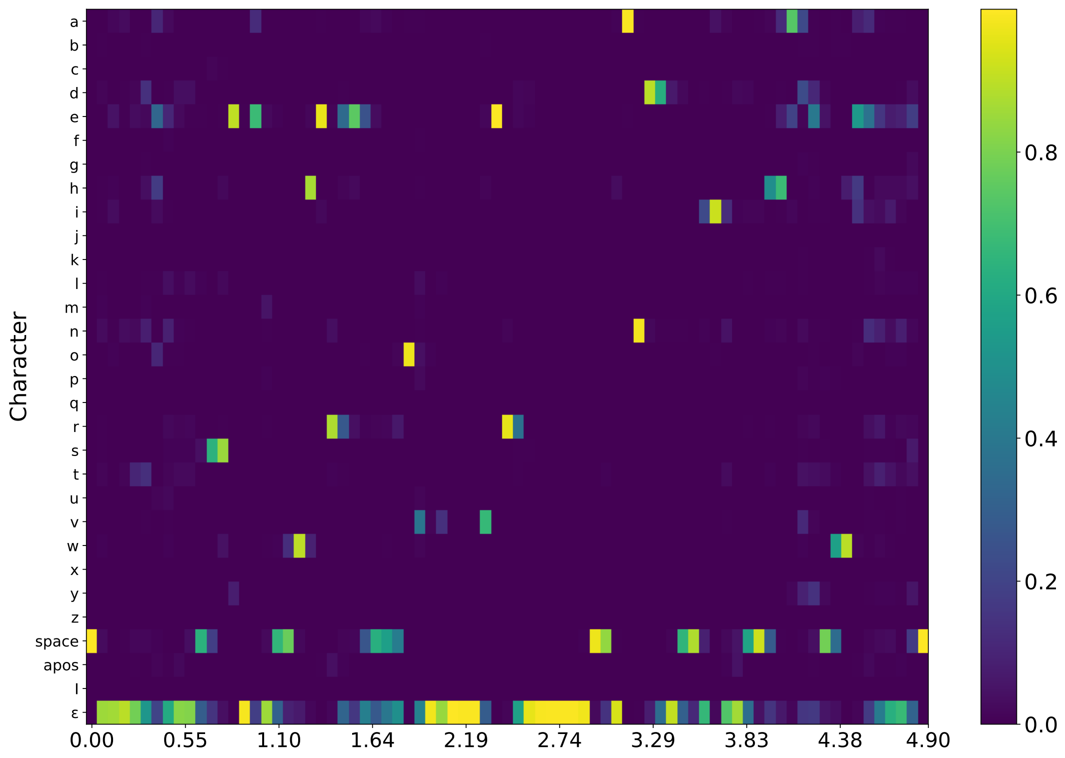


Figure 4.3 – An example of a posteriorgram inferred by our system. Ground truth: "to see we're over and I hate when". Obtained transcription with a prefix CTC beam search combined with a lyrics language model: "e see where over and I had we". Here, CER = 0.24 and WER = 0.5. Probability values are given in level of colors.

We can postulate that the quality of the DALI dataset annotations is higher, which could explain the on par performance reached by our implementation. Moreover, the dataset used to train Stoller19 is annotated at line level whereas the DALI dataset is annotated at word level. As described in Section 2.2, most recent implementations, *i.e.* Gupta20 and Demirel21 [DAD21] both significantly outperform the CTC-based architectures such as Stoller19 and ours.

Both approaches were published after the work on explicit content detection and were not considered for this study. Our SVR implementation displayed similar results to Gupta20 on the lyrics-to-audio alignment task and was then kept for the multilingual lyrics-to-audio alignment study. The trained multilingual architecture was subsequently used for the language identification task. Finally, for the cover detection task, we chose to adapt Gupta20 for our system since the approach is based on complete transcripts. Demirel21 was published at the end of the thesis and is solely displayed in the table for reference purposes.

Part II

Applications to MIR

Chapter 5

Explicit content detection

Explicit content detection is the task of classifying an audio recording as either *explicit* or *non-explicit*. It is particularly sensitive for streaming services for parental advisory. For over three decades [Rad19], a parental advisory label has been found on musical recordings when they include explicit content (*e.g.* lyrics potentially unsuitable for children). Such recordings included harsh language, description of acts of violence, substance consumption or sex.

As of today, this labeling is mainly done manually following a set of guidelines [Ind19]. This process is slow and hard to scale to industry-size catalogs. Existing automatic approaches are scarce and rely on the availability of the lyrics in text format. Lyrics transcriptions could be obtained from audio using *Singing Voice Recognition* (SVR) algorithms, but with relatively limited performance as we have seen in Section 1.2. The question of whether lyrics information extracted from audio can be leveraged to perform the explicit content detection remains open.

When lyrics are available, explicit content detection can be approximately achieved through assessing the presence of words from a fairly small specialized dictionary. In fact, as proven in [FCCG19], state-of-the-art deep neural network algorithms perform just slightly better than dictionary-based methods with suitable keywords. This suggests detecting a set of carefully chosen keywords directly from the audio signal is a valid strategy to perform the explicit detection task in the general case.

Keyword Spotting in audio is an actively studied task, with state-of-the-art systems achieving high performance on speech signals [TL18]. A few attempts to transfer them to singing voice have been proposed [FGO08, Kru18]. Kruspe et al. rely on a keyword-filler *Hidden Markov Models* (HMM) algorithm [Kru18]. This method requires synchronized annotations, at the frame level, between audio and text. Since no readily accessible dataset exists for polyphonic music at this granularity, such annotations are generated with an acoustic model trained on speech, by aligning textual lyrics to music, seriously hindering model performance.

Our proposed work is the first audio-based explicit content detection system in music. Our approach is based on an acoustic-to-character acoustic model, described in the previous chapter, and a keyword spotting model associated with a dictionary of carefully chosen keywords related to the explicit detection task. The decoding is directly performed on the

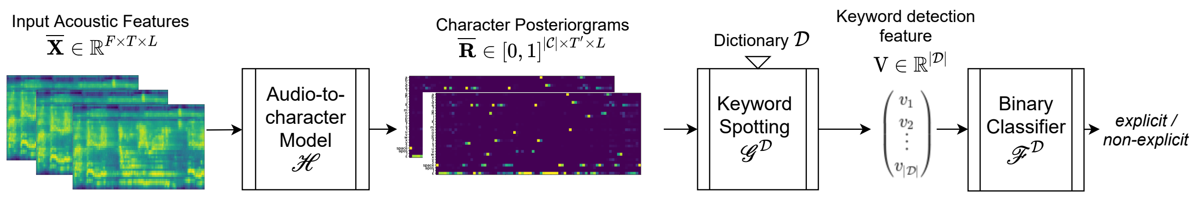

Input Acoustic Features
$\overline{\mathbf{X}} \in \mathbb{R}^{F\times T\times L}$
Audio-to-character Model $\mathscr{H}$
Character Posteriorgrams
$\overline{\mathbf{R}} \in [0,1]^{|\mathcal{C}|\times T'\times L}$
Dictionary $\mathcal{D}$
Keyword Spotting $\mathscr{G}^{\mathcal{D}}$
Keyword detection feature
$\mathrm{V} \in \mathbb{R}^{|\mathcal{D}|}$
v_1, v_2, $v_{|\mathcal{D}|}$
Binary Classifier $\mathscr{F}^{\mathcal{D}}$
explicit / non-explicit

Figure 5.1 – General overview of the proposed modular explicit content detection system. To obtain the input acoustic feature tensor, multiple preprocessing modules are applied on the input audio, for both training and inference. These modules are, in order of application: *Singing Voice Separation* (SVS), audio segmentation and feature extraction.

output of the audio-to-character model and, contrary to an *End-To-End* (E2E) keyword spotting approach like in [CPH14], new keywords can be added easily to the dictionary without retraining the model.

In [HLS15], the authors have demonstrated the better performance of this architecture over the keyword filler to do keyword spotting in speech. As described in the previous chapter, this method does not require to create neither a pronunciation lexicon nor frame level synchronized annotations. The *explicit* label is finally inferred by a binary classifier using the output of the keyword spotting system.

The chapter is divided into four sections. First of all, the proposed approach is presented in Section 5.1. Following that, the experimental parameters are provided in Section 5.2. Finally, results and discussion are followed in Section 5.3 and a conclusion is given in Section 5.4. A large part of this chapter is directly adapted from the paper: "Andrea Vaglio, Romain Hennequin, Manuel Moussallam, Gaël Richard and Florence d'Alché-Buc. Audio-Based Detection of Explicit Content in Music. *In Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2020".

5.1 Proposed approach

A general overview of our approach is given in Figure 5.1. A song X is sliced into L segments of equal size using a segmentation procedure. It is worth noting that L may be different for each song since it depends on the song duration. The system takes as input an acoustic feature tensor $\overline{\mathbf{X}} \in \mathbb{R}^{F\times T\times L}$ built from X, where T is the number of temporal frames per segment and F is the features dimension. Given a dictionary $\mathcal{D}$, the predictive model $\mathscr{L}^{\mathcal{D}}$ is the composition of three modules

$$\mathscr{L}^{\mathcal{D}}(\overline{\mathbf{X}}) = \mathscr{F} \circ \mathscr{G}^{\mathcal{D}} \circ \mathscr{H}(\overline{\mathbf{X}}) \tag{5.1}$$

5.1.1 General presentation

For a given input tensor $\overline{\mathbf{X}}$, the audio-to-character module $\mathscr{H}$ outputs a 3D-tensor $\overline{\mathbf{R}}$. It is simply based on the acoustic model described in the previous chapter, applied on each matrix of the tensor $\overline{\mathbf{X}}$. It takes as input $\mathbf{X}_l$, the l^{th} matrix of the tensor $\overline{\mathbf{X}}$, and outputs $\mathbf{R}_l$, the l^{th} matrix of the tensor $\overline{\mathbf{R}}$. The preprocessing modules, and theirs parameters, preceding the acoustic model, *i.e.* SVS, audio segmentation, feature extraction, are identical to those described in Chapter 4. For a given input tensor $\overline{\mathbf{R}} = \mathscr{H}(\overline{\mathbf{X}})$, a vector V is outputted by the keyword spotting module $\mathscr{G}^{\mathcal{D}}$, whose each coefficient v_n gives an estimate of the (log) posterior probability of $\boldsymbol{k}_n$, the n^{th} keyword of the dictionary $\mathcal{D}$, by averaging on all segments

$$v_n = \underset{\ell=1,\dots,L}{\text{mean}} \log \hat{P}(\boldsymbol{k}_n \mid \mathbf{R}_\ell), \tag{5.2}$$

where $\mathbf{R}_\ell$ is the ℓ^{th} matrix in tensor $\overline{\mathbf{R}}$. Other ways of aggregating probability through segments, *e.g.* using a max function, have been tested on a validation dataset and showed no improvement.

For a given input vector $\mathrm{V} = \mathscr{G}^{\mathcal{D}} \circ \mathscr{H}(\overline{\mathbf{X}})$, an *explicit* label $\mathscr{L}^{\mathcal{D}}(\overline{\mathbf{X}})$ is outputted by the binary classifier $\mathscr{F}$ discriminating the content of the song X as *explicit* or *not-explicit*. Among these three modules, only the modules $\mathscr{H}$ and $\mathscr{F}$ require training. Training the module $\mathscr{H}$ boils down to training the acoustic model of Chapter 4. Training the module $\mathscr{F}$ requires to apply the preprocessing $\mathscr{G}^{\mathcal{D}} \circ \mathscr{H}$ to the training dataset $\{(\overline{\mathbf{X}}_i, y_i)_{i=1}^{n_{songs}}\}$ containing songs annotated by *explicit/non explicit* labels.

5.1.2 Keyword detection

Following the work of [HLS15], we implement the keyword spotting module $\mathscr{G}^{\mathcal{D}}$ as a *Connectionist Temporal Classification* (CTC)-based decoding function. For a given searched keyword $\boldsymbol{k}$, we consider $\boldsymbol{k'}$ which is the keyword $\boldsymbol{k}$ with an ϵ at the beginning, end, and between every character to allow the use of ϵ during decoding. A decoding network is then created the same way as described in 3.5. The goal of the decoding function is to find the path in the network maximizing the CTC scoring for the keyword $\boldsymbol{k'}$. The coefficients $\alpha_{s,t}$ are afterwards computed using the same dynamic programming algorithm. Finally, the keyword probability is given at each step t by:

$$P(\boldsymbol{k}, t) = \alpha_{|\boldsymbol{k'}|,t} + \alpha_{|\boldsymbol{k'}|-1,t} \tag{5.3}$$

We consider the detection score s of the keyword $\boldsymbol{k}$ to be the maximum of the keyword probability over all time steps with:

$$s(\boldsymbol{k}) = \max_{t=1,\dots,T'} P(\boldsymbol{k}, t) \tag{5.4}$$

With these computation rules, we empirically found that we can only find keywords at the beginning of segments. In practice, keyword probabilities after the first sung word in the recording are artificially low. To prevent this and allow the keyword detector to be fired at any time, we choose to reinitialize the first node at each time step

$$\alpha_{1,t} = P(k'_1, t \mid \mathbf{X}_\ell) \tag{5.5}$$

As the naive CTC scoring is numerically unstable, computations are done in log-space using the log-sum-exp trick [Han17].

5.2 Experiments

5.2.1 Explicit Dataset

To train the binary classifier $\mathscr{F}$, we use an internal dataset built from the metadata available at Deezer. In this chapter, we restrict ourselves to recordings with English lyrics. Songs are either labeled *explicit* or *non-explicit*. These labels are obtained from external providers and are known to be noisy, yet we manually verified this noise level remains low. We discard tracks also present in the DALI dataset to avoid overfitting. We notice the music genre distribution of explicit tracks and non-explicit tracks is very different: rap is strongly over-represented in explicit tracks (40% of all tracks), but not in the non-explicit ones (few percent).

To avoid creating an explicit content detection relying mostly on the genre information, we sample both explicit and non-explicit tracks to obtain the same genre distribution for the two types of songs. The complete dataset then consists of 2600 non-explicit tracks and 2530 explicit ones. Finally, we make an artist aware split [Fle07] between training, validation, test of respectively 70%, 15%, 15%. We create another dataset the same way for dictionary creation. The dataset consists of 24250 non-explicit tracks and 24250 explicit ones. No songs are common between the two datasets.

5.2.2 Dictionary creation

Following the work of [KM19] we select the keywords of the dictionary $\mathcal{D}$ based on the explicit and not-explicit lyrics word distributions. To generate the dictionary $\mathcal{D}$, we use the importance I defined in [FCCG19]. For a chosen keyword, I is computed as the ratio between the frequency of the word in explicit and non-explicit lyrics. As in [FCCG19], we manually discard stop words, too common words, too rare words, onomatopoeia and abbreviations. The dictionary is constructed with 128 words with the highest importance.

5.2.3 Baseline

Our model is compared to two baseline systems. The first one is a classic *Convolutional Recurrent Neural Network* (CRNN) audio classifier. This architecture was successfully used in a variety of music classification tasks, like genre recognition [CFSC17] or music emotion recognition [MAD$^+$17]. Unlike our approach, this classifier tries to directly infer *explicit* labels from audio in an E2E manner. The model is composed of four convolutional layers, followed by one gated recurrent unit layer and a dense layer. For each input sample, values of mel-scale log filter-bank coefficients are extracted using a Hann window of 48 ms with a step size of 48 ms. The model is trained using a binary cross-entropy loss which is optimized using an Adadelta optimizer and a batch size of one. The model is trained for 3000 epochs with 450 steps per epoch. We use validation-based early stopping.

The second baseline is a dictionary lookup based on lyrics as in [FCCG19]. Given the dictionary $\mathcal{D}$, this method classifies a song as *explicit* if its lyrics contain at least one of the keywords in $\mathcal{D}$ and as *non-explicit* otherwise. Unlike our system, this baseline is informed by lyrics at test time. As such, this baseline can be considered as an oracle (*e.g.* providing an upper bound for performance) for our task of detecting explicit content from audio only. It is a safe assumption because as described in the beginning of this chapter, when lyrics are available, a simple lookup dictionary gives results as good as more sophisticated models.

5.2.4 Explicit module

For the binary classifier $\mathscr{F}$, we use a random forest classifier [Ho95]. The hyperparameters of the classifier are tuned using a first step of random search and a second step of grid search. The number of keywords of the dictionary, for our model and for the lyrics baseline, is tuned on the validation dataset. We report precision, recall and F1-score for the explicit class. Since the explicit content might be sensitive to certain audiences, emphasis is put, at highest F1-score, on the system maximizing the recall. We use this rule to optimize parameters on the validation dataset. The optimized number of keywords is 128 for the lyrics baseline and 32 for our model. The baselines and our approach with their optimized parameters are evaluated on the test dataset.

5.3 Results and discussion

5.3.1 Preliminary results

The performance of decoding with the dictionary $\mathcal{D}$ is assessed on the DALI test set. Results show 75% of $\mathcal{D}$ keywords, with at least one occurrence in the test dataset, have a ROC-AUC greater than 0.81. This metric is favored over the PR-AUC one as being independent of the class distribution of the considered keyword in the test set. Thus ROC-AUC values obtained for different keywords with various frequencies in the evaluated dataset can be compared which is not the case for the PR-AUC metric. Properties of both metrics are extensively described in [Faw04].

Being the first time such a metric is computed for keyword spotting in the singing case, we cannot compare it to other results. However, since these values are significantly higher than the random, the feature vector V carries some information on the presence of keywords of the dictionary $\mathcal{D}$ in a considered input music. An example of decoding is displayed in Figure 5.2. The example is 'positive', as the searched keyword is indeed present in the ground truth character sequence. A decoding line is visible in the figure, the position of 'space' characters delimiting the decoding line (3.94s to 4.28s) are quite close to the ground truth position of the word (3.9s to 4.29s). This figure suggests that the acoustic model $\mathscr{H}$ has learnt how to use the 'space' appropriately and to determine word positions. This is consistent with results found for lyrics-to-audio alignment [SDE19].

5.3.2 Explicit content detection results

Results are reported in Table 5.1. Scores reached by the lyrics baseline are similar to those found in [FCCG19]. The performance of a naive audio baseline on this challenging

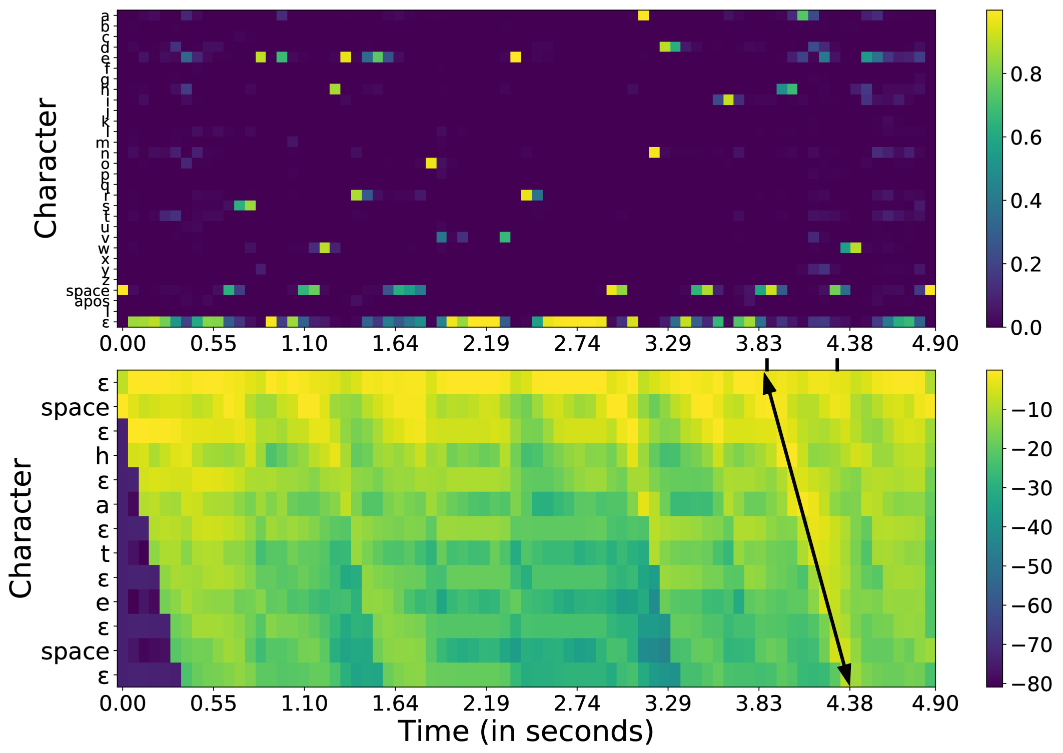


Figure 5.2 – A positive sample for keyword "hate". Top: posteriorgram $\mathbf{R}_\ell$ inferred by acoustic model $\mathscr{H}$. Probability values are given in level of colors. Bottom: Decoding matrix composed of coefficients α, described in Section 5.1.2. Values of α coefficients are given in level of colors. The decoding line and ground truth position of keyword are displayed in the figure.

Metrics	Audio baseline	Our system	Lyrics baseline
Precision	.61 (.02)	.63 (.02)	**.65 (.02)**
Recall	.59 (.02)	.73 (.02)	**.84 (.02)**
F1-score	.60 (.02)	.67 (.02)	**.73 (.02)**

Table 5.1 – Results for explicit detection task on the test set; the standard errors are given in parentheses. The best performance of each metric is displayed in bold.

task is significantly outperformed by our modular approach. While still not equivalent to a lyrics-informed scenario, these results are encouraging and show the validity of the proposed method. Still, the performance of these systems are insufficient to be deployed without human oversight. In [FCCG19], the authors argue that the explicit detection task is an inherently hard task. They state explicit content detection is a highly subjective task depending on both social, cultural and temporal context. They thus propose using these systems as tools to help annotators doing the final labelling.

5.4 Conclusion

We address the novel task of explicit musical content detection from audio only. Despite the task being challenging, our proposed modular approach yields promising results. Moreover, the system's decisions can be explained in terms of specific keyword presence probability which is a desirable property given the sensitivity of the task. Future work could investigate keyword decoding augmentation with a character level language model as in [HPR+17].

Chapter 6

Multilingual lyrics-to-audio alignment

Lyrics-to-audio alignment aims at synchronizing lyrics text units like paragraphs, lines or words to the timed position of their appearance in the audio signal. Tools dedicated to this task have many practical applications: they can be applied to generate new annotated data to train more robust singing voice recognizers [Kru18]; or be used as building blocks in specific applications like karaoke [Mes12], navigation within songs [FGOO11] or explicit lyrics removal [Kru16a].

Recent studies have proposed efficient alignment methods for singing voice [SDE19, GYL20], opening the door to practical applications. Results on the lyrics-to-audio alignment task of the *Music Information Retrieval Evaluation eXchange* (MIREX) challenge have notably greatly improved through the years, as displayed in Figure 6.1. However, previous studies mostly focus on the English language, for which annotated data is abundant. The question of their ability to generalize to other languages, especially in low (or even zero) training resource scenarios, has not been properly addressed.

Arguably a monolingual evaluation is unrepresentative of the variety of music recordings available in large scale collections. Commercial streaming services commonly serve content in hundreds of languages and a non-negligible number of popular songs even have multilingual lyrics [DB08]. However, annotated data on this type of content is scarce. A source of inspiration comes from the related field of multilingual speech recognition [WHH17] where transfer learning methods [CBL+19] have been shown to improve the performance on languages with few to zero training data. However, this improvement on low-resource languages can sometimes be detrimental to the efficiency on languages with more resources [WHH17].

The goal of this chapter is to evaluate and extend state-of-the-art lyrics-to-audio alignment methods to a language-independent setup. It is the first attempt to create a language-independent lyrics-to-audio alignment system. To do so, we review the fitness of these approaches to the multilingual framework. Then, we focus on one architecture and study two key features likely to allow generalization to several languages: 1) the intermediate representation space (character versus phoneme) and 2) the design of the training dataset. Evaluation is performed on multiple datasets, from diverse sources, languages and scripts, with various amounts of data available, from plenty to zero.

We notably consider the specific case of songs with multilingual lyrics generally referred to

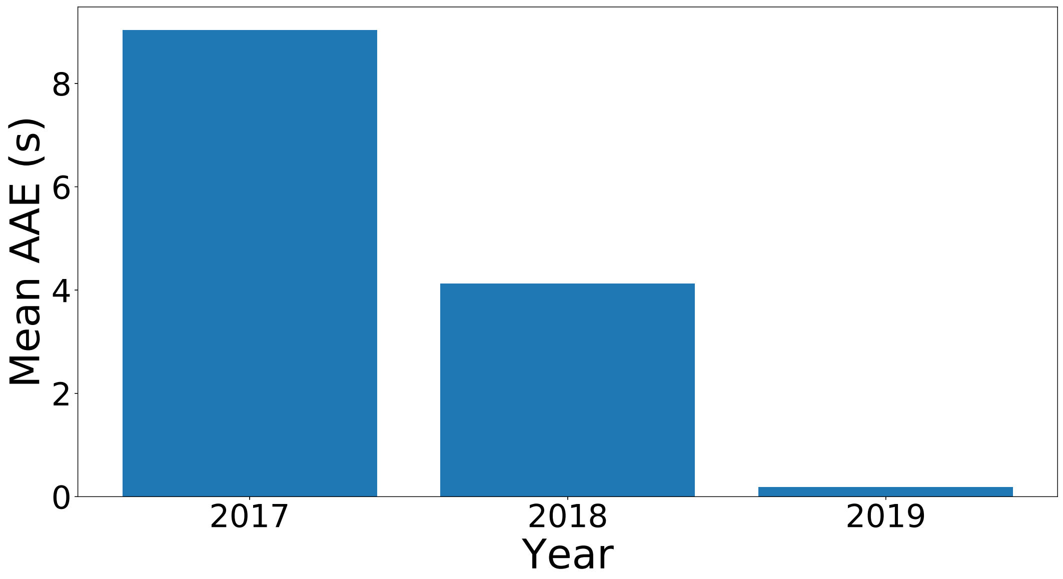


Figure 6.1 – Performance of the system obtaining the best results during the MIREX challenge on the Mauch dataset through the years. Performance is quantified using the AAE metric described in Section 6.1.1. AAE metric is better if smaller. Metric is averaged on all tracks on the Mauch dataset.

as code-switching in the *Automatic Speech Recognition* (ASR) literature. Code-switching denotes the case when two or more languages are used in the same song. Two cases of code-switching exist: intra-sentential and inter-sentential. In the first case, the change of language happens inside one line of lyrics. The second case is when the switch occurs between two lines of lyrics. This problem has been well researched for speech [LLY$^+$19] but has never been addressed for singing. First results for this type of data are thus presented. Finally, we try leveraging phoneme similarity information during alignment and discuss obtained results. Unfortunately, no method used is shown to significantly improve performance.

The chapter is organized in six sections. First and foremost, some background is provided in Section 6.1. We next describe the proposed method in Section 6.2. The experimental setup and results are described respectively in Section 6.3 and Section 6.4. Afterwards, incorporating phoneme similarity information during alignment is attempted in Section 6.5. Finally, conclusions are drawn in Section 6.6 and future work is discussed. A large part of this chapter is directly adapted from the paper: "Andrea Vaglio, Romain Hennequin, Manuel Moussallam, Gaël Richard and Florence d'Alché-Buc. Multilingual Lyrics-to-Audio Alignment. *In Proc. of Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2020".

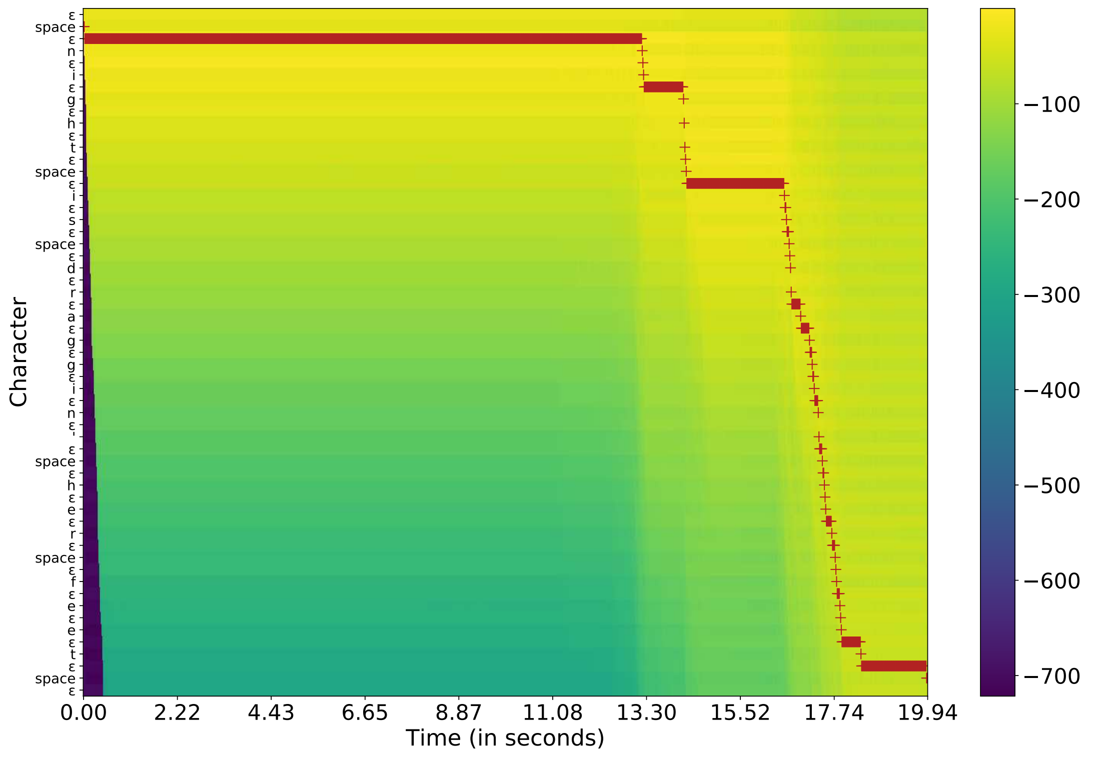


Figure 6.2 – One example of lyrics-to-audio alignment. Here, lyrics to align are "Night is draggin' her feet". The characters from lyrics are mapped to audio across time. Values of coefficients α, defined in Section 6.2.3, are described in level of colors.

6.1 Required background

6.1.1 Lyrics-to-audio alignment

General presentation:

Singing voice alignment methods are typically inspired from text-to-speech alignment systems. Classically, an acoustic model is trained and used to force text to audio alignment using a Viterbi algorithm [WT97]. The acoustic model's output is also utilized to conduct the alignment in the most recent lyrics-to-audio systems [SDE19, GYL20]. The efficiency of these systems is then largely dependent on the quality of the trained acoustic model. An example of such an alignment is displayed in Figure 6.2. State-of-the-art approaches for lyrics alignment were compared in the MIREX 2019 challenge[1]. Two submitted architectures showed particularly strong performances. The first one is Stoller19 [SDE19]. The second one is Gupta20 [GYL20], which obtained the best results on the challenge. Both systems are extensively described in Section 2.2.

1. https://www.music-ir.org/mirex/wiki/2019:Automatic_Lyrics-to-Audio_Alignment

Extending systems to multilingual framework:

Although it achieved the best performance in the MIREX challenge, Gupta20's framework can not be straightforwardly used in a multilingual setup: it is composed of multiple parts, some of them, such as the pronunciation lexicon, being specific to English. To be able to use it in a new language, it would require modifying or retraining these parts. In comparison, Stoller19 is based on an *End-To-End* (E2E) acoustic model, trained with the *Connectionist Temporal Classification* (CTC) algorithm, that directly outputs characters. It is more suitable to perform multilingual lyrics-to-audio alignment as it can be theoretically applied to any language based on the same script (writing system) as the training language. We thus choose to base our multilingual generalization study on the acoustic model trained in Chapter 4, being inspired by Stoller19.

Employing characters may not be optimal for multilingual lyrics-to-audio alignment: authors in [SDE19] suggest using phonemes as an intermediate representation could be more relevant for aligning songs in other languages. They argue that, for phoneme based systems, only the pronunciation lexicon has to be replaced for a new language, while a character based approach is limited by the set of characters the acoustic model outputs. For instance, the acoustic model described in Chapter 4 can only be used to align songs in Latin script languages. It could be extended with characters from scripts of new languages, as in [TSW+18], but it would require a complete retraining each time a new script is added in the language pool.

Using phonemes as an intermediate representation, any language can be theoretically aligned for any trained model as long as a pronunciation lexicon is available. Character-based and phoneme-based intermediate representations are then considered for our multilingual system. This system is evaluated using classic metrics of the lyrics-to-audio alignment task described in the next section.

Metrics:

To evaluate our approach, we use the *Average Absolute Error* (AAE) [MV08] metric, denoted as σ. For its calculation, the absolute difference between the actual start of the word timestamp t_i and its estimation $\hat{t_i}$ for each word $\boldsymbol{w}_i$ of the considered track is calculated, where $\boldsymbol{w}_i$ is the i^{th} word of this track. The final error score for a song is obtained by averaging over all word-level errors by:

$$\sigma = \frac{1}{N}\sum_{i=1}^{N} |\hat{t_i} - t_i| \tag{6.1}$$

where N is the number of words in the considered song. A known issue of this metric is its perceptive dependence on tempo: one absolute error will not be perceived the same if the tempo is fast or slow [Dzh17]. The *Percentage of Correct Onsets* (PCO) [MFG12] metric, denoted by ρ_τ, was proposed to mitigate this effect. It is computed as the percentage of starts of the word timestamps whose estimation $\hat{t_i}$ are below a certain distance τ from the ground truth t_i. This metric reflects that errors below this threshold fall within human listeners' perceptual tolerance. More precisely, it assumes humans still perceive as synchronous lyrics and audio whose onsets are separated by this threshold. Formally, it is defined as:

$$\rho_\tau = \frac{1}{N} \sum_{i=1}^{N} 1_{|\hat{t_i} - t_i| < \tau} \times 100 \tag{6.2}$$

Both metrics are classic metrics of the MIREX lyrics-to-audio alignment challenge. The PCO threshold is fixed to 0.3 seconds as in this challenge. We will discuss the perceptual validity of this threshold in the next chapter. They are computed using the same evaluation script as the one used for the challenge [Dzh17] [2].

As described in Section 2.4.2, in some evaluation datasets, for several songs, annotations appear to have been offsetted with a constant offset in order to enhance visual display above precise synchronization. This shifting of annotations causes issues to correctly estimate the PCO metric for these tracks. In these cases, making the assumption a system is aligning lyrics perfectly to the audio, if the offset is superior to the value of the threshold of the PCO (here 0.3 seconds), a metric of zero is obtained. Correcting the existing annotation offset is therefore primordial, for these tracks, to obtain a PCO metric correctly estimating the performance of the system.

For each track of the considered dataset, the offset is corrected automatically as the value maximizing the PCO of the given track after alignment. This type of correction undoubtedly delivers optimistic value to the PCO metric, but allows for the generation of a value for the metric representing more properly the system performance by automatically correcting offsetted annotations. Moreover, in the case where the system is performing badly for alignment, the metric should remain low even after the offset correction. This offset correction is only applied during evaluation to datasets where offsetted songs exist. In this chapter, this was the case for all datasets created from the Deezer catalog.

6.1.2 Computing phoneme similarity

The *International Phonetic Alphabet* (IPA) phoneme similarity could be used to increase the alignment performance by allowing for greater information transfer across phonemes. Indeed, for a phoneme present in the audio rarely seen in the training dataset, an acoustic model might have a tendency to (incorrectly) detect a more common similar phoneme, making alignment harder. In this case, the acoustic model could nevertheless give the uncommon phoneme a significant score based on phoneme similarity. In [RÁA14], authors notably show that taking this information into account improves long audio alignment for automatic subtitling. Methods for computing phoneme similarity are typically expert knowledge based or data-driven. Perceptually based techniques are another type of approach that will be briefly discussed at the conclusion of this section.

Knowledge based methods:

Generally, expert knowledge based methods are focused on phonological features. They define, for a given feature theory, a phoneme-similarity function taking as input a pair of phonemes. Classically, a Hamming distance is used between respective feature vectors of

2. https://github.com/georgid/AlignmentEvaluation

phonemes like in [DK16]. Each dimension of a feature vector indicates if a corresponding feature is present, not present, or not applicable for its corresponding phoneme. The Hamming distance then computes the number of features for which two phonemes differ. Recently, more complex ways of computing similarity using features were used as detailed in [LCC+18].

Data-driven methods:

Data-driven methods to compute phoneme similarity are numerous. First of all, similarity can be computed by directly comparing the acoustic realizations of two phonemes using algorithms such as *Dynamic Time Warping* (DTW) [HMN16]. Another classic method is to compute the distance between the *Hidden Markov Models* (HMM)s modeling monophones. For example, in [SB01], this type of distance is used to map phonemes from a target language to a source language. In this study, an acoustic model is produced for a target language, where few training data is available, by employing an acoustic model that has previously been trained on a significant amount of data from a source language.

To do so, one HMM is trained for each language given the available training data. After computing the distance between all source and target monophones, each target phoneme is then mapped to the nearest phoneme of the source language. Evaluation is finally performed using the source language acoustic model, with its mapped outputs, on the target test datasets. Results are on par with an expert-based phoneme mapping.

Phoneme similarity can also be computed using the confusion matrix of an acoustic model. This matrix is generally computed using ground truth annotations. The confusion matrix is computed by forced alignment of the ground truth phoneme annotations to the recognized sequence of phonemes using a dynamic programming algorithm (*e.g.* the Viterbi algorithm). This matrix is afterwards transformed to a similarity matrix using the Houtgast algorithm as in [LP12].

In this method, we consider the model is more prone to confuse phonemes if they are similar. However, one drawback of this method is that the quality of the obtained matrix is largely dependent on the quality of the acoustic model used to generate it. In [LP12], the authors generate such a matrix from the output of a hybrid system combining a HMM and a *MultiLayer Perceptron* (MLP) network. This matrix is then clustered to obtain clusters of phones. Taking into account this information during phoneme recognition is shown to improve the performance over a baseline. The authors also expose that human made natural classes are similar to the computed ones.

The phoneme similarity can also be computed by using phoneme embeddings. In [SMH18], the authors use various models, like word2vec [MCCD18], to generate such embeddings. The models were trained using around 10000 lines from each of the three considered languages: Finnish, Spanish and Turkish. The pairwise similarity of two phonemes are then simply computed using the cosine similarity between their respective embeddings. The authors demonstrate a significant correlation between this embedding space and a phonological distinctive feature space. They also show that this space can capture some proportional analogies (*e.g.* [p] is to [b] as [t] is to [d]).

In [PDvG17], the authors demonstrate that the phoneme embeddings, learned by their multilingual neural grapheme-to-phoneme model, also contain many regularities. Their model is a *Bidirectional Long Short-Term Memory* (BiLSTM) encoder-decoder network, trained with an attention algorithm, using $600k$ words from 311 languages of 42 scripts and outputting more than 450 IPA symbols.

Perceptually based methods:

Perceptual-based techniques are another sort of method for calculating phoneme similarity. In this case, the similarity is quantified directly in a perceptual experiment. In general, a group of participants is asked to listen to audio samples of phonemes that have been distorted by noise and subsequently identify the supplied phonemes. The perceptual judgments are next collected in a confusability matrix as in [EA18]. Nonetheless, such experiments are complex to set up and present a number of sources of variability that must be considered, as detailed in [LCC+18]. The results of previously conducted studies might be gathered; however, due to the protocol's intricacy, this sort of experiment is typically limited to a restricted selection of phonemes (around ten).

6.2 Proposed approach

A general overview of the proposed system is described in Figure 6.3. It is composed of two parts: an acoustic model $\mathscr{H}$ and a lyrics-to-audio alignment procedure. It takes as input an acoustic feature matrix $\mathbf{X}$, its corresponding lyrics $\boldsymbol{y}$ and outputs the synchronized lyrics $\hat{\boldsymbol{y}}$. The preprocessing modules, and theirs parameters, preceding the acoustic model, *i.e.* *Singing Voice Separation* (SVS) and feature extraction, are identical to those described in Chapter 4. The acoustic model is also akin to the one trained in Chapter 4, *i.e.* a *Recurrent Neural Network* (RNN) trained with the CTC algorithm. The set of outputs of the acoustic model is either the characters of the Latin alphabet or phonemes of an universal phoneme set. The latter is usually defined, in the literature, as the resulting set of the concatenation and the merging of phoneme sets of multiple languages. Lyrics-to-audio alignment is performed on the outputs of the acoustic model by a CTC-based alignment decoding function.

6.2.1 Acoustic model

The acoustic model $\mathscr{H}$ is the same as the one described in Chapter 4, with the exception that we do not include the convolutional layers to maintain the maximum temporal precision possible for alignment. CTC-based acoustic models were successfully used for multilingual speech recognition [TSW+18, MSW17]. Moreover, as described in Section 3.5, the BiLSTM layers allowed this type of model to provide reliable alignments.

On the specific problem of code-switching in speech recognition, CTC based architectures have been preferred over models capable of learning a language model over data [LLY+19] (*e.g.* attention ones). The dependence of one output on the preceding outputs makes switching from one language to another challenging in these models. On the contrary, the CTC based systems are trained under the conditional independence assumption, as

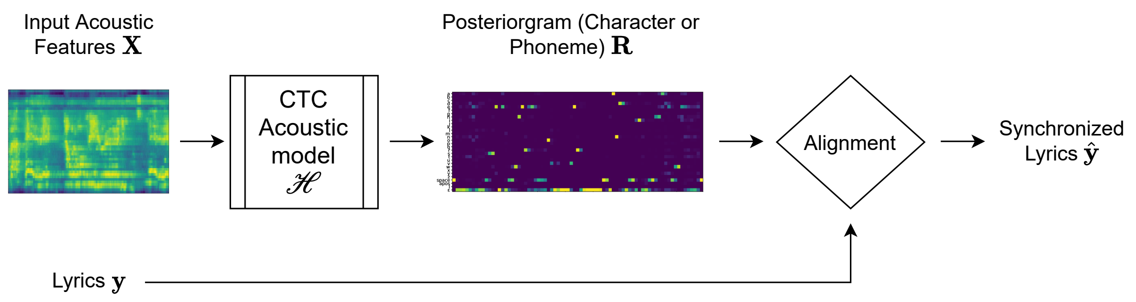

Input Acoustic Features $\mathbf{X}$
CTC Acoustic model $\mathscr{H}$
Posteriorgram (Character or Phoneme) $\mathbf{R}$
Alignment
Synchronized Lyrics $\hat{\mathbf{y}}$
Lyrics $\mathbf{y}$

Figure 6.3 – Overview of the lyrics-to-audio alignment system. Our study focuses on the training of the acoustic model in Section 6.2.1 and the design of the intermediate posteriorgram representation space in Section 6.2.2. The alignment block is described in Section 6.2.3. To obtain the input acoustic feature matrix, multiple preprocessing modules are applied on the input audio, for both training and inference. These modules are, in order of application: SVS and feature extraction. They are not displayed here for the sake of readability.

described in Section 3.5, which makes them more suitable for the code-switching case, all outputs being considered independently.

6.2.2 Intermediate representation

We consider two different intermediate representations for our system. The first one is a character set, here the Latin alphabet. As described in Section 3.4.1, this representation does not need any kind of expert linguistic knowledge as the acoustic model directly outputs character probabilities. However, such a representation is not suitable to perform alignments of songs in a language with a different script. To process those, the acoustic model would need to be retrained with new data on the given script. Moreover, even for languages sharing the same script, a character-based representation is sub-optimal for transferring knowledge between languages, as character pronunciations can significantly differ from one language to another.

Our approach relies on the following remarks: all languages share some common phonemes and phonemes are considered to be language independent [SW01], *i.e.* to be pronounced the same way across languages. Therefore, using an universal phoneme set as an intermediate representation could take advantage of the similarity between sounds across languages by utilizing consistent phonemes throughout training languages. Furthermore, using a large language training pool, most phonemes from unknown languages are expected to occur in training languages.

It can be achieved using the IPA symbols presented in Section 3.4.2. IPA pronunciations of words from all languages can be obtained using *Grapheme-To-Phoneme* (GTP) tools. Such tools are available for most common languages. The universal phoneme set is created by merging the phoneme sets of all considered languages in this chapter based on their IPA symbols.

6.2.3 Alignment

In order to align a song to its corresponding lyrics $\boldsymbol{y}$, the audio is sliced into segments of five seconds with a step size of 2.5 seconds. A posteriorgram is generated by the trained acoustic model for each segment. To obtain the final posteriorgram, all segment posteriorgrams are concatenated, after cropping them to half of their duration centered in their middle. A posteriorgram $\mathbf{R} \in [0,1]^{|\mathcal{C}| \times T}$ is obtained, where $\mathcal{C}$ is the set of symbols supported by our acoustic model, either characters or phonemes, and T is the number of temporal frames of the song. Alignment annotations are finally predicted, using the generated posteriorgram $\mathbf{R}$ and lyrics $\boldsymbol{y}$, with a CTC-based alignment function.

This function is inspired from the CTC scoring algorithm presented in Section 3.5 and is akin to a Viterbi forced alignment [For73]. The Viterbi forced alignment is a simpler version of the Viterbi decoding algorithm where the possible paths in the decoding graph are limited to the lyrics symbols sequence. To allow the use of ϵ during decoding, $\boldsymbol{y}$ is extended to $\boldsymbol{z}$ by adding ϵ at the beginning, end, and between every unit. A decoding network of size $|\boldsymbol{z}| \times T$ is afterwards constructed from $\boldsymbol{z}$. The goal of the decoding function is to find the path in the network giving the most probable alignment $\hat{\boldsymbol{y}}$ of $\boldsymbol{y}$ given the posteriorgram $\mathbf{R}$. More precisely:

$$\hat{\boldsymbol{y}} = \underset{B(\hat{\boldsymbol{y}})=\boldsymbol{y}}{\arg\max} \prod_{t=1}^{T} P(\hat{y}_t, t) \tag{6.3}$$

where B is an operator removing blanks and repetitions from a sequence $\hat{\boldsymbol{y}}$. To do so, the network's node $\alpha_{s,t}$ is defined as the probability of the best alignment of the sub-sequence $\boldsymbol{z}_{1:s}$ after t frames. $\alpha_{s,t}$ scores can be efficiently calculated using a forward-backward algorithm, by merging together paths reaching the same node. $\alpha_{s,t}$ is then computed recursively from the values of α in the previous frames. Only transitions between blank and non-blank units, and between pairs of distinct non-blank units are allowed. The ϵ symbol at the beginning and end of the sequence being optional, there are two valid starting nodes and two final nodes. The coefficients α are initialized as:

$$\alpha_{s,1} = P(z_s, 1) \text{ for } s \in \{1,2\} \text{ and } \alpha_{s,1} = 0, \forall s > 2 \tag{6.4}$$

Recursion is given by:

$$\begin{aligned} \alpha_{s,t} &= \max_{\tau \in \{0,1\}} (\alpha_{s-\tau,t-1}) P(z_s, t), \text{if } z_s \in \{\epsilon, z_{s-2}\} \\ \zeta_{s,t} &= \underset{\tau \in \{0,1\}}{\arg\max} (\alpha_{s-\tau,t-1}) \\ \alpha_{s,t} &= \max_{\tau \in \{0,1,2\}} (\alpha_{s-\tau,t-1}) P(z_s, t), \quad \text{otherwise} \\ \zeta_{s,t} &= \underset{\tau \in \{0,1,2\}}{\arg\max} (\alpha_{s-\tau,t-1}) \end{aligned} \tag{6.5}$$

Then, the probability of the best alignment is given by:

$$P(\hat{\boldsymbol{y}}) = \max_{\tau \in \{0,1\}} (\alpha_{|z|-\tau,T}) \tag{6.6}$$

Alignment $\hat{y}$ can finally be computed with an inverse recursion. The initial unit is initialized such as:

$$\hat{y}_T = |\boldsymbol{z}| - \arg\max_{\tau \in \{0,1\}} (\alpha_{|\boldsymbol{z}|-\tau,T}) \tag{6.7}$$

Inverse recursion is given by:

$$\hat{y}_{t-1} = \hat{y}_t - \zeta_{\hat{y}_t,t} \tag{6.8}$$

Calculations are performed in log-space using the log-sum-exp trick [Han17] to avoid numerical instabilities. As some phonemes from the target languages can be unseen in the training languages, the acoustic model will be unable to predict them, resulting in all alignments having a probability of zero. To get rid of this problem, a small amount of uniformly distributed noise is added to all the entries of the posteriorgram, as suggested in [SDE19].

6.3 Experiments

6.3.1 Datasets

DALI dataset:

In this chapter, we consider several language subsets of the DALI dataset. They are described in Table 6.1. Experiments are conducted using five source languages for the initial multilingual system development. These source languages are: English, German, French, Spanish and Italian. English is considered as a high-resource language. The four other languages are considered low-resource languages. The split between train, validation and test datasets for the first five languages is an artist aware split [Fle07]. We also consider four additional target zero-resource languages: Portuguese, Polish, Finnish and Dutch. Data from these languages is only used for evaluation. The splits of the different language data, *i.e.* DALI ids belonging to each dataset, are made publicly available[3].

One dataset, that we name *5lang*, is created for multilingual training. The training and validation sets of this dataset are generated by simply concatenating the training and validation sets of the five source languages. This dataset is largely unbalanced, English data dominating the corpus. Balancing the dataset with oversampling is tried without modification on the performance of the multilingual trained model when evaluated on low-resource and zero-resource language datasets. Similar results are also found for speech in [ATS16]. Besides, when tested on the English language dataset, it drastically worsens results. These results are expected as the quantity of English data being far superior in comparison to other languages in the *5lang* dataset, diminishing their importance could only degrade results for the multilingual trained model when evaluated on the English dataset. Results of multilingual models trained with a balanced dataset are displayed in Appendix A.

The procedure to generate training samples and corresponding labels for the acoustic model is the same as the one described in Chapter 4. For phoneme models, the phoneme

3. https://github.com/deezer/MultilingualLyricsToAudioAlignment

Language	# Phonemes	Train (h)	Test (h)
English (en)	44 (5)	210.8	34.2
German (de)	44 (1)	20.12	2.6
French (fr)	42 (0)	10.3	1.0
Spanish (es)	35 (3)	9.0	1.2
Italian (it)	33 (0)	9.2	1.3
Portuguese (pt)	37 (0)	-	1.8
Polish (pl)	31 (2)	-	4.3
Finnish (fi)	25 (0)	-	3.2
Dutch (nl)	41 (2)	-	3.2

Table 6.1 – Description of DALI language subset datasets used in this chapter and corresponding phoneme dictionary sizes. In parentheses are displayed the number of phonemes only occurring in the given language and its ISO 639.1 code.

sequence associated with a segment is generated from its corresponding character sequence using Phonemizer[4]. Phonemizer includes GTP tools for most common languages. It decomposes each word into a sequence of IPA symbols. To create the phoneme dictionary of one given language, we collect all the IPA phonemes present in the corresponding dataset. For simplicity, we did not consider IPA symbols other than vowels and consonants.

Sizes of dictionaries of phonemes of each language are given in Table 6.1. After concatenating and merging the nine dictionaries, we obtain an universal phoneme set of 62 phonemes. The language sharing factor [SW01] for the nine languages we use is 5.35. It means, on average, one unit of the universal phoneme set is shared by five to six languages of the language pool. This supports the fact that the IPA phonemes collected are rather consistent across languages considered in this chapter.

Deezer dataset:

We also evaluate our lyrics-to-audio alignment systems on a variety of large datasets built from the Deezer dataset. These datasets are used to see whether the type of results observed on DALI language datasets are consistent when bigger, more realistic and more diversified datasets are evaluated. Notably, an Indonesian dataset is created to evaluate our systems on a non-European language. However, the Indonesian language uses Latin script.

A Japanese dataset is then also created to evaluate our systems on a non-Latin script language. In this scenario, for character architectures, the lyrics are transliterated to Latin script using Pykaksi[5]. Phonemes are directly computed from hiragana script lyrics. Large English and Italian datasets are also constructed. Datasets are respectively composed of 1481, 450, 936 and 1084 tracks. They are mainly composed of classic western genres as the one present in the DALI dataset. No artist in this dataset is present in the DALI dataset to avoid artist bias.

4. https://github.com/bootphon/phonemizer
5. https://pypi.org/project/pykakasi/

Code-switching dataset:

To evaluate our systems on code-switching cases, we gather a dataset of 281 songs from the Deezer dataset, where both French and English languages occur. It is principally composed of cases of inter-sentential code-switching. No artist in this dataset is present in the DALI dataset.

6.3.2 Parameters of acoustic models

We use the same architecture for all acoustic models. Several sets of regularization and architecture size parameters were tested without a clear impact on the performance. Except for convolutional layers, the parameters of architecture and training are the same as those used in Chapter 4. The only other change that may be made is the intermediate representation. The acoustic model produces either character probabilities or IPA phonemes. In the first case, the set of outputs is the same as the one described in Chapter 4. More precisely, it is the concatenation of the Latin alphabet, the apostrophe, the instrumental token, the space token and the CTC blank symbol ϵ. A set of size 30 is obtained. In the second case, it consists of the universal phoneme set, plus the instrumental token, the space token and the CTC blank symbol ϵ. A set of size 65 is obtained.

6.4 Results and discussion

6.4.1 Preliminary studies

To validate our implementation, we first compare our approach with two state-of-the-art methods. Results are collected from the 2019 MIREX lyrics-to-audio alignment challenge. The systems presented at the 2020 challenge are not considered as they do not show any significant improvement. For this comparison, we choose character as an intermediate representation and the English dataset for training. We use the standard evaluation datasets for the lyrics-to-audio task described in Section 2.4. All three datasets are annotated with start-of-word timestamps.

The results are summarized in Table 6.2. Our system's results are close to those of the Gupta20, with no significant differences for the PCO metric on the three evaluation datasets. Although we use an architecture somewhat similar to Stoller19, we report a significantly better performance. It certainly could be explained as the quality of data used to train Stoller19 being of lower quality, as assumed in Section 4.4. Moreover, the annotations of this dataset are at line level whereas the DALI dataset is annotated at word level.

6.4.2 General results

Results of multilingual generalization experiments on the DALI subset datasets are displayed in Figure 6.4. Precise numerical values are reported in Appendix A. Several conclusions can be drawn:

Dataset	System	Mean AAE (s)	Mean PCO (%)
Hansen	Stoller19 [SDE19]	0.39 (0.12)	88 (3)
	Gupta20 [GYL20]	**0.10 (0.03)**	**97 (1)**
	Our system	0.18 (0.05)	95 (2)
Mauch	Stoller19 [SDE19]	0.26 (0.04)	87 (2)
	Gupta20 [GYL20]	**0.19 (0.03)**	**91 (2)**
	Our system	0.22 (0.03)	**91 (2)**
Jamendo	Stoller19 [SDE19]	0.38 (0.11)	87 (3)
	Gupta20 [GYL20]	**0.22 (0.06)**	**94 (2)**
	Our system	0.37 (0.12)	92 (2)

Table 6.2 – Comparison between our character based architecture trained with the English part of DALI and state-of-the-art systems on standard lyrics-to-audio alignment evaluation datasets. AAE is better if smaller, PCO is better if larger. The standard errors over tested songs are given in parentheses. The best metrics for an evaluation dataset are displayed in bold.

• **Using a multilingual training set helps.** For both character and phoneme based architectures, the model exhibiting the best multilingual generalization is trained with the multilingual dataset. In fact, this model significantly outperforms the ones trained on English on low-resource and zero-resource languages without degrading the performance on English. With phoneme as intermediate representation, it even improves results on English. On low-resource languages, the multilingual trained model obtains results on par with models trained uniquely on the target language (*e.g.* French trained model on the French dataset). It is worth noticing the multilingual training dataset is only marginally larger than the English one. The performance difference is to be attributed to the additional information the model was able to extract from the diversity of languages seen during training.

• **Use phoneme over character as an intermediate representation has better performance.** Performances of the phoneme based architectures are almost always better than those of their character based counterparts in all our experimental setups. The gap is bigger for models trained on the multilingual dataset than for those trained on monolingual ones. The only models that are not improved are the ones trained and tested on the same languages. These results show the use of phonemes as an intermediate representation enables knowledge transfer between languages better than a character representation.

• **Training on multilingual data and using a phoneme intermediate representation yields the best results in all considered cases.** Training the acoustic model on multilingual data and the use of an universal phoneme set are relevant ways for improving the generalization capacity of the considered lyrics-to-audio alignment architecture even in zero-resource scenarios.

Evaluations on the Deezer and code-switching datasets are respectively displayed in Figure 6.5 and Figure 6.6. The drop of performance, in comparison to the one obtained previously, can be explained as these evaluation datasets being much more noisy than

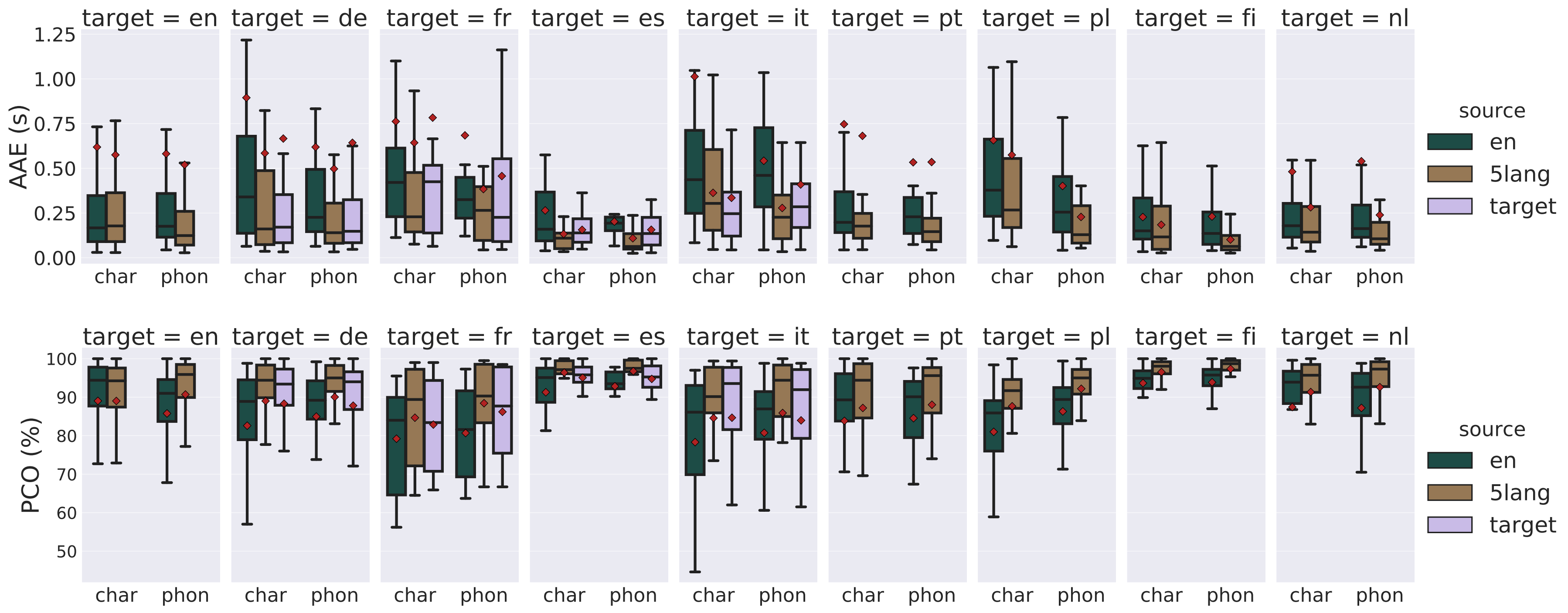


Figure 6.4 – Lyrics-to-audio evaluation on the DALI language subset datasets for phoneme and character based architectures. Several training set design strategies are considered. Languages are given by their ISO 639.1 code. Here 'source' refers to the language dataset used to train the given model and 'target' refers to the language dataset used to evaluate the trained model. When source is equal to target, architectures are trained and tested on the same language. AAE is better if smaller, PCO is better if larger. Mean values are displayed using red squares.

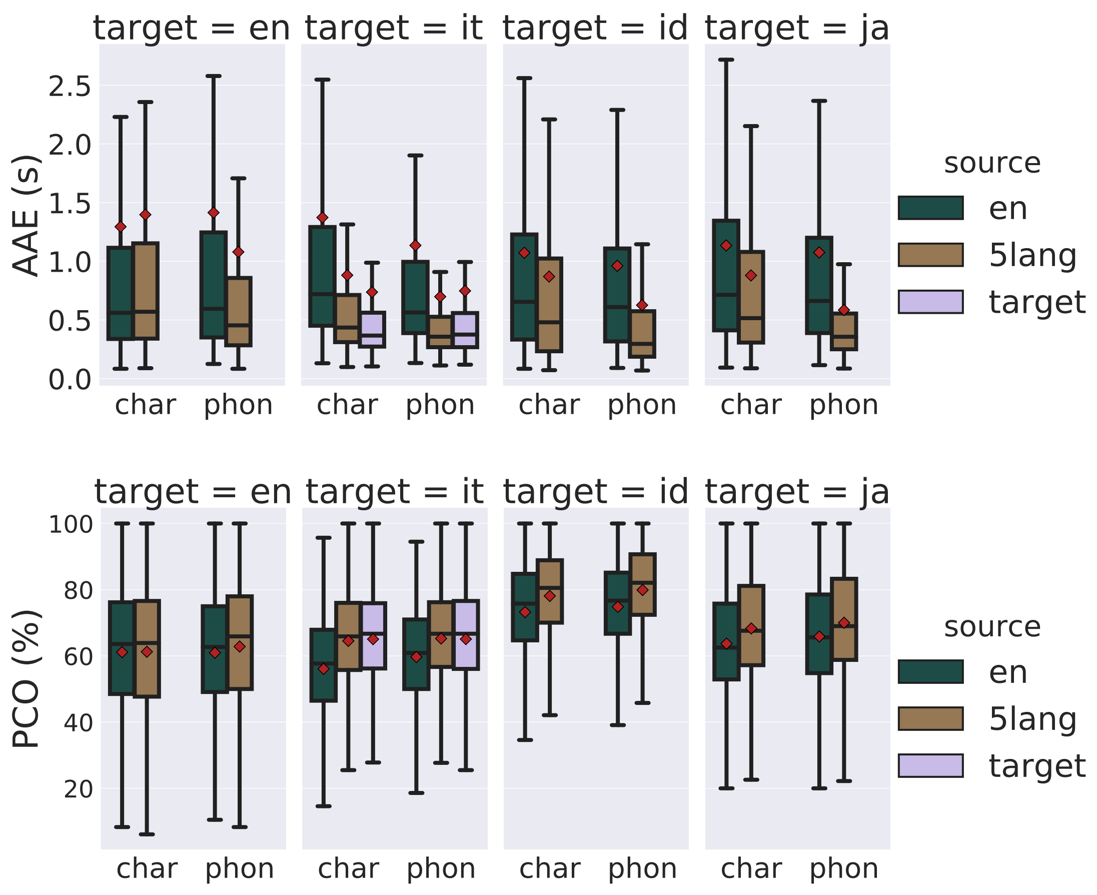


Figure 6.5 – Lyrics-to-audio evaluation on various Deezer datasets for phoneme and character based architectures. Several training set design strategies are considered. Languages are given by their ISO 639.1 code. AAE is better if smaller, PCO is better if larger. When source is equal to target, architectures are trained and tested on the same language. Mean values are displayed using red squares.

the DALI language subset datasets, as described in Section 2.4.2. This thus reflects the poorer data quality of these datasets in comparison to the ones available in the DALI dataset. The same conclusions can be drawn on these bigger, more realistic and noisier datasets, confirming our statements. Notably, in the code-switching case, the increase of performance for the systems trained on the multilingual dataset over those trained on a monolingual dataset, for both phoneme and character based architectures, is particularly important with around 10% of relative increase on the PCO metric. Interestingly, efficiency of the character architectures on the Japanese language does not drop completely over the phoneme architecture one, indicating the romanization of Japanese lyrics being representative of what is actually pronounced. In future work, it could be interesting to extend evaluation of our systems to other non-Latin scripts (*e.g.* the Cyrillic script or Arabic script).

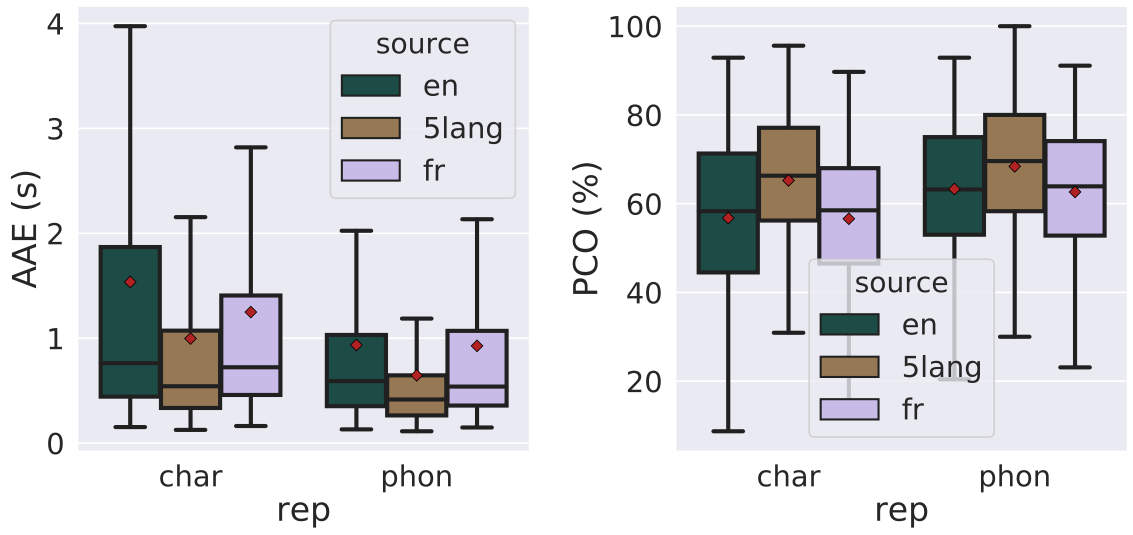


Figure 6.6 – Lyrics-to-audio evaluation on the code-switching dataset for phoneme and character based architectures. Several training set design strategies are considered. Languages are given by their ISO 639.1 code. AAE is better if smaller, PCO is better if larger. Mean values are displayed using red squares.

6.5 Taking phoneme similarities into account

In this section, various experiments are presented where the phoneme similarity is leveraged during alignment. Using this information could allow better knowledge transfer between phonemes and improve the lyrics-to-audio alignment systems performance. Moreover, this information could help dealing with phonemes that cannot be predicted by acoustic models during evaluation. These phonemes are generally phonemes with few to zero occurrences in the training dataset.

To address this issue, a tiny amount of uniformly distributed noise was introduced, in earlier sections, to the posteriorgrams produced by the acoustic model before alignment. Using the phoneme similarity to address this problem could be a better approach than this method. In this scenario, if a phoneme unseen at training is present in the audio, the acoustic model may nonetheless gives it a significant score, providing it identifies comparable phonemes. This section compares different techniques for phoneme similarity computation and assesses how they impact the lyrics-to-audio alignment performance. Evaluation is still performed on the DALI language subset datasets. The architecture used to perform alignment is the one presenting the best multilingual generalization performance in the previous sections, *i.e.*, the one trained on a multilingual dataset and outputting sequences of IPA phonemes.

For some zero resource language subset datasets, a few phonemes are unseen in the training languages. The languages where these phonemes exist are Polish, Dutch and Portuguese. The number of unseen phonemes is respectively three (one vowel, two consonants), two (one vowel, one consonant) and one (one vowel). All of these phonemes are frequent in these languages and then could be a source of misalignment. Unfortunately, the first experiments presented in this section did not show any improvement on the lyrics-to-audio alignment task. No more study was conducted due to a lack of time. Thus, preliminary results are presented here. Still, we believe this field of research shows potential and that our approach may be improved.

6.5.1 Phoneme similarity computation

General presentation: Various methods to compute phoneme similarity are compared in these experiments. For each approach, a similarity matrix $\mathbf{A} \in [0,1]^{|\mathcal{C}| \times |\mathcal{C}|}$, is obtained where $\mathcal{C}$ is the set of symbols supported by our acoustic model, here phonemes and special labels. Each coefficient $a_{i,j}$ of this matrix provides the value of the similarity between the i^{th} symbol of $\mathcal{C}$ and the j^{th} symbol of $\mathcal{C}$. We consider the special symbols (*i.e.* whitespace, instrumental and blank) are only similar to themselves. In other words, the similarity of one special symbol to another symbol is always equal to zero, except with itself, where it is equal to one.

As described in Section 6.2, the CTC-based alignment function takes as input a posteriogram $\mathbf{R}$ and its corresponding lyrics y. The posteriogram is generated by the acoustic model of the system that takes as input the considered song. Before alignment, the posteriorgram is simply multiplied by the matrix similarity to account for the computed phoneme similarity. Some approaches given in the next section do not consider similarity for all phonemes (*e.g.* just considering vowels), resulting in a similarity matrix with some coefficients equal to zero. As a result, the matrix generated after multiplying the posteriorgram by the similarity matrix might still contain phonemes present in lyrics $\boldsymbol{y}$ with a constant null probability. To avoid such situations, a tiny amount of uniformly distributed noise is still added to all the elements of the resulting matrix.

Method description: The first method studied to compute phoneme similarity is based on phonological features as described in Section 6.1.2 and Section 3.4.2. With singing voices made up primarily of vowels [Mes13], it may be assumed that they are primarily utilized to produce alignment. Coming from this assumption, this approach solely considers vowel similarity. As described in Section 3.4.2, all IPA vowels can be simply described by a few features. These definitions are gathered from IPA phoneme chart features[6]. The features collected include 'High', 'Low', 'Front', 'Back', 'Round' and 'Tense'. For each vowel, a feature vector F of size six is constructed, with $f_i = 1$ when the feature is present and $f_i = 0$ otherwise. The similarity $s_{i,j}$ between two vowels v_i and v_j is then computed such as:

$$\begin{aligned} s_{i,j} &= 1, \text{ if } v_i = v_j \\ s_{i,j} &= 0.5, \text{ if } h(\mathrm{F}_1, \mathrm{F}_2) = 1 \\ s_{i,j} &= 0, \text{ otherwise} \end{aligned} \tag{6.9}$$

where, F_1 and F_2 are respectively the feature vectors of vowels v_i and v_j and the function h is the Hamming distance. Consonants, for this method, are only similar to themselves.

6. Available at http://www.artoflanguageinvention.com/papers/features.pdf

The second method to compute phoneme similarity is based on the use of IPA phoneme embeddings. In this case, the similarity of two phonemes is simply calculated from the cosine similarity of their respective embeddings. The embeddings from [PDvG17] are used as they are generated using a large multilingual dataset which could be an advantage for our multilingual system. Coefficients of the similarity matrix are then given by:

$$\begin{aligned} a_{i,j} &= \cos(\mathrm{E}_1, \mathrm{E}_2), \text{if } \cos(\mathrm{E}_1, \mathrm{E}_2) > 0 \\ a_{i,j} &= 0, \text{ otherwise} \end{aligned} \tag{6.10}$$

where E_1 and E_2 are respectively the embeddings of the phonemes p_1 and p_2.

Taking into account the phoneme similarity of all phonemes, in the similarity matrix **A**, could add too much noise into the posteriorgram and decrease the alignment performance. Only the most comparable phonemes from the matrix generated using the prior embedding-based approach are thus preserved. The only similarity coefficients kept for a particular phoneme are the one computed with itself and the one with the greatest value corresponding to the most similar phoneme. The latter is only conserved if its value is greater than 0.5. If this coefficient is not rejected, it is allocated 0.5. According to the resulting matrix, the vast majority of phonemes and all unseen phonemes in the training languages have one nearest phoneme assigned.

Multiple systems are examined to see how the presented methods for computing phoneme similarity affect the outcomes of lyrics-to-audio alignment. All approaches use the same acoustic model and account for phoneme similarity during alignment in the same way. As a result, the sole difference between them is how the phoneme similarity is calculated. The phoneme system computation method used for each system is:

- B: No phoneme similarity information is taken into account, it is the baseline system
- $M1$: A fixed value of 0.5 is assigned to the closest vowels determined by their feature vectors as displayed in Equation (6.9)
- $M2a$: Phoneme similarity is computed using phoneme embeddings on all phonemes as displayed in Equation (6.10)
- $M2b$: A fixed value of 0.5 is assigned to the closest vowels determined by their embeddings

Results are given in Figure 6.7. No approach seems to improve performances over the baseline. The performance of the system $M2a$, in particular, deteriorated on nearly all evaluation datasets, suggesting that taking the similarity of all phonemes into consideration during alignment adds too much noise to the posteriorgram. In comparison, the $M1$ and $M2b$ systems just consider a fixed value for the similarity of the most comparable phonemes and get superior performances. Nonetheless, the performance of $M1$ is slightly worse than the baseline for the majority of languages.

However, the results of the $M2b$ approach appear to be on par with those of the B approach on all evaluation datasets. This might imply that using phoneme embeddings is more effective to find closest phonemes to inform lyrics-to-audio alignment than feature vectors. It can be also assumed that, contrary to the assumption made in the first method, consonants are also important to perform alignment. Yet, with the exception of the Dutch language evaluation dataset, the $M2b$ approach does not significantly improve the

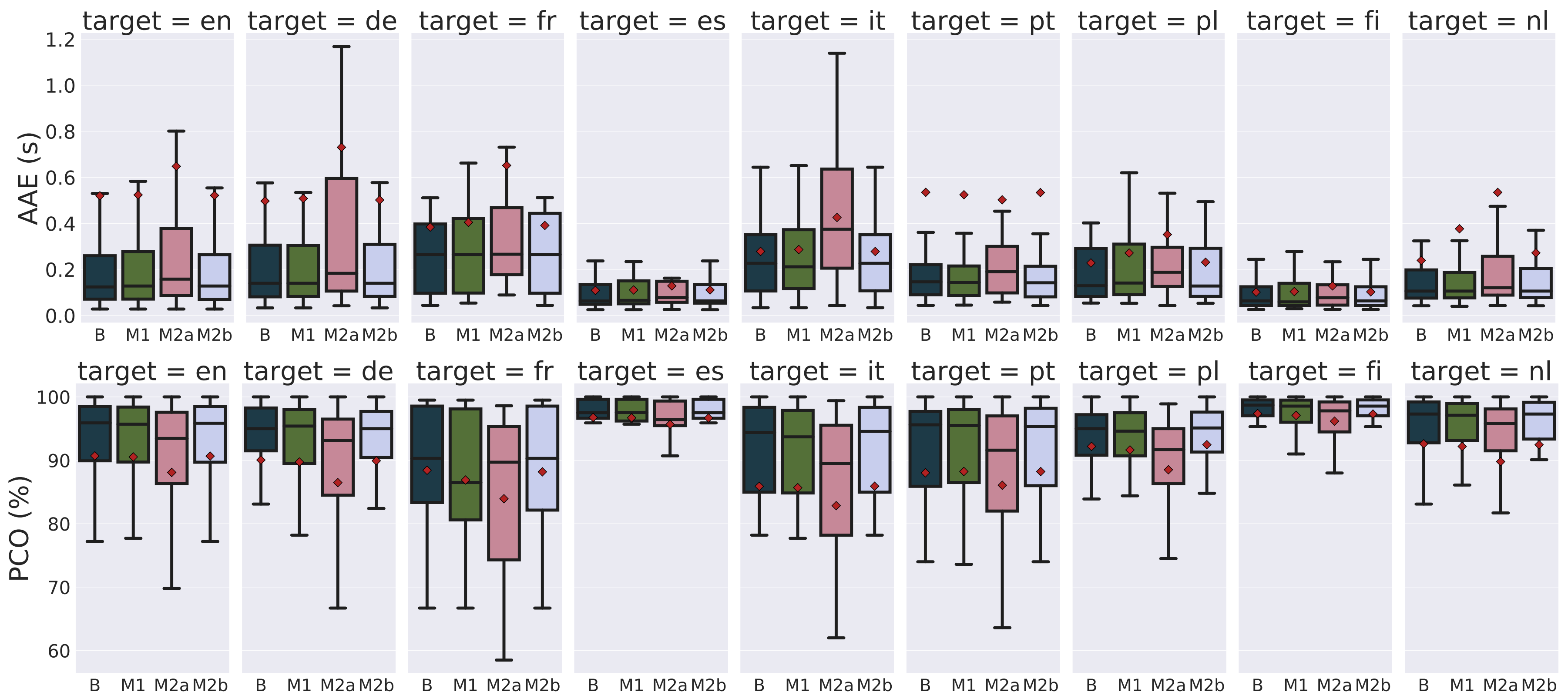


Figure 6.7 – Lyrics-to-audio alignment evaluation on various DALI language subset datasets. All systems are based on the same acoustic model and take phoneme similarity into account during alignment the same way. For each system, a specific method to compute phoneme similarity is used. Languages are given by their ISO 639.1 code. AAE is better if smaller, PCO is better if larger. Mean values are displayed using red squares.

Language	Phoneme	B	M1	M2a	M2b
Dutch	[ʋ]	**93.5 (0.70)**	92.6 (0.73)	90.9 (0.81)	**93.6 (0.70)**
	[ɵ]	**94.9 (1.29)**	**95.3 (1.25)**	94.2 (1.32)	**95.9 (1.14)**
Polish	[ɕ]	92.0 (0.98)	92.2 (0.96)	91.7 (0.97)	**93.4 (0.88)**
	[ɨ]	**91.8 (0.62)**	**92.0 (0.61)**	88.6 (0.72)	**92.0 (0.60)**
	[ʑ]	**91.1 (1.43)**	89.6 (1.52)	88.3 (1.60)	**90.1 (1.50)**
Portuguese	[ɨ]	**89.4 (0.80)**	**89.4 (0.80)**	87.8 (0.87)	**89.6 (0.80)**

Table 6.3 – PCO metrics (%) of unseen phonemes in training data for various DALI language subset evaluation datasets. All systems are based on the same acoustic model and take phoneme similarity into account during alignment the same way. For each system, a specific method to compute phoneme similarity is used. The standard errors are given in parentheses. The best results for each unseen phoneme are displayed in bold.

scores over the baseline, making it difficult to infer that taking phoneme similarity into consideration during alignment has a beneficial influence in this situation.

6.5.2 Studying unseen phonemes

As stated at the outset of this section, utilizing phoneme similarity might enhance the way alignment systems handle phonemes that the acoustic model cannot predict during evaluation. This set notably includes phonemes that are unseen in the training datasets. To quantify this effect, we introduce an adaptation of the PCO metric for phonemes. For a given song and a particular phoneme, this measure restricts the computation of the sum in the PCO, defined in Equation (6.2), to timestamps of words that contain the phoneme. This measure is calculated for all phonemes in the language phoneme set of each DALI language subset dataset evaluated. The performance of unseen phonemes and other phonemes may then be notably compared for a particular dataset.

Looking at the results for the case where no phoneme similarity is considered, the accuracy of unseen phonemes is already fairly good and comparable to that of other phonemes. For the Polish dataset, for example, [ɨ] achieved a value of 91.8. The performance of unseen phonemes are depicted in Table 6.3. Comparing the system *M2b* and the baseline approach *B*, the performance is equivalent for five out of six missing phonemes. The system *M2b* is only significantly better than the *B* approach for one phoneme. For other systems, the metrics of unseen phonemes are almost always on par or degraded with respect to those of the baseline system.

It is thus difficult to establish that taking phoneme similarity into consideration during alignment has any effect on these systems. These findings suggest that knowledge of other phonemes is sufficient in zero resource languages to manage the unseen phonemes during the alignment. Nonetheless, the number of unseen phonemes is very modest for the test zero resource languages, with only a maximum of three unseen phonemes per language.

6.6 Conclusion

In this chapter, we investigated extending state-of-the-art lyrics-to-audio alignment methods to the multilingual context. We selected an appropriate architecture for generalization and we demonstrated design choices regarding the training dataset and the acoustic representation space are salient factors. We have shown that using many languages to train the acoustic model and an universal phoneme set improves the multilingual generalization of such architectures. Finally, we tried taking into account phoneme similarity during alignment. Various methods were implemented to compute this phoneme similarity without significantly improving the results. The system giving the best results was further deployed in production at Deezer under the form of an API allowing to obtain automatic alignments for any desired track of the catalog.

Future works that we could investigate are numerous. In this study, we have built a training dataset using the language distribution found in DALI, which resulted in a largely unbalanced dataset. For comparison, we also conducted experiments with a balanced dataset, in which all five languages were equally present. The results were similar, except for English, where performance was significantly degraded. This raises the issue of how to design training sets in a setting where several high-resource languages are available. Although there are no publicly available datasets exhibiting such characteristics, this case should be investigated in future research. Existing works on multilingual speech processing [WHH17] point towards increasing model complexity to circumvent this.

No attempt to take into account phoneme similarity during alignment were shown to improve the performance. An extensive evaluation of the implemented systems, for zero resource languages where a large part of phonemes are unseen phonemes, is yet to be done. Other methods to compute phoneme similarity could also be employed, including using a more complex feature system like in [LCC+18]. Finally, alternative approaches for accounting for phoneme similarity information during alignment might be investigated as in [RÁA14].

The systematic evaluation of language similarity was out of the scope of this chapter and must be further researched. This work could be performed by quantifying the phoneme-overlap between source and target languages and its influence on the results. This could not be done using the DALI subset datasets, as the evaluation set of each language is different. For this aim, a dataset containing the same recordings in multiple languages might be produced (*e.g.* using songs from cartoons in different languages).

Chapter 7

User-centered evaluation of lyrics-to-audio alignment

As described in the previous chapter, great results of state-of-the-art lyrics-to-audio alignment systems have opened the door to practical applications. It could be notably used as a building block in a karaoke framework, providing automatically aligned lyrics for any song. Karaoke is an important application for lyrics-to-audio alignment at the frontier of music perception and human machine interaction. However, the ecological validity of metrics used to quantify the performance of these systems is yet to be studied. The ecological validation here refers to the judgment of whether these metrics are relevant in a 'real-world' context [Bor80]. The amount of correlation between these metrics and human perception hence remains an open question.

Among the metrics developed for the *Music Information Retrieval Evaluation eXchange* (MIREX) challenge to evaluate lyrics-to-audio alignment, the most commonly used is the *Percentage of Correct Onsets* (PCO) described in Section 6.1.1. A tolerance window for errors was fixed at 0.3s for the MIREX competition, albeit no psychology experiment was conducted to confer validity to this threshold. Additionally, while spectacular progress has been made in the past years, the gap between state-of-the-art systems, as measured in the MIREX challenge, tends to narrow, with many systems achieving close to perfect PCO scores on the test sets.

Therefore, it might now be important to make room for qualitative rather than quantitative metrics. In this chapter, following an interdisciplinary approach, fueled by psychology and musicology insights, we consider the lyrics-to-audio alignment evaluation from a karaoke user-centered perspective. More precisely, we challenge the PCO metric, focusing on how humans perceive lyrics-to-audio asynchrony to derive stricter metrics for the task.

To this aim, after describing the relative literature in Section 7.1, we expose the design of two perceptual experiments in Section 7.2 and their respective results in Section 7.3. Results are further discussed in Section 7.4 and the conclusion is exposed in Section 7.5. A large part of this chapter is directly adapted from the paper: "Ninon Lize Masclef, Andrea Vaglio, Manuel Moussallam. User-Centered Evaluation of Lyrics-to-Audio Alignment. *In Proc. of Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2021".

7.1 Required background

7.1.1 Introducing human-grounded MIR

Nowadays, the machine learning community is raising the question of how to design explainable [AB18] and human-grounded algorithms [PRWZ02]. In the field of *Music Information Retrieval* (MIR), especially, user studies and evaluation metrics play a pivotal role in this shift towards Human-centered MIR. Subjective listening tests [YJMTTS07, PCKSH13, PCKSH13, HW16] and ethnomusicological studies [HB19] previously demonstrated the feasibility to include tasks in the real setting context of user experience. Regarding metrics, we are witnessing the transition from exclusively system-centered evaluation to user-aware evaluation. In the reference toolkit mir_eval [RMH+14b], the lack of human-centered metrics was justified by the complexity and cost required to develop robust subjective evaluation methods [RMH+14a].

Yet there are a few limitations of system-based evaluation, such as their inability to capture the inherently subjective experience of MIR and the absence of necessary correlation between system-centered evaluation and users' perceptions [HK12]. One could, and indeed should, ask oneself what is the meaning of the effectiveness of an algorithm without the presence of an embodied experience of human perception? In epistemic terms, how is the distance to the ground truth translated into an error measurement without the mediation of an individual? Since the advent in 2005 of the system-centered evaluation approach in the MIREX challenge, there were several attempts in creating perceptually grounded metrics, notably among the fields of music transcription [DED08, YLBP20a, YLBP20b], source separation [Vin12] and audio similarity [HK12].

7.1.2 Perception of lyrics and audio synchrony

Singing karaoke engages coordination of articulatory movements, music and language processing systems, as well as crossmodal integration of audio and visual stimuli. It is thus a rich context of perception involving complex stimuli. As a consequence, we briefly consider the research on all the domains outlined above to illustrate paradigms and hypotheses relevant to lyrics-to-audio alignment perception.

When presented with a pair of audiovisual stimuli, individuals reported an asymmetric perception of asynchrony, with audio lagging preferred over visual lagging [vEKJvdP08, VFMA18]. This asymmetry has been correlated with faster transmission of the visual signal over the audio signal [vEKJvdP08] or with the auditory dominance in temporal processing [RP02]. The latter hypothesis asserts that, when emitting a judgment of synchrony, audio would provide individuals a more accurate sensory information in the case of dynamic events like music, and also a more stable internal representation of periodicity, contrary to the visual modality [RP02].

The listening experience is a continuous production of rhythmic expectancies [JB89]. In the case of sensorimotor synchronization experiments, one effect induced by rhythmic expectancies is the anticipation of the stimuli in a sequence, also called the *Negative Mean Asynchrony* (NMA). First reported by Dunlap [Dun10], it states the reaction to an audio stimulus tends to precede rather than follow the stimulus. Repp [Rep06] found out individuals anticipate audio events up to 100ms ahead of time. Klemmer [Kle57] revealed that the anticipation effect varies with the tempo of the rhythmic stimuli, usually measured in terms of the *InterOnset Interval* (IOI) duration. He found the reaction time of individuals when attempting to stay in phase with an isochronous stimulus, is a function of the IOI between stimuli. The reaction time was greater for shorter IOI, suggesting individuals have less sensibility in slow tempos. These observations were further formalized as a function of local and global rhythmic context by McAuley [MS07].

Besides global rhythmic factors, the listening experience is punctuated by local variations. Metric events are periodic peaks of attention organized into nested hierarchies coordinating attention to events on various time-scales, allowing for grouping and accentuation of notes [LJ96]. Musical stresses are the cues to infer a general rhythmic pattern [LJ96]. Among the significant factors of stress were reported the duration of syllables [LJ96], loudness [SvHP97], alignment with beats [JB89] and sequence boundaries [JB89, KNH$^+$05].

7.2 Method

Given previous studies, our theoretical hypothesis, about the perception of lyrics-to-audio alignment, is formulated in two points. Firstly, we expect individuals to tolerate more audio lagging than lyrics lagging. Secondly, we expect the perception of lyrics-to-audio synchrony to rely both on global and local rhythmic context. To investigate these hypotheses, we designed two psychological experiments inspired from the main application of this task, karaoke. We chose karaoke as it is a popular practice where the participants' rhythmical precision is important, which requires that they remain attentive to displayed lyrics as much as to the audio. The first experiment is designed to test the influence of global parameters on human perception of audio/displayed lyrics synchrony and to investigate its symmetrical properties. The second one intends to explore local factors' influence.

To run both experiments we developed a karaoke application prototype, whose displayed textual lyrics were intentionally misaligned with the background audio according to various, controlled conditions, thereby creating an audiovisual offset. The stimuli were presented to individuals who then annotated their perceived quality of alignment in different error scenarios. Participants were also asked to sing along the stimuli. The welcome page of the interface is displayed in Figure 7.1 and one example of a karaoke excerpt is given in Figure 7.2. Both experiments were run online, through a web interface called Dalida that was designed to be correctly displayed on both computer and phone screens, for a total duration of two weeks each, between January and April 2021. The first experiment was only opened to Deezer employees whereas the second one was publicly available and therefore permitted to collect data from a larger and more diverse pool of people.

Before engaging in karaoke, participants are asked to fill out a questionnaire allowing us to determine their level of musical expertise and familiarity with the practice of karaoke. We collect, with their consent, a range of information about their age, declared gender and native language. We do not have control on their external environment when performing karaoke (external noise) and any other factor that could disturb the readability of the

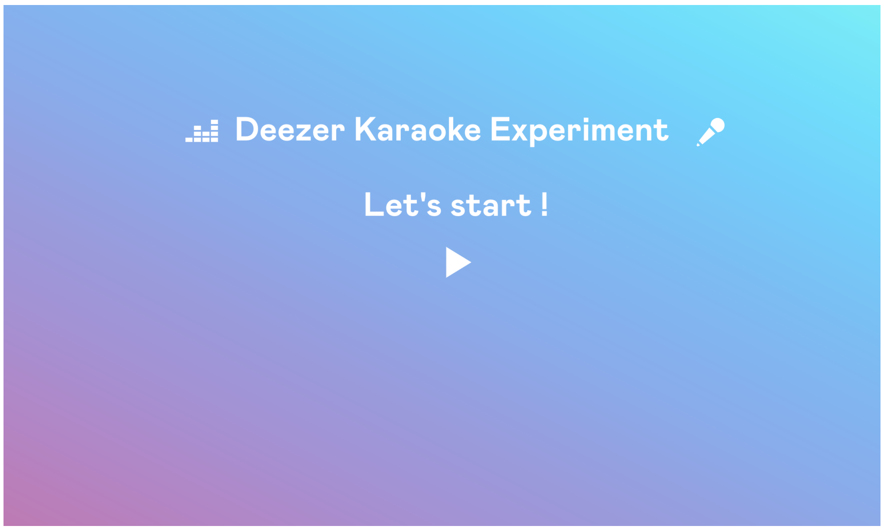


Figure 7.1 – Welcome page of our Dalida application.

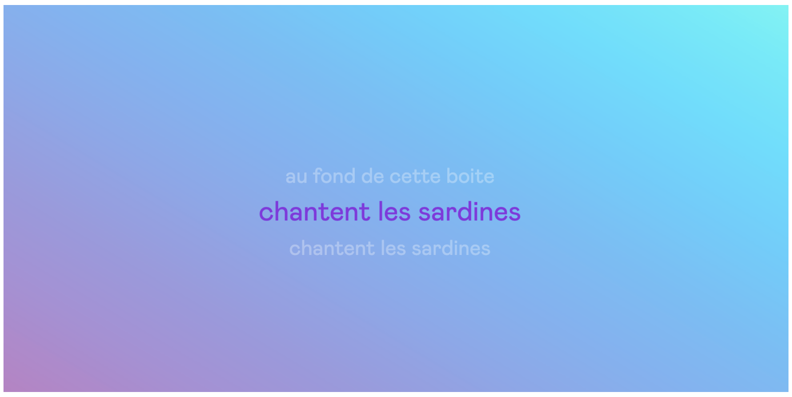


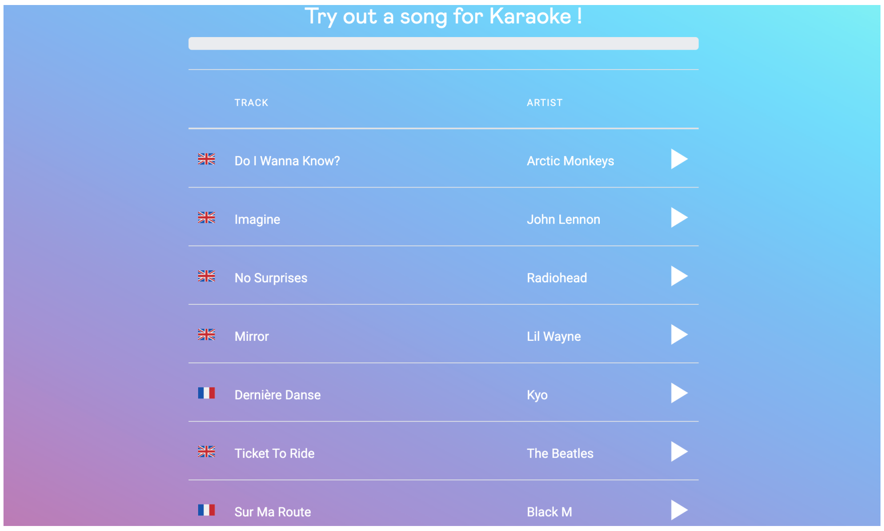
Figure 7.2 – One karaoke excerpt to sing with displayed aligned lyrics, with or without alignment errors.

karaoke (low light, uncorrected vision problem). Nevertheless, the instructions of the experiment encourage them to put on headphones and favor a quiet environment.

In both experiments the dependent variable measured is the perceived synchrony and the amount of offset between lyrics and audio is a within subjects factor. In order to prevent an order effect, the values of audiovisual offset are presented in random order. These two experiments are akin to the Simultaneity Judgment task (SJ) widely used in the literature for studying the synchrony perception of audiovisual stimuli [vEKJvdP08, VFMA18].

7.2.1 Dataset

Since the measured effects should be valid irrespective of the song, we allow participants to choose their song for karaoke within a set of 80 songs from various genres (pop, rock, rap and metal) and language (English, French, German). A snippet of the page for selecting a song is displayed in Figure 7.3. We selected popular songs in the DALI dataset[MBCHP20] with alignment done at word level. The first criterion for the choice of songs was their



Try out a song for Karaoke !

	TRACK	ARTIST
🇬🇧	Do I Wanna Know?	Arctic Monkeys
🇬🇧	Imagine	John Lennon
🇬🇧	No Surprises	Radiohead
🇬🇧	Mirror	Lil Wayne
🇫🇷	Dernière Danse	Kyo
🇬🇧	Ticket To Ride	The Beatles
🇫🇷	Sur Ma Route	Black M

Figure 7.3 – Choice of one song for a trial.

popularity, so that we can expect a large proportion of participants to be knowledgeable of their lyrics and melody. Other important points guiding our choice was the correct lyrics-to-audio alignment and the absence of syntactical problems.

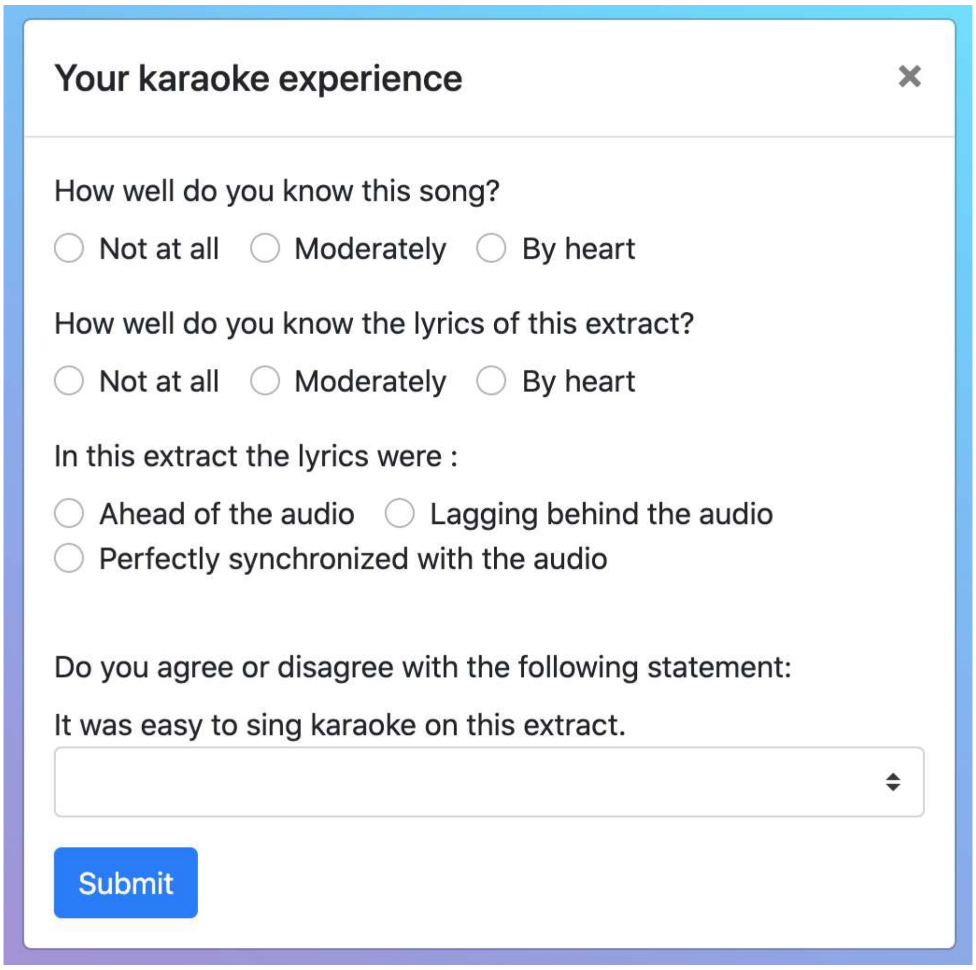
We manually controlled the alignment quality of this subset by visualizing their lyrics in the karaoke prototype and got rid of poorly aligned songs from our selection. To avoid a learning effect of the song, each song can be selected once for a trial and the number of listening during a trial is limited to two times. Moreover, the order of the songs in the selection menu for karaoke is randomized for each trial.

7.2.2 Influence of global factors

Experiment design:

In this experiment, each participant is asked to choose 14 songs from the dataset from which karaoke excerpts are presented. Each audio extract lasts 35 seconds and consists of a sequence of words within lyrical lines, highlighting each word subsequently according to their aligned onset times. A lyrics-to-audio alignment error is generated for each user-song pair randomly from a set of positive and negative offsets between the audio and the lyrics displayed on screen. The offset is fixed for the whole sequence, which means all words in the stimulus are shifted by the same amount. At the end of each trial, participants are asked to report whether they perceive an asynchrony between lyrics and audio with a ternary response ('lyrics ahead', 'lyrics lagging', 'synchronous'). The corresponding questionnaire is displayed in Figure 7.4.

This experiment has a repeated measure design, with lyrics-to-audio synchrony perception as a dependent variable, and the lyrics-to-audio error offset as the independent variable having 14 modalities. It aims to measure an overall threshold of lyrics-to-audio synchrony perception and to study the influence of global rhythmic factors on this threshold, such as the tempo and the word rate. If our theoretical hypothesis is confirmed, we expect to observe a greater proportion of 'synchronous' responses for lyrics ahead than lyrics



Your karaoke experience

How well do you know this song?

○ Not at all ○ Moderately ○ By heart

How well do you know the lyrics of this extract?

○ Not at all ○ Moderately ○ By heart

In this extract the lyrics were :

○ Ahead of the audio ○ Lagging behind the audio
○ Perfectly synchronized with the audio

Do you agree or disagree with the following statement:

It was easy to sing karaoke on this extract.

Figure 7.4 – Questionnaire used to evaluate lyrics-to-audio alignment first experiment.

lagging, as well as a modulation of the perceptual threshold with the global rhythmic context (tempo, word rate).

Choice of offsets:

In order to precisely define a threshold, we use a wide range of 14 offsets from -1s to 1s with negative offsets corresponding to lyrics ahead and reversely positive offsets meaning lyrics lagging behind audio. The list of offsets is: $[-1, -0.5, -0.4, -0.3, -0.2, -0.1]$ for the negative values and $[0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 1]$ for the positive ones. We intentionally keep this number as small as possible, since this value is equal to the number of annotated songs required for each participant. Meanwhile, we wish to highlight effects around the commonly used threshold of 0.3s and -0.3s. Thus, we use smaller steps around these values.

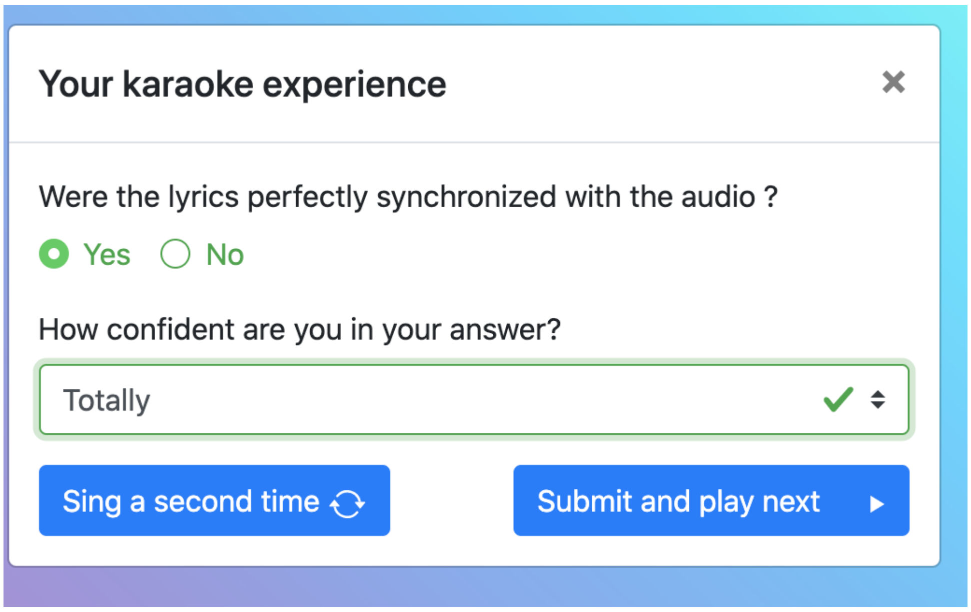
We also included larger offsets (1s, 0.7s, -1s) as control values, to test that individuals systematically report those as asynchronous. The value -0.75s is not included to reduce the amount of offsets. In the same spirit, we expect the offset value 0 to trigger 'synchronous' answers. The full experimental protocol was carefully tested beforehand with user testing sessions on a half-dozen people. Based on these, we evaluated completing the annotation required approximately 12 minutes per participant. Overall, the experiment gathered 53 participants who finished the task.

7.2.3 Influence of local factors

In this second experiment, we made some changes in the karaoke interface. This time, we require each participant to choose one song from the dataset from which ten audio excerpts are presented with different audiovisual offsets. Each sample is composed of three lyrical lines from the given song. The experience can be done multiple times with other songs if wanted. Each song takes around three to five minutes to annotate. Whilst in the first experiment the alignment errors were located on all the words of the sentence, in the second experiment, the position of the error may be located on the first, the last word of the sentence, or close to a beat. These choices are motivated by some of the significant factors of stress described in Section 7.1.2 being alignment with beats [JB89] and sequence boundaries [JB89, KNH+05].

We decided to leave apart for this study the influence of long syllables [LJ96] and the loudness [SvHP97]. In fact, long syllables and loud words are found to be overlapping respectively with the last word of the sentence and words close to beats. The perceived synchrony is reported as a binary response ('yes', 'no') with confidence on a 5-point Likert scale. The corresponding questionnaire is displayed in Figure 7.5. This experiment intends to quantify the interaction of the error location in the sentence and the offset on the perceived alignment. It has a factorial design with the lyrics-to-audio offset and the position of the error as within subject factors. If our theoretical hypothesis is confirmed, we expect to observe a modulation of the perceptual threshold with the location of the error in the sequence.

Proximity of a word to a beat is defined as at a distance less than a *sixteenth note* from the beat, computed as 𝅘𝅥𝅯 $= 15/BPM$. The tempo estimation, here given in *Beats*


Your karaoke experience

Were the lyrics perfectly synchronized with the audio ?

[✓ Yes] No

How confident are you in your answer?

Totally

Figure 7.5 – Questionnaire used to evaluate lyrics-to-audio alignment second experiment.

Per Minute (BPM), relies on Anssi Klapuri's algorithm, which showed 80% accuracy with constant tempo during the *International Society for Music Information Retrieval* (ISMIR) 2004 tempo induction challenge [GKD$^+$06]. The position of beats is also given by this algorithm. Starting from the baseline threshold of synchrony perception established in the previous experiment, the second experiment focuses only on lyrics lagging with 3 offsets (0.25, 0.5, 0.75) and a control sample with no offset.

We chose only positive offsets because of practical constraints. Indeed, applying a negative offset at the word level can (and does frequently) result in overlapping with previous words, at least for beat-aligned and end words. Filtering out cases of overlapping words resulted in an important selection bias toward very slow songs. To avoid it, we could apply linearly decreasing offsets to preceding words until no overlap remains, as a naive way to 'catch-up' with the true annotations.

Such a behavior is consistent with what is observed in errors made by lyrics-to-audio alignment systems, multiple errors on consecutive words being recurrent. The problem is that we would not control which first offsetted word the participant will be confronted to. Overall, we decided to not consider negative offsets altogether in this experiment.

Still, the problem of overlapping words remains for positive offsets. However, after applying linearly decreasing offsets to consecutive words until no overlap occurs, the first offsetted word to which each participant is confronted remains the word of interest. Finally, we collected 2458 annotations from 193 participants.

7.3 Results

As we intend to compute an overall threshold of synchrony perception, we perform the analysis at the level of the aggregated results, considering all annotations from all users.

We remove beforehand all the trials from participants who did not answer correctly to our control levels *i.e.* 'non synchronous' at 1s and 'synchronous' at 0s. Precisely, it represents 11% of answers for the first experiment. We do a similar cleaning phase for users of the second experiment using the control offset of 0 and remove 8% of answers.

7.3.1 Asymmetry of Lyrics-to-Audio Alignment Perception

Using the data collected in the first experiment, we compute an aggregated proportion of respondents who indicate lyrics and audio are 'synchronous', and display it as a function of the lyrics offset in Figure 7.6. We see synchrony perception is typically asymmetric, positive offsets being more easily detected than negative ones. This was expected as it resonates with previous findings [vEKJvdP08, VFMA18].

Beyond aggregated data, we also looked at individuals and found the thresholds were indeed asymmetric for 72% of individuals. The observed synchrony boundaries support the hypothesis that individuals tolerate more lyrics ahead than lyrics lagging. Whether it is due to the faster traveling of light, an anticipation effect or a more reliable rhythmic information provided by the audio while singing karaoke is beyond the scope of this chapter and could be studied in future work.

To give perspective, we plot the window function corresponding to the PCO metric scoring as used in the MIREX challenge, with an absolute threshold value of 0.3s. We also fit a function akin to a scaled skew normal distribution function to the data points. Among several attempts with asymmetrical continuous functions, this is the best fit we obtain, although it does not respect the maximality at 0. Parameters of the fitted function are a skewness factor of 1.12, a location of -0.22 and a scale of 0.29; a multiplicative factor is also applied in order to have a value of 1 at the maximum.

Using this function we can derive new perceptive thresholds for synchrony using a simple rule of 50% of respondents being able to detect the offset. For lyrics ahead and lyrics lagging we respectively identify the offsets -0.33s and 0.22s. Given the amount of noise in the data, we can reduce these to -0.3s and 0.2s , allowing us to make statistical tests on these points, and examine if the differences of perception are significant for these values. Indeed, pairwise tests revealed a significant difference of proportions of response 'synchronous' on the levels -0.3 and 0.3s ($\chi^2(1) = 4.26$, p $=$ .038), while proportions on the levels -0.3s and 0.2 are not statistically different ($\chi^2(1) = 0.04$, p $=$ .08).

7.3.2 Sensitivity to global rhythmic context

In order to assess whether there is an influence of the global rhythmic context on lyrics-to-audio alignment perception, we compare the distribution of 'synchronous' responses at each offset for two rhythmic factors: the tempo and the *Words Per Second* (WPS). We split our dataset of songs into two classes of tempo, defined as the upper and lower quartiles of the distribution of tempo, respectively fast ($\geq$138BPM) and slow ($\leq$93BPM). Although it is correlated with tempo, we also consider the average WPS rate of songs as a meaningful global factor. Again, we look at the first and last quartiles as Low ($\leq$1.16WPS) and respectively High WPS ($\geq$1.2WPS) classes. Figure 7.7 and Figure 7.8

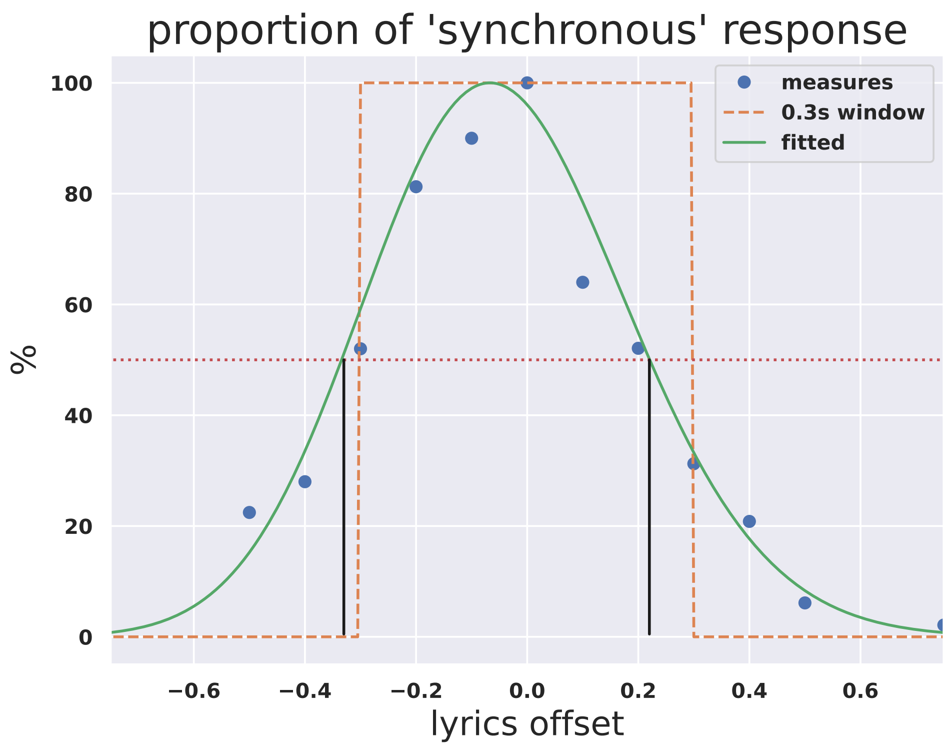


Figure 7.6 – Aggregated results of *synchronous* judgment as a function of the lyrics/audio offset.

show the aggregated reported synchrony profiles for the negative offsets for the derived tempo and WPS classes. On both metrics, we observe no threshold discrepancy between the two classes for positive offsets.

To highlight the differences between classes, we fit a relatively simple sigmoid function to the data points. Among several candidates, a Gauss error function seemed most appropriate. Fitted functions are also displayed in Figure 7.7 and in Figure 7.8 and emphasize the different synchrony slopes. As we did earlier, we particularly consider the offset value intersecting with an average of 50% of 'synchronous' responses as an indicator of participants' sensitivity to temporal asynchronies. Interestingly, the 50% threshold for the perception of synchrony is located at a larger offset (-0.36) for slow tempo than in fast tempo (-0.31). These results show that individuals report more frequently lyrics ahead as synchronous with slow tempo than with fast tempo. The lower sensitivity to lyrics-to-audio alignment errors in slow tempo is consistent with the results of [Kle57]. Significance of these results are tested. The proportions of 'synchronous' responses at the offset -0.3s display a significant difference between songs with high and low tempo ($\chi^2(1) = 5.44$, and $p < .02$).

Analogously, we find out subjects are more tolerant to lyrics ahead (audio lagging) in high word rate than in low word rate. The 50% threshold for the perception of synchrony is, indeed, located at a larger offset for high word rate (-0.39) than in low word rate (-0.28) (Figure 7.8). These results show that the subjects are more tolerant to lyrics ahead (audio lagging) in high word rate than in low word rate. We again test the significance of these results. The proportions of 'synchronous' responses at the offset -0.3s are significantly

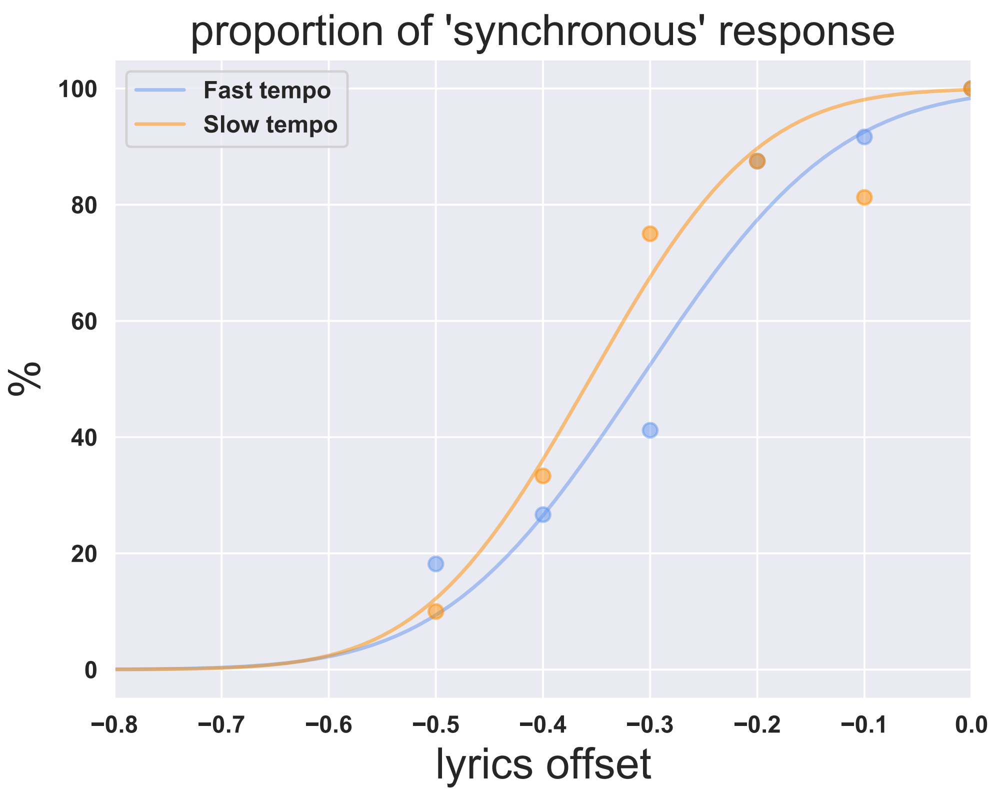


Figure 7.7 – Proportion of response ’synchronous’ by tempo class.

different between songs with high and low WPS ($\chi^2(1) = 16.86$, and $p < .00004$).

7.3.3 Interaction between offset and word position

We design the second experiment to distinguish perception of the asynchrony as a function of the word’s position in the sentence. As explained in Section 7.2.3, we are only able to test for positive offsets. As insights from the previous experiment, we can assume that user sensibility is less affected by global factors for positive offsets. As a result, we expected it to be challenging for local factors too. For this reason, we aimed at collecting a much larger set of annotations, with a reduced set of tested offsets.

Figure 7.9 presents an overview of the results. There is a fairly large amount of noise in the collected data points, and few clear differences between synchrony perceptions for the three classes of word positions. The noise is particularly clear from the displayed level of confidence of participants who were unable to detect the asynchrony even for large values of the offset, but still were quite confident about their choice (average around 3.8). Regarding the location of the alignment error within the sentence, Cochran’s Q test did not indicate a notable difference among the proportions of synchrony responses reported for the three error positions, $\chi^2(2) = 5.77$, $p = .056$.

The only visible effect seems to be for words aligned on *beats*, for which the confidence in the ’asynchronous’ answer at the 0.25 level is markedly higher than for the *end* class. More precisely, a Wilcoxon signed-rank test reveal that comparing errors located on the

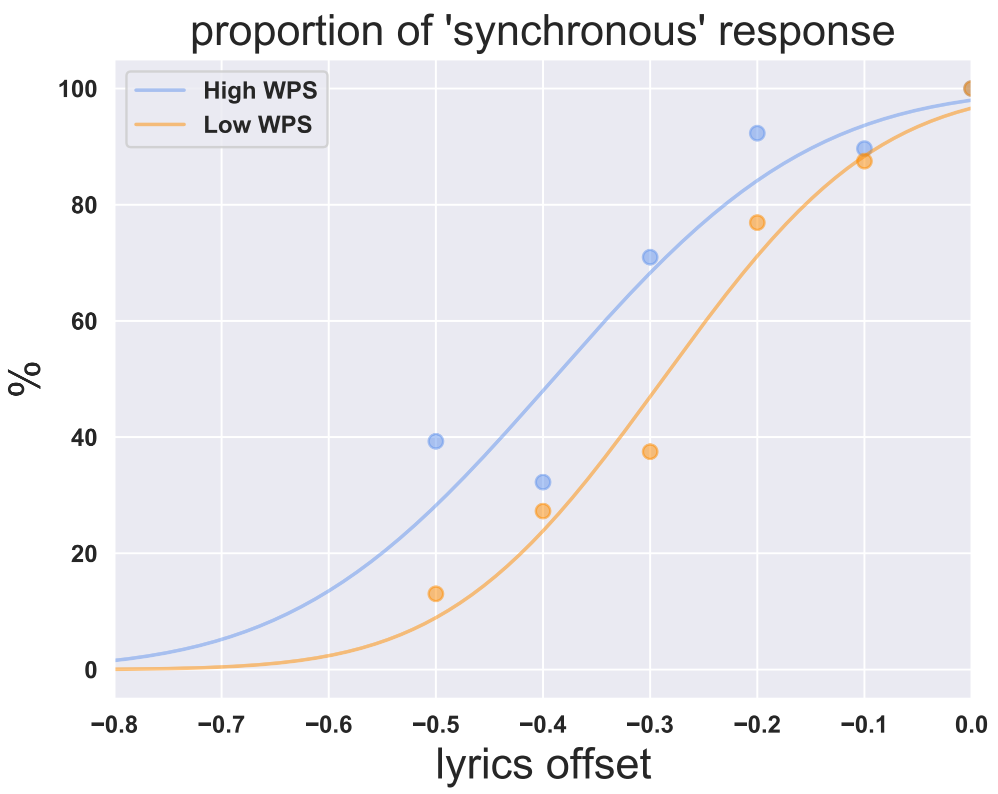


Figure 7.8 – Proportion of response 'synchronous' by WPS class.

beat with those on the last word did elicit a statistically significant change in the reported confidence of perception of error in individuals at the 0.25 level ($Z = 2.756$, $p < 0.006$). Indeed, the mean confidence rating is 4.1 for the errors on the beat and 3.5 for the errors on the last word of the sentence. Such a phenomenon is not observed for the synchronous case.

7.4 Discussion

7.4.1 General discussion

Building on psychological theory and previous studies, we had hypothesized that a perceptual evaluation of lyrics/audio alignment quality would be asymmetrical and dependent on both global and local factors. Using a first experiment we did find strong evidence for asymmetry and, to some extent, for global factor influence. Despite a much larger experimental setup which involved hundreds of participants, we were not able to exhibit a clear influence of the local factors we tested. This negative result could mean that the local factor considered, *i.e.* the word position in the lyrical line, is not the relevant ones.

It is possible words' grammatical or semantic functions are more subject to human attention in a karaoke context. Indeed, the only significant phenomenon we observed was on words located on *beats*, for which the asynchrony perception was more acute. We can assume safely that words with important grammatical or semantic functions are mainly

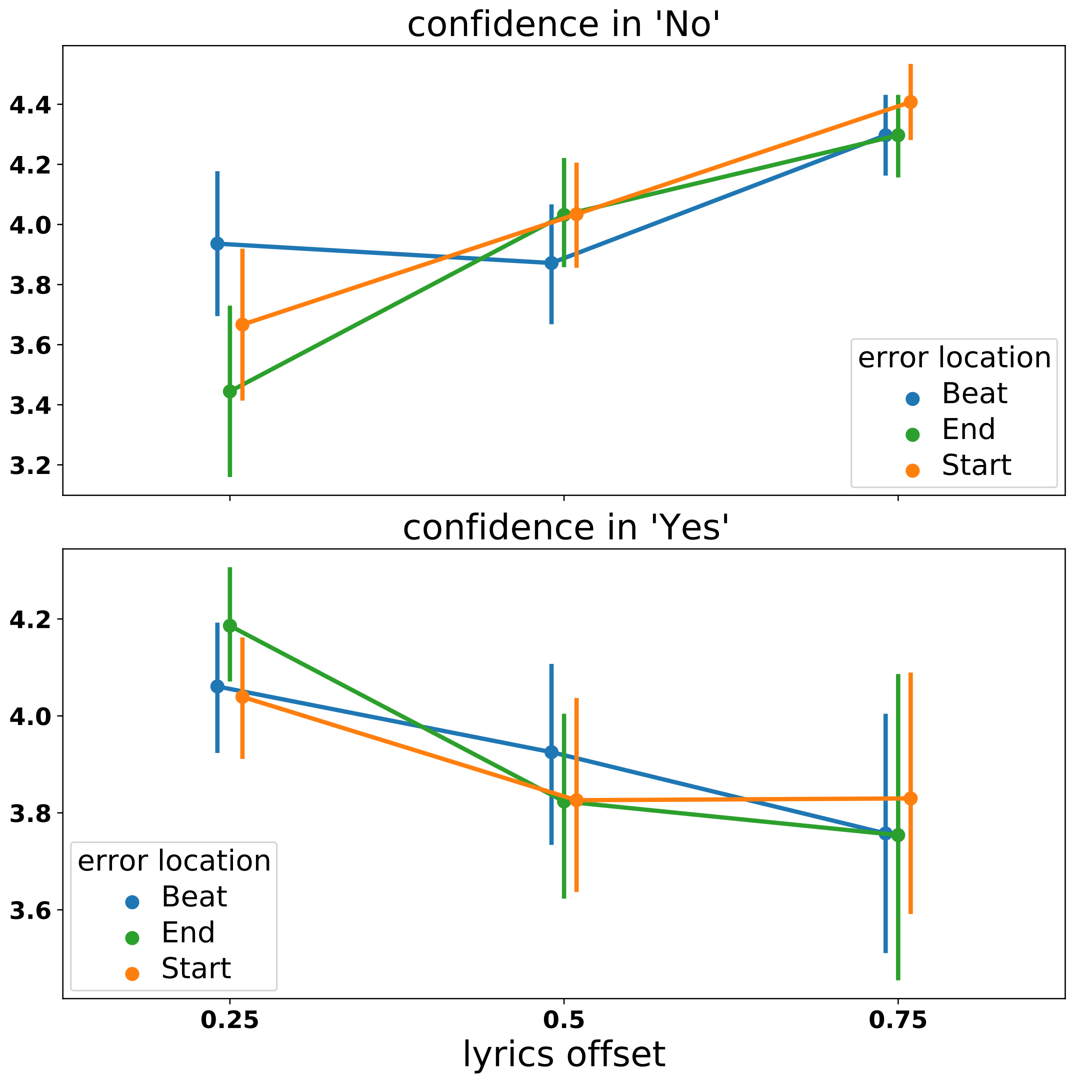


Figure 7.9 – Plot of the reported answer (synchronous is 'yes', asynchronous is 'no') and confidence score (5 level Likert scale) in the perception of synchrony, by location of the alignment error within the sentence.

located on *beats*. Future work should then investigate the relationship between rhythmic position and lyrical function of words and test new hypotheses of perceptual differences.

Finally, although we did our best to build a realistic yet controlled experimental setup, we acknowledge that as a psychological experiment mostly conducted online, we can not completely rule out the possibility the measurement noise was too high to allow us to detect signals on local factors. There are arguably other factors that could influence this perception, notably at the human level. Indeed, familiarity with the song (*e.g.* previous knowledge of the lyrics and/or the music), but also level of musical expertise and even karaoke practice could be important variables to consider. In the conducted experiment, we collected such information from participants. Preliminary studies did not show any interesting results on this data. For a question of time, no further research was pursued. Additional experiments on this data are left for future work.

7.5 Conclusion

In this chapter, we challenged the objective evaluation of the lyrics-to-audio alignment task using hypotheses from psychological theory. We postulated three effects: asymmetry, influence of song features and influence of words' local positions. We were able to demonstrate the first two effects using a large scale online experiment, disguising the synchrony annotation task as a Karaoke experience. This framework proved less efficient for the third effect, despite our efforts to collect up to several thousand annotation points. Future work could investigate more diverse sets of factors, globally or locally, both on musical attributes and user features. Furthermore, the studies described in this chapter could open the way for the development of perceptually grounded versions of lyrics-to-audio metrics.

Chapter 8

Singing language identification

Language identification is the task of detecting the language sung in a given song. Applications are numerous: informing lyrics transcription, supporting regional and genre classification or helping monitoring the language of songs broadcasted in the media (*e.g.* Toubon Law). One may want to estimate the song language from metadata frequently accessible such as song title or artist name. Yet, this method is limited as the metadata language can differ from the song language, and metadata may not contain enough information for retrieving the language [TW04]. The speech community has long tackled language recognition from audio data, notably with the *Language Recognition Evaluation* (LRE) series [MGH+14]. However, the task was scarcely transposed into the music domain and most of the techniques used for spoken *Language IDentification* (LID) have yet to be adapted for *Singing Language IDentification* (SLID). The latter task is more complex, with the *Singing Voice Recognition* (SVR) domain raising challenging issues as detailed in Section 1.2.

Previous works on the SLID task include acoustic-phonetic systems which characterize language-specific acoustic events and their distribution with carefully chosen acoustic features, such as *Mel-Frequency Cepstral Coefficients* (MFCC) [SBWH06], *Stabilized Auditory Images* (SAI) [CSR11] and *Temporal Patterns* (TRAP) [KAD14]. Statistical modeling and supervised classification are then applied to identify the language. In particular, Kruspe's system using the i-vector extraction technique obtains the current best performance on SLID [Kru14b], reaching 78% accuracy on an *a cappella* dataset in three languages.

Another popular category of systems are the phonotactic approaches that try to identify phonemes from the audio and examine their combinations and sequences, which are distinctive from one language to another [LML13]. These approaches are more resource-demanding as acoustic models, used as phoneme recognizers, have to be trained. Mehrabani et al. [MH11] use multiple language-specific acoustic models trained on speech data, to then compute language likelihoods with n-gram language models for each target language. While the performance is on par with Kruspe's i-vector-based approach [Kru14b], it is more complex to train and it is hardly scalable to a large set of languages.

In [Kru16c], the author simplifies the approach by using a unique *Deep Neural Network* (DNN) based English acoustic model and identifies the language from phoneme statistics.

While singing data is included in the acoustic model training, the frame-wise phoneme annotations are obtained by a forced-alignment step, which leads to poorly annotated data. Also, this statistics-based language modeling overlooks the information contained in the phoneme transitions.

In this chapter, we propose a new modernized phonotactic SLID system using most recent DNN techniques: in particular, we use an acoustic model inspired from the one described in Chapter 4. This model is a *Recurrent Neural Network* (RNN) trained on multilingual data with the *Connectionist Temporal Classification* (CTC) algorithm, alleviating the need for frame-level aligned lyrics, and outputting sequences of *International Phonetic Alphabet* (IPA) phonemes. This acoustic model was shown to display great results for the lyrics-to-audio alignment task in a multilingual framework. We thus assume this model could provide information for the language identification task. Moreover, outputting sequences of IPA phonemes makes this acoustic model trainable using data from any language as long as a pronunciation lexicon is available for it.

For language classification, we use a recurrent architecture that can capture temporal information in phoneme estimation sequences. We evaluate our system in a standard closed-set scenario and in a harder setup with out-of-set languages where we can acknowledge the limits of our model. This is the first time out-of-set languages are considered for the SLID task. The question of whether singing language recognition algorithms could handle these cases remains to be answered.

The chapter is organized in various sections. Firstly, we introduce some background for the task in Section 8.1. Then, in Section 8.2, we describe the key aspects of our deep phonotactic SLID approach. The dataset, baselines and implementation details are given in Section 8.3. Results are presented in Section 8.4, providing a first reproducible benchmark on the DALI dataset. The dataset split used is made publicly available[1]. A large part of this chapter is directly adapted from the paper: "Lenny Renault, Andrea Vaglio, Romain Hennequin. Singing Language Identification Using a Deep Phonotactic Approach. *In Proc. of Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2021".

8.1 Required background

8.1.1 Speech Language Identification

Humans use multiple cues to efficiently identify languages they are familiar with. They have been described extensively in various perceptual studies [MJC94, LML13] and can be divided into four levels of abstraction:

- Acoustic-phonetic: native speakers can generate acoustic events (or phones) peculiar to their language through speaking. The number of phones available and their distribution vary by language (for example, the [th] sound exists in English but not in Mandarin).

1. `https://github.com/deezer/SingingLanguageIdentification`

- Phonotactic: each phone may be tokenized as a phoneme (a linguistic unit) and each language has lexico-phonological norms ruling their possible combinations and patterns. As a result, phoneme sequence distributions sequences vary between languages.
- Prosodic: often known as the 'speech melody', the prosodic descriptors define how phones are uttered, *e.g.* their length, rhythm, intonation, and stress.
- Vocabulary and grammatical: specific words can be enough to guess a language following the phonological and syntactic norms of this language.

To identify a familiar language, humans employ these cues simultaneously. Still, the most salient are the acoustic-phonetic and phonotactic [LML13]. The majority of LID systems are thus based on one of these two cues. As defined in the introduction of this chapter, they are called acoustic-phonetic and phonotactic approaches.

8.1.2 I-vector extraction

The i-vector extraction technique is an unsupervised machine-learning method created for speaker verification in [DKD+11] and subsequently successfully applied to the SLID task [Kru14b]. It is generally used as a feature post-processing step. The goal is two-fold. Firstly, it helps discard the most ordinary irrelevant information embedded in the training examples. Secondly, by compressing the relevant data into a fixed-size vector, it helps diminish the dimensionality of the training features.

To do so, it notably uses a *Universal Background Model* (UBM) to summarize training data. The most frequent choice for this model is a *Gaussian Mixture Model* (GMM) of C Gaussian components. An utterance $\mathbf{U} = [\mathrm{U}_0, ..., \mathrm{U}_{T-1}]$ is mapped, in the GMM-UBM paradigm, to its GMM-supervector $S(\mathbf{U})$ of size CF. Here, T is the number of frames, F the number of features and U_t the t^{th} frame-level feature vector of dimension F.

The GMM-supervector $S(\mathbf{U})$ is computed using the trained UBM. This supervector can be decomposed like:

$$S(\mathbf{U}) = \mathrm{M} + \mathbf{V}\mathrm{W} \tag{8.1}$$

here, M is a language-session-independent supervector, also computed from the trained UBM, and $\mathbf{V}$ a $CF \times R$ low-rank total variability matrix. This matrix models the significant variability in the supervector space. Finally, W is the desired i-vector, a vector of dimension R, from a normal distribution. The size of the vector is a tunable parameter of the algorithm. Before being able to do i-vector extraction, both GMM-UBM and i-vector extractor $\mathbf{V}$ are trained one after the other using an expectation maximization algorithm on the complete training set. Then, for a desired utterance $\mathbf{U}$, the i-vector W can be finally computed using the extractor $\mathbf{V}$. Exhaustive description of the complete algorithm is given in [GBM+11].

8.1.3 Metrics

Classic binary classification metrics can be generalized to multi-class labeling. In this chapter, we consider two of these metrics. Firstly, the multi-class balanced accuracy is

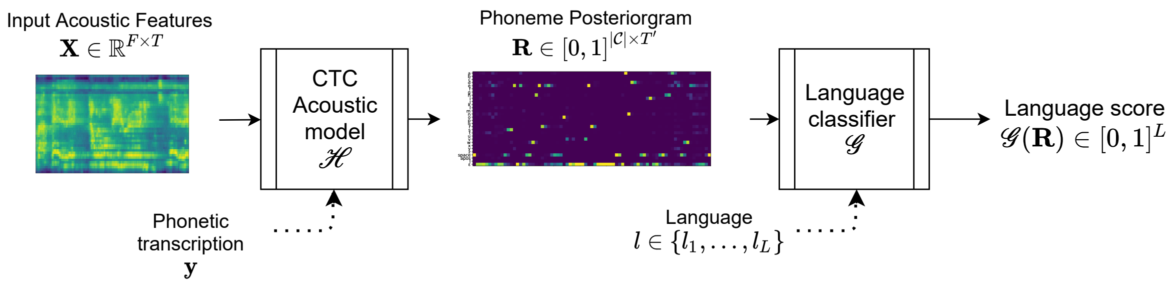

Input Acoustic Features
$\mathbf{X} \in \mathbb{R}^{F \times T}$
CTC Acoustic model $\mathscr{H}$
Phonetic transcription $\mathbf{y}$
Phoneme Posteriorgram
$\mathbf{R} \in [0,1]^{|\mathcal{C}| \times T'}$
Language classifier $\mathscr{G}$
Language $l \in \{l_1, \ldots, l_L\}$
Language score
$\mathscr{G}(\mathbf{R}) \in [0,1]^L$

Figure 8.1 – Overview of the proposed SLID system. During training, the phonetic transcription of the lyrics $\boldsymbol{y}$ and the corresponding language label l of the excerpt $\mathbf{X}$ are provided. To obtain the input acoustic feature matrix, multiple preprocessing modules are applied on the input audio, for both training and inference. These modules are, in order of application: SVS and feature extraction. They are not displayed here for the sake of readability.

obtained by computing the average of the recall of each class. More formally, it is defined as:

$$bAccuracy = \frac{1}{W} \sum_{l \in \mathcal{L}} \frac{TP(l)}{N_l} \tag{8.2}$$

where $\frac{1}{W}$ is a normalizing factor, N_l is the number of samples of the l^{th} class, $\mathcal{L}$ is the set of languages considered and TP(l) is the number of true positives of the l^{th} class. Secondly, the multi-class F1-score is computed using the macro-averaged strategy, considering equal weights for each class. Beforehand, the per-class F1-score is computed, for each class, by defining the considered class as the positive class and aggregating all other classes as the negative class. The desired metric is then obtained from the arithmetic mean of the per-class F1-scores like:

$$F1_{macro} = \frac{1}{L} \sum_{l \in \mathcal{L}} F1(l) \tag{8.3}$$

where L is the number of considered languages.

8.2 Proposed approach

As in previous SLID works, we frame the problem as a multiclass classification task. The system takes as input an acoustic feature matrix $\mathbf{X} \in \mathbb{R}^{F \times T}$, extracted from a musical excerpt. Here, F denotes the feature dimensionality and T is the number of time frames. The architecture then estimates the language l used in the musical excerpt, among a set of L languages $\{l_1, l_2, ..., l_L\}$. The preprocessing modules, and theirs parameters, preceding the acoustic model, *i.e.* the *Singing Voice Separation* (SVS) and the feature extraction ones, are identical to those described in Chapter 4.

8.2.1 General presentation

Our deep phonotactic system, as illustrated by Figure 8.1, is composed of two main models: an acoustic model $\mathscr{H}$ for phoneme estimation, followed by a language classifier $\mathscr{G}$. The acoustic model $\mathscr{H}$ is the same as the one described in Chapter 4, except that it is here outputting sequences of IPA phonemes. More precisely, the set of units supported by the model $\mathcal{C}$ encompasses the IPA symbols appearing in the training excerpts, a word-boundary 'space' token, an instrumental 'I' token and the blank token ϵ introduced by the CTC algorithm in Section 3.5. The rest of the architecture and training parameters are unchanged.

A posteriorgram is generated from the acoustic model $\mathscr{H}$ given the input acoustic features $\mathbf{X}$. The language classifier $\mathscr{G}$ then produces a language probability vector score $\mathscr{G}(\mathbf{R}) \in [0,1]^L$ from it. The language decision is finally taken from this vector score with:

$$\hat{l} = \underset{l \in \{l_1, l_2, \ldots, l_L\}}{\arg\max} \; [\mathscr{G}(\mathscr{H}(\mathbf{X}))]_l. \tag{8.4}$$

Previous phonotactic approaches on the SLID task made distinct training of the two models, with different training sets [MH11, Kru16c]. The acoustic model $\mathscr{H}$ needs the phonetic transcription $\boldsymbol{y}$ of each training excerpt X, whereas the language classifier $\mathscr{G}$ needs the posteriorgram representation $\mathbf{R}$ and the language label l of its training excerpts. We use the same dataset of musical excerpts for training both models. Following the work on joint *Automatic Speech Recognition* (ASR) and LID [WHH17], we train both models simultaneously. The joint loss function can be expressed as:

$$\mathscr{L}_{Joint}(\mathbf{R}, \hat{l}, \boldsymbol{y}, l) = \mathscr{L}_{\text{CTC}}(\mathbf{R}, \boldsymbol{y}) + \lambda \mathscr{L}_{\text{LID}}(\hat{l}, l), \tag{8.5}$$

where λ is the weight given to the cross-entropy LID loss with regard to the CTC loss. The balance between the two losses is decisive for the system performance: various strategies are considered and described in Section 8.3.4.

8.2.2 Language Classifier

The language classifier $\mathscr{G}$ is built upon a RNN. The usage of recurrent architectures has been successful for *End-To-End* (E2E) spoken LID [NTHAL16, CDHL19]. Here, the phoneme posteriorgram representation is given as input, instead of the raw acoustic features extracted from the audio excerpt. *Bidirectional Long Short-Term Memory* (BiLSTM) layers are chosen with the last layer only outputting a single probability vector for the whole input segment.

This architecture takes the combination of phonemes into account, as in n-gram modeling [MH11], but with confidence scores on the phoneme predictions by using the full probability vectors, as in statistical modeling [Kru16c]. To avoid vanishing gradients on a very long input sequence [HBF+01], we choose to perform the SLID task on fixed-length segments for a given song. Song language is then inferred from the mean of language scores output by the system on all segments.

Subset	# Phon	Train (h)	Val (h)	Test (h)
Closed-set	66	259.4	9.4	40.1
Open-set	71	269.2	10.6	46.0

Table 8.1 – Description of DALI language subset datasets used in this chapter and corresponding phoneme dictionary sizes.

8.3 Experiments

8.3.1 Datasets

We design two language sets from the DALI dataset: a *closed-set* scenario and an *open-set* scenario. The closed-set retains languages with more than ten hours of data each: English, French, German, Italian and Spanish. The open-set also adds a sixth label 'Others' regrouping low-resource languages as defined in Chapter 6 (Dutch, Finnish, Portuguese, Polish). Train, validation and test sets are obtained with an 84%-3%-13% language-wise and artist-aware split [Fle07].

Songs in neither target nor low-resource languages are also labeled as 'Others' and added to the open-set test set only. These out-of-domain samples help monitor the generalization of out-of-set modeling learned from the subset of in-domain 'Others' languages. As English is over-represented in the dataset, all systems and baselines are trained with a class-weighted LID objective function. The description of these splits are more precisely displayed in Table 8.1.

Our system performs SLID at segment-level. Each song is split into 20s segments with a 0.5 overlap factor between consecutive segments. Segment lyrics are retrieved using the word-level annotations from DALI and decomposed into IPA symbols using Phonemizer as in Chapter 6. Collecting all phonemes occurring in the training segments and adding the space, instrumental and blank tokens, the total number of units obtained is $|\mathcal{C}| = 66$ in the closed-set scenario and $|\mathcal{C}| = 71$ in the open-set scenario.

For the segment language label, the FastText language identifier [JGB+16] is used on the segment lyrics. A segment is labeled *instrumental* when it has less than three words, or *ambiguous* when the lyrics' repetitiveness or the FastText non-confidence score is above an empirically found threshold. During inference, *ambiguous* and *instrumental* scores are not taken into account when estimating the song language from segment language scores.

8.3.2 Baseline systems

Two baseline systems are implemented for comparison with our approach. The *Metadata* baseline is a text-based language identifier using the artist name and song title metadata provided with the DALI dataset. The language is extracted using the FastText language identifier [JGB+16]. The *i-vector* baseline is an acoustic-phonetic i-vector-based system, as in [Kru14b]. Implemented with the Kaldi LRE recipe [PGB+11], this system computes a 600-dimensional i-vector per vocal-isolated song. Sequences of 20-MFCC feature vectors are extracted then modeled by a GMM-based UBM. Supervised language classification is performed from the i-vector representation of the song using a *Support Vector Machine* (SVM) algorithm with a cosine kernel, as in [Kru14b, DTCRD11].

8.3.3 Language classifier

Inputted posteriorgrams are pre-processed by a deterministic cleaning module: frames with ϵ-emission probability $P(\epsilon) > 95\%$ are removed, to account only for frames with actual phoneme predictions. The language classifier model is composed of two BiLSTM layers with 64-dimensional hidden states each and a 0.1 recurrent dropout factor. An additional 0.2 dropout is applied between each layer. The second layer outputs a single vector per segment, which is processed by a dense layer with a softmax activation function to produce one language probability vector. A class-weighted categorical cross-entropy loss function is used for training the model given the one-hot encoded language label.

8.3.4 Training strategies for our approach

We test two strategies for training our system, implemented in Tensorflow. Each training variant relies on the ADAM optimization algorithm [KB15] with a learning rate of 10^{-3}, a batch size of 32 and validation-based early stopping. In the *2-step* variant, the acoustic model $\mathscr{H}$ is first trained alone. The language classifier $\mathscr{G}$ is therefore trained for the SLID task from the posteriorgrams of the training segments computed by $\mathscr{H}$. The *Joint* variant trains both models at the same time from scratch. With hyper-parameter tuning, we found that training the system with a loss balance $\lambda = 0.1$, then fine-tuning it with $\lambda = 100$ yields the best performance on the validation set.

8.3.5 Ablation study

We evaluate the relevance of our approach parts by designing two simplified systems for comparison. The *E2E* system is an E2E approach to the SLID task with the same architecture as the *Joint* variant, except for the CTC component which is removed from the loss function. The phoneme recognition task is ignored and the model is solely trained to identify the language in song segments.

The *Statistics* system is a modified *2-step* variant. Instead of the recurrent layers, the language classifier is a pooling step of the mean and variance statistics of each phoneme class over the full song length. Song language is directly predicted from these statistical vectors using a SVM. This system is analogous to a modernized version of [Kru16c], with a CTC-based acoustic model instead of the DNN-based one.

System	bAccuracy (%)	F1-score (%)
Metadata	76.48 (3.98)	76.71 (3.45)
i-vector	77.26 (3.88)	67.78 (3.57)
E2E	59.90 (4.33)	65.43 (4.47)
Statistics	88.46 (3.04)	89.00 (2.95)
2-step	88.62 (3.03)	90.75 (2.62)
Joint	**91.74 (2.70)**	**92.39 (2.31)**

Table 8.2 – System evaluation in the closed-set scenario. It is measured by balanced accuracy (bAccuracy) and macro-averaged F1-score. The standard errors are given in parentheses. The best value for each metric is displayed in bold.

8.4 Results and discussion

8.4.1 Performance in the closed-set scenario

The results of the evaluation of our systems on the test songs in a closed-set scenario are reported in Table 8.2. All phonotactic approaches (*Statistics*, *2-step* and *Joint*) outperform the *Metadata* baseline, unlike the *E2E* approach. The phonetic information contained in the audio data is thus better suited for estimating the language than common metadata. Reliable estimations from the raw audio can not be achieved with a naive E2E approach and seems to require more refined techniques. Our deep phonotactic system also significantly outperforms the re-implemented state-of-the-art *i-vector* approach. In particular, joint training of the acoustic model and language classifier further improves the system performance, as the *Joint* variant yields the best overall scores, with 91.7% of balanced accuracy.

Regarding the efficiency of each system's part, the *Statistics* approach has better performance that the *i-vector* baseline, which was not the case between the two analog approaches from Kruspe [Kru14b, Kru16c]. Hence, our CTC-based acoustic model seems to offer better modeling capability than the DNN-based model from [Kru16c]. The *2-step* variant does not significantly outperform the *Statistics* system, which implies that the language classifier can be improved. Finally, even though the side phoneme recognition task requires more detailed information for training, it proves to be profitable for the SLID task since the *2-step* and *Joint* systems outperform the *E2E* baseline.

8.4.2 Performance in the open-set scenario

The results of the evaluation of our systems on the test songs in the open-set scenario are reported in Table 8.3. All phonotactic systems still outperform the *Metadata*, *E2E* and *i-vector* approaches. However, both variants of our deep phonotactic approach are less robust to the introduction of the 'Others' class than the simpler *Statistics* system. Indeed, they seem to overfit on the 'Others' training data. It can be explained as this class has a greater linguistic variability than other classes but has the same amount of data as a low-resource target language. This effect is further demonstrated in Table 8.4

System	bAccuracy (%)	F1-score (%)	Target (%)	'Others' (%)
Metadata	70.16 (3.46)	70.52 (3.08)	73.30 (3.45)	56.60 (4.73)
i-vector	70.87 (3.16)	54.79 (2.90)	58.74 (3.18)	35.06 (4.92)
E2E	39.57 (3.17)	35.78 (2.57)	42.93 (3.07)	0.00 (0.00)
Statistics	**83.30 (2.83)**	**80.28 (2.79)**	**81.70 (3.03)**	**73.14 (3.79)**
2-step	78.49 (2.77)	74.35 (2.92)	79.89 (3.14)	46.62 (5.40)
Joint	72.89 (2.86)	64.46 (3.14)	72.02 (3.51)	26.67 (5.35)

Table 8.3 – System evaluation in the open-set scenario. It is measured by balanced accuracy (bAccuracy) and macro-averaged F1-score. Macro-averaged F1-score on the target languages and the F1-score on the 'Others' class are also presented. The standard errors are given in parentheses. The best value for each column is displayed in bold.

as only the *Statistics* approach can generalize the out-of-set modeling to out-of-domain languages unseen during training.

System	In-domain 'Others' (%)	Out-of-domain 'Others' (%)
i-vector	50.00 (10.66)	20.00 (4.46)
E2E	0.00 (0.00)	0.00 (0.00)
Statistics	**86.36 (7.29)**	**56.25 (5.58)**
2-step	63.64 (10.28)	21.25 (4.61)
Joint	31.82 (9.87)	11.25 (3.56)

Table 8.4 – Performances comparison on 'Others' labelled test songs in in-domain and out-of-domain languages cases. It is measured by accuracy. The standard errors are given in parentheses. The best value for each class is displayed in bold.

8.5 Conclusion

In this chapter, we investigated modernized phonotactic systems for the SLID task on polyphonic music, using recurrent models for both phoneme recognition and language classification. Trained on a publicly available multilingual dataset, the proposed approach outperforms metadata-based and the previous state-of-the-art SLID approaches. The CTC-based acoustic model greatly contributes to the performance increase, both in closed-set and open-set scenarios. However, the proposed language classifier hardly exceeds statistical modeling in a closed-set scenario, and deteriorates with out-of-set languages.

Future work could focus on exploring hierarchical language modeling techniques for SLID with out-of-set languages, taking inspiration from the speech literature [ISAL18]. Our SLID approach could also be used in a multilingual lyrics transcription system, conditioning the latter by the language detected using *e.g.* *Feature-wise Linear Modulation* (FiLM) layers. This technique has been particularly successful in conditioning the music source separation algorithm on the desired instrument to be extracted [MBP19].

Chapter 9

Cover song detection

Cover detection, also known as version identification, aims at detecting whether two recordings are of the same underlying musical work. A cover can be played by the same artist as the original song, or by another artist, and can be quite similar or vastly different. Practical applications include music browsing, recommendation and plagiarism detection. Generally, it is assumed, as in [SGH10], that tonal progression features (chord, melody, and harmony) are mostly preserved between covers of the same work. Inversely, musical attributes such as key, timbre, tempo, and structure significantly vary across covers [SGH10]. Variations of these features between covers were extensively studied in [YTC+19]. Cover detection systems are then built to be insensitive to these variations and exploit tonal progression features.

This task has been frequently studied as a *query and answer* [SGH10] one, *i.e.* given an input query, the system outputs a ranked list of possible covers from a music collection. True covers are to be ranked as highly as possible while other songs should be ranked low. This list is usually obtained by computing pairwise similarities between the query and each song of a predefined dataset [SGH10]. Obtained similarities can be further used to train a classifier to detect cover/non-cover pairs [RE10]. If earlier cover detection systems were shown to be efficient on small datasets (1000 songs or less) [SSA09], the performance quickly dropped on larger ones [CHA18, YTC+19]. Recent works have made significant advances in scalability and accuracy, for larger datasets, taking inspiration from metric learning [YSG20a] and knowledge distillation [YSG20b].

Almost none of the existing approaches explicitly consider the textual information provided by the lyrics. It is only used in [CHA18], in which lyrics are assumed to be available for a significant part of the dataset. In this study, the authors use metadata and lyrics alongside audio to perform cover detection. The textual similarity of lyrics and song titles is computed using a plain bag-of-word *Term Frequency–Inverse Document Frequency* (TFIDF).

The authors show results obtained with lyrics are on par with those given by audio-based features on a large-scale dataset. Moreover, the best results are obtained when combining all of the features. However, each feature is only used in a separate part of a multi-layer database pruning method; the information carried by each modality is not optimally combined. One limitation of this work is that it assumes the lyrics of most songs are available. The question of whether lyrics information extracted from audio might be employed to complete the cover detection task remains unresolved.

Considering the task of query by singing, which may be regarded as a related task to the cover detection one, authors in [WJ15] employed lyrics and melody recognition to recognize a singing query and match it against a collection of songs. They employed a basic bigram *Hidden Markov Models* (HMM) model that is trained on speech and adapted to singing voice. However, this approach also presupposes that the lyrics of songs are available. Lyrics from the considered dataset are, in fact, utilized to inform singing voice recognition.

In this chapter, we propose a novel cover detection approach leveraging lyrics information extracted from audio. It is based on the fusion of a *Singing Voice Recognition* (SVR) framework and a more classic tonal-based cover detection system. This is the first time an estimation of lyrics transcripts from audio is explicitly leveraged to perform cover detection. Our assumption, based on the results of [CHA18], is that lyrics are often preserved between covers in western popular music.

For the first modality of our fused system, we therefore propose using transcription methods to obtain estimates of these lyrics for all songs. The cover song here is framed as a noisy text-matching task. We expect a lyrics-recognition-based system to be particularly relevant for pairs of covers displaying hugely different tonal features while using the same lyrics. An example of such cases is the cover of *Summertime* by *Janis Joplin* where the harmony and melody are considerably different from the original score, but the lyrics remain quite similar.

Nevertheless, it is clear a pure lyrics-based approach is inadequate for instrumental music (*e.g.* without a singing voice). Therefore, we use a tonal-based system as the second modality of our fused architecture. An instrumental detector is applied on the output of the lyrics recognition framework to inform the fusion strategy. Extra attention is placed on the scalability of our proposed approach using an *Approximated Nearest Neighbor* (ANN) method.

The chapter is divided into five sections. First of all, some background for cover detection is presented in Section 9.1. Following that, our approach is described in Section 9.2. Then, the parameters of experiments are given in Section 9.3. Next, the results and discussions are presented in Section 9.4. At last, the conclusion is given in Section 9.5. A large part of this chapter is directly adapted from the paper: "Andrea Vaglio, Romain Hennequin, Manuel Moussallam, Gaël Richard. The Words Remain the Same: Cover Detection with Lyrics Transcription. *In Proc. of Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2021".

9.1 Required background

9.1.1 Cover detection

Related work:

Classically, cover song detection systems use tonal features, which are thought to be the least altered between a song and its covers. Chroma [EP07] and derived features like *Harmonic Pitch Class Profile* (HPCP) [SSA09] and CremaPCP [YSG20a] are among the most effective examples. Other features used include: melody [DP19], chord progression and profiles [KO13], and *Mel-Frequency Cepstral Coefficients* (MFCC) self-similarity matrices [Tra17]. Before computing the similarity between two songs, multiple preprocessing steps can be applied to obtain features that are invariant to the key, the tempo, or the structure of the song. Examples include multi-channel attention for structure invariance [YSG20a], optimal transposition index for key invariance [SGH08], and beat-synchronous Chroma for tempo invariance [BME11].

After extracting the features to be compared for both songs, a cross similarity matrix [LCCL18], or a cross recurrent plot [SGH08], is generally computed. A similarity score is then computed using dynamic programming algorithms such as *Dynamic Time Warping* (DTW) [SGHS08] or the recurrence quantification analysis one [SSA09]. For a given query, this score is calculated for all tracks in a pre-defined dataset and accordingly yields the desired sorted list. These methods achieve satisfactory results for small datasets of up to a thousand songs [SSA09], but are computationally costly for larger datasets.

To address this issue, some authors have attempted to reduce the size of the input representation to obtain a low-dimensional fixed size representation for each track. The similarity comparison thus boils down to a basic distance metric such as Euclidean distance or cosine similarity [YSG20a] that are much faster than dynamic programming algorithms of quadratic complexity. Early approaches of this type include using fingerprinting in the form of Chroma landmarks [BME11] and $2D$ Fourier transform of Chroma vectors [ET12], both obtaining low performance.

More recent approaches using metric learning, triplet loss, and distillation methods show great improvements [YSG20b, YSG20a] in terms of computation speed and retrieval performance. Database pruning was also used to decrease the overall complexity in [SHG17, CHA18]. A first fast global candidate selection using text and metadata was performed, followed by a more complex similarity function to re-rank the subset.

Most of these approaches, which are based on low-dimensional embeddings and simple distance functions, are then simply exploited into existing scalable nearest-neighbor methods. For example, the authors in [BME11, TPM16] use index-based matching on extracted audio fingerprints. A *Locality Sensitive Hashing* (LSH) method is also used on chord profiles [KO13], melodic fragments [Mar08], and audio shingles [GM12]. Scalable nearest-neighbor methods are broadly discussed in Section 9.2.6.

Metrics:

The empirical evaluation of the cover detection task performance is given using the *Mean Average Precision* (MAP) metric. It is computed with the Metrics toolkit [1]. For a query, the *Average Precision* (AP) metric quantifies the number of actual covers that are highly ranked in the returned sorted list. The AP score increases when actual covers are detected in the top ranks. It is formulated as:

$$AP = \frac{1}{G} \sum_{r=1}^{N} p(r) i(r) \tag{9.1}$$

where N is the size of the scrolled music collection and G is the number of actual covers in it. The function $i(r)$ is a binary indicator function returning one when the r^{th} song of the ranked list is a cover song and zero if not. The function $p(r)$ is the precision at rank r defined at:

$$p(r) = \sum_{j=1}^{r} \frac{i(j)}{r} \tag{9.2}$$

The MAP metric is then simply obtained by averaging on the AP of all queries. As the MAP is not properly defined for systems not scoring every track, like those using ANN methods, we report MAP@100 metric for these cases considering only the top 100 ranked items of each query. This choice can be explain by the fact all the systems in this chapter score at least 100 tracks. In any case, the MAP score does not significantly evolve after the top-100 pruning as explained in [CHA18]. In this case, the AP@100 metric is computed as:

$$AP@100 = \frac{1}{\min(100, G)} \sum_{r=1}^{100} p(r) i(r) \tag{9.3}$$

In practice, in this chapter, the number of relevant covers for a given query in the music corpus is always far below 100, hence obtaining:

$$AP@100 = \frac{1}{G} \sum_{r=1}^{100} p(r) i(r) \tag{9.4}$$

9.1.2 String matching

In this chapter, we assume lyrics of covers being mostly invariant. For two covers, the transcripts obtained from a SVR pipeline can then be considered as two noisy versions of the same text. Several metrics are therefore discussed in this section to compute the similarity of two noisy transcriptions coming from the same text. Looking at a few situations when the lyrics of covers diverge, two particular instances occur where lyrics are still similar. The first one, the most frequent one, is the case where both lyrics are only displaying minor differences (e.g. words added or deleted). Most string matching algorithms are robust to such modifications. The second one, is the case of the lyrics of one cover are a subsequence of lyrics of another one.

1. https://github.com/benhamner/Metrics

The similarity function used should then be invariant to such differences of length. The most natural choice to compute this similarity is an edit distance such as the Levenshtein distance. One of its interesting properties for the noisy text-matching task is it takes the order of characters into account. As described in Section 3.8, it is computed as the lowest number of needed character operations to transform one string to the other. In this chapter, possible operations are insertion and deletion, hence the substitution operation is not considered. A normalized similarity s can thus be defined from the Levenshtein distance l for two strings $\boldsymbol{a}$ and $\boldsymbol{b}$ by:

$$s(\boldsymbol{a},\boldsymbol{b}) = \frac{|\boldsymbol{a}| + |\boldsymbol{b}| - l(\boldsymbol{a},\boldsymbol{b})}{|\boldsymbol{a}| + |\boldsymbol{b}|} \tag{9.5}$$

This similarity takes its values between zero (when the two compared strings have no common characters) and one (when the two strings are equal). The complexity of this algorithm is $O(|\boldsymbol{a}||\boldsymbol{b}|)$. Unfortunately, the difference in size of the compared strings has a large influence on this measure, preventing it from possessing the necessary invariance characteristic.

In contrast, the cosine similarity between TFIDF vectors is another widely used metric which is unaffected by the difference of length of strings compared. These vectors are created using bag-of-words features. A bag-of-words simply counts, for a string $\boldsymbol{a}$, the number of occurrences of each word in a vocabulary $\mathcal{V}$ of size N, discarding unknown words. As a result, a vector of size N is obtained. The TFIDF is then used to generate a frequency representation of the bag-of-words, taking into consideration that certain words are common and so provide little information. More formally, it is defined as:

$$TFIDF(\boldsymbol{w},\boldsymbol{a},\mathcal{D}) = TF(\boldsymbol{w},\boldsymbol{a}) * IDF(\boldsymbol{w},\mathcal{D}) \tag{9.6}$$

where $TF(\boldsymbol{w},\boldsymbol{a})$ represents a function counting how many times the word $\boldsymbol{w}$ appears in the string $\boldsymbol{a}$ and $IDF(\boldsymbol{w},\mathcal{D})$ is a function computing how much information the word $\boldsymbol{w}$ is providing. The latter penalizes words too frequent in the corpus $\mathcal{D}$ and inversely favors very rare words in this corpus. It is usually defined as:

$$IDF(\boldsymbol{w},\mathcal{D}) = \log \frac{|\mathcal{D}|}{|\{\boldsymbol{d} \in \mathcal{D} : \boldsymbol{w} \in \boldsymbol{d}\}|} \tag{9.7}$$

where $\boldsymbol{d}$ can be any document of the corpus $\mathcal{D}$. The similarity between two strings $\boldsymbol{a}$ and $\boldsymbol{b}$ is finally obtained by computing cosine similarity of their respective TFIDF vectors of size N. For a given string $\boldsymbol{a}$, this vector is defined by $TFIDF(\boldsymbol{w},\boldsymbol{a},\mathcal{D})_{\boldsymbol{w}\in\mathcal{V}}$. The vocabulary $\mathcal{V}$ is commonly created from the collection of words found in the corpus $\mathcal{D}$. Contrary to the Levenshtein metric, this similarity does not consider the order of appearance of each word. The complexity of this algorithm is $O(|\boldsymbol{a}| + |\boldsymbol{b}|)$, thus outperforming the Levenshtein one in terms of complexity. One drawback of this measure is that it can provide a high score between two quite different lyrics when the same uncommon terms are employed in both. Though it diminishes the importance of common words in the similarity computation, it also increases the importance of rare words. This typically happens when both transcripts include the same rare onomatopoeia.

This metric can also be defined at character level and be generalized for sequences of the considered text units using n-grams. In this case, the vocabulary considered is composed

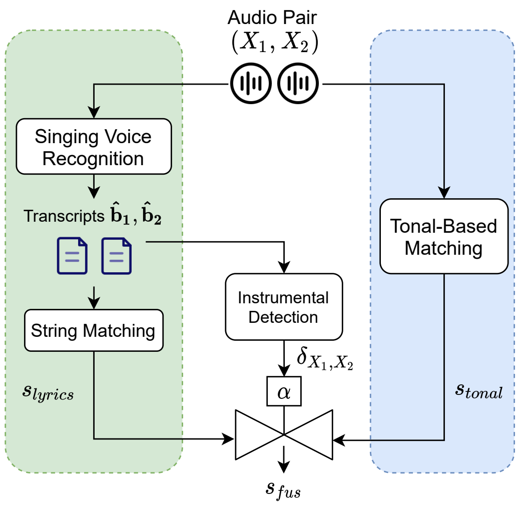


Figure 9.1 – Audio from a pair of tracks is processed in parallel by two branches computing lyrics and tonal-based similarities respectively. The fusion mechanism is informed by an *instrumental* detection on the transcripts.

of n-grams of characters or words. It allows to obtain vectors capturing some sequential information as opposed to the vanilla TFIDF. In practice, only n-grams of characters are employed since the size of TFIDF vectors generated with word n-grams quickly becomes very large for increasing values of n, even for bigram and trigram models. In this case, despite having a huge corpus of lyrics, a substantial proportion of word n-grams are only observed once in the corpus, severely weakening the statistical characteristics of the IDF function.

9.2 Proposed approach

A general overview of our cover detection approach is described in Figure 9.1. It is composed of a lyrics-recognition-based cover detection system and a classic tonal-based approach. The first branch is constituted of a lyrics recognition framework and a string matching function. It takes two songs X_1 and X_2 as input and outputs the respective estimated lyrics $\hat{\boldsymbol{b}}_1$ and $\hat{\boldsymbol{b}}_2$. A similarity estimation s_{lyrics} is obtained afterwards using these transcriptions. The second branch of our approach, the classic tonal-based system, also takes these two songs as input and outputs a similarity estimation s_{tonal}.

They are next fused using a fusion function to obtain a new similarity estimation s_{fus}. An extra input is added to the fusion function α to weight the participation of both modalities during the fusion. The value of this input depends on the instrumental detector taking as input both transcripts and outputting the probability that at least one of the tracks is purely instrumental. This avoids using the lyrics-based recognition approach during the fusion in the absence of lyrics. To obtain the desired sorted list, for a given query, a similarity is then computed for the considered system between the query and each track of the dataset. Finally, a fast approximate index search technique is used on our implementation to make it scalable. For that, we rely on the ANN approach where the similarity is only computed between the query and the nearest neighbors returned by the method.

9.2.1 Lyrics recognition

We choose as a lyrics recognition framework Gupta20 [GYL20] obtaining the best results in the *Music Information Retrieval Evaluation eXchange* (MIREX) 2020 lyrics transcription challenge [2]. The model is described extensively in Section 2.2. The complete framework extracts MFCC feature vectors of dimension 40 from the audio input and outputs the transcribed English words. As this model cannot output non-English words, extra care on the results of non-English tracks will be considered later in this chapter.

The acoustic model and the pronunciation lexicon are collected from the code implementation of the authors [3]. The language model used is the same as the one trained in Chapter 4. All three modules are compiled as a decoding graph using the Kaldi framework [PGB+11]. The graph obtained thus gives the desired estimated transcript for each audio input. Obtained transcription results are on par with those in the MIREX competition with a *Word Error rate* (WER) of 62% on the Jamendo dataset.

9.2.2 String matching

To allow for a swift computation of the similarity between pairs of estimated transcripts, each string is transformed to a vector using a TFIDF based on a trigram at character level with *Inverse Document Frequency* (IDF) values computed from the DALI dataset. Using a word level TFIDF was not shown to improve the performance on a cover song tuning set described in Section 9.3.1. We also considered the Levenshtein metric for the string matching, but it did not yield significant gains in performance while inducing a higher quadratic complexity. Similarily, taking into account both metrics for fusion did not significantly improve the performance.

9.2.3 Detecting instrumentals

Looking at various transcripts given by our SVR framework, we notice that, for most instrumental tracks, the transcript obtained is composed of either a few number of words, or highly repeated ones such as onomatopoeia. Therefore, we consider a track as instrumental if the respective transcription is composed of less than l different words with l tuned on the cover song tuning set. The module outputs $\delta_{x_1,x_2} = 1$ if both tracks are not detected as instrumentals, and zero otherwise. For some rare cases where the SVR framework is performing poorly, it is also only outputting few words. The instrumental detector then helps by filtering out some marginal cases where the lyrics transcription fails completely. We chose to keep this module very simple as it performed sufficiently well for our purposes and allowed for improvements in future work.

2. `https://www.music-ir.org/mirex/wiki/2020:MIREX2020_Results`
3. `https://github.com/chitralekha18/AutoLyrixAlign`

9.2.4 Tonal-based cover detection

The tonal-based cover detection method selected is described in [YSG20b]. This system, called Re-MOVE, is an updated version of MOVE [YSG20a] and obtains the second most accurate benchmark on the Da-Tacos dataset [YTC+19]. Compared to the best one reported [DYZ+21], it has the advantage of being publicly available[4] and, thus, perfectly reproducible.

The system is trained using the training part of Da-Tacos, as described in Section 9.3.1, and early stopping is performed using its validation component. For a given track, the system takes CremaPCP extracted from the audio as input and outputs a corresponding compact embedding. The CremaPCP feature is an intermediate representation of a chord estimation model. It is considered an efficient way to capture the tonal information of music and is shown to outperform the more classic HPCP features for cover detection [YTC+19]. The similarity between the query and each track of the dataset is then the cosine similarity of their respective embeddings.

The Re-MOVE architecture uses a latent space reconfiguration technique on top of MOVE in order to reduce the embedding dimension (and then reduce memory requirements and retrieval time) while maintaining a high detection performance. This technique reconfigures a pre-trained learned distance metric into a compact embedding space with the same learned semantic relations.

9.2.5 Fusion

It has been demonstrated in multiple domains that the fusion of different modalities can yield better performance than those obtained with each single modality [KHDM98]. For cover detection, fusing modalities, features or similarity matrices has already shown to improve results [CLX18, Tra17], notably using rank aggregation methods [OEFD15]. The fusion function chosen here is a weighted sum. It is precisely described by:

$$s_{fus} = \begin{cases} \alpha s_{lyrics} + (1-\alpha) s_{tonal} & \text{if } \delta_{x_1,x_2} = 1 \\ s_{tonal} & \text{otherwise} \end{cases} \tag{9.8}$$

α is a simple scalar defined as a hyperparameter to tune. As the distributions of both similarities are very different, calibration before fusion was also tested. However, no improvement was shown on the cover song tuning set. Other fusion functions, such as linear regression or max function, did not lead to improvements in our simulations.

9.2.6 Scalability

Pairwise comparisons between a given query and all tracks in a dataset have a linear complexity in the size of the dataset without optimization, which cannot be considered scalable. In fact, a linear complexity for the queries involves a quadratic complexity for retrieving all musical works in the dataset, as similarity of all pairs of tracks must

4. `https://github.com/furkanyesiler/re-move`

be calculated, which can quickly become prohibitive for large collections. To achieve better scalability properties, most cover detection studies use ANN methods like LSH [KO13, GM12].

The idea behind ANN is that for a given query x and a database D the method outputs an approximation of the k nearest neighbors of the query in the database with the complexity being sublinear in the size of the database. For a given query, in contrast with a classic *K-Nearest Neighbors* (KNN), the ANN methods are only browsing a subset of the complete search graph. All these methods are based on an index table allowing fast queries by outputting a 'good' guess of the k nearest neighbors of a given query, making it possible to recover most of the highly classified covers in the ranked list obtained with all candidate points. The recall is used to quantify the quality of an ANN method by averaging percentages obtained, on all queries, of true k-nearest-neighbors from k points returned by the method.

In our case, we use the *Hierarchical Navigable Small World Graph* (HNSW) algorithm which is the current state-of-the-art ANN method. An extensive description of it is given in [MY18]. This algorithm has a logarithmic complexity in the size of the dataset. This method is directly applied on Re-MOVE and TFIDF embeddings, outputting for a given query k nearest neighbors for each of them. Both sets of points are then concatenated and merged, obtaining a maximum of $2k$ points to consider for the fusion. Pairwise similarities between the query and these points are finally generated using the Re-MOVE system and our lyrics-recognition system.

9.3 Experiments

9.3.1 Datasets

Da-Tacos [YTC+19, YSG20b] is the largest publicly available dataset for cover detection; the training set is composed of 83904 songs in 14999 cliques and the validation set of 14000 songs in 3500 cliques. A clique is defined as a cover group gathering multiple recordings of the same underlying 'piece'. The Da-Tacos benchmark test subset is a 15000 track dataset composed of 1000 cliques with 13 songs each and 2000 noise songs (*i.e.* that are in a single-song clique) which are not queried. To avoid overfitting, no clique overlaps with any set of Da-Tacos. Instrumentals represent around 20% of the dataset which motivates our choice of using an instrumental detection process.

Currently, only a set of precomputed audio features is publicly available for the benchmarking subset test dataset. The dataset is mainly composed of English tracks and popular western music with a few non-English cliques. All hyperparameter tuning made during this chapter is carried out on a subpart of the Da-Tacos validation that we choose to refer to as a *Da-Tacos tuning* set. We verified no clique of this subset overlapped with any clique present in the dataset used to train the tonal-based cover detection system, *i.e.* the Da-Tacos training set. Also, a clique is discarded if it possesses one track present in the dataset used to train the SVR framework, *i.e.* the DALI dataset.

Detection of overlapping tracks and cliques is made using metadata, *i.e.* titles and artist names. *Da-Tacos tuning* is notably used to choose the string matching algorithm and the fusion function. We recover audio of 12862 tracks from the test dataset. 1849 are in single-song cliques and thus are not queried and only used as noise songs. We make sure no clique of this dataset overlaps with cliques in the Da-Tacos train and validation and that cliques possessing tracks existing in the DALI dataset are discarded. It will be simply referred to as *Da-Tacos test* in the rest of the document.

9.3.2 Fusion parameters

A track is classified as instrumental if the number of different words of its transcript is less than $l = 8$. This number is adjusted using *Da-Tacos tuning* as the value maximizing the recall for the highest $F1$ score. Emphasis is put on the recall in order to avoid taking into account the lyrics-recognition-based similarity for an instrumental track that has been misclassified as non-instrumental. An α value of 0.6 for fusing both systems is tuned on *Da-Tacos tuning*.

9.3.3 Parameters of ANN

We use the HNSW implementation of the NMSLIB similarity search library [BN13]. For each query, we return the $K = 100$ nearest neighbors. This choice is derived from [CHA18], which shows the performance does not evolve significantly after the top-100 pruning. We use an approximated cosine similarity function to retrieve 100 candidates for each branch which results in, at most, 200 items for the fused model after concatenating and merging both sets.

9.4 Results and discussion

9.4.1 Lyrics-recognition-based system results

Instrumental detection results:

Among the 12862 tracks in the test set, 3269 are detected as instrumentals. We compare this with the 'Instrumental' tag available in Da-tacos for all tracks. We obtain a precision of 82.86% for the instrumental detection, a recall of 96.68% and a F1 score of 89.24%. A deeper examination of misclassified tracks reveals the existence of annotation noise in the Da-Tacos annotations, which could artificially reduce the prior metrics. As simple as it is, the instrumental detection performance seems suitable for our application. After filtering detected instrumentals, we obtain a subset of 9593 tracks that we label *Da-Tacos-voice*. 1582 tracks are in single-song cliques. 8011 tracks are finally queried.

Lyrics-based cover detection results:

We first evaluate our lyrics-recognition-based system on the *Da-Tacos-voice*. Our results, displayed in Table 9.1, show it is generally performing better than the tonal-based one in

Query	System	MAP (%)
Da-Tacos-voice	Lyrics	**66.4 (0.4)**
	Tonal	54.0 (0.4)
Da-Tacos-instr	Lyrics	0.45 (0.06)
	Tonal	**47.8 (0.7)**

Table 9.1 – Results of lyrics-recognition-based and tonal-based cover detection systems on *Da-Tacos-voice*. *Da–acos-instr* is the subset of the *Da-Tacos test* restricted to instrumental tracks. The standard errors are given in parentheses. The best performance on each evaluation dataset is displayed in bold.

terms of MAP. They validate the assumption that lyrics can be considered as a strong invariant between covers. It also proves the most recent state-of-the-art SVR framework produces transcriptions of sufficiently good quality to perform the cover song task as a noisy text matching task. Looking empirically at results coming from both systems, most improvements of the lyrics-recognition approach over the tonal-based system come, as expected, from covers with significantly different tonal-content and lyrics being roughly the same.

We also query the tracks detected as instrumental and not from single-song cliques. Results are also displayed in Table 9.1. As expected, the performance of the lyrics-recognition-based system is close to zero. For the tonal-based approach, results seem to degrade when compared to non-instrumental tracks. This suggests the system has either learned characteristics of the melody carried by the singing voice or implicitly estimated some of the lyrics information to perform cover detection. Covers of instrumental music (which may include less pop and more jazz) could also be more difficult to match than non-instrumental ones.

The case of non-English tracks:

As stated in Section 9.2.1, our lyrics recognition framework cannot output non-English words, therefore non-English tracks may produce unexpected results. In order to assess the impact of this issue, we predict a language label for every track of the *Da-Tacos-voice* using a language classifier [JGBM16] taking track metadata as input. Results show the dataset is largely composed of English with more than 92.4% of the tracks being detected as English. Looking at tracks outside single-song cliques detected as non-English, half of them are false positives.

We query non-English tracks of the resulting 44 cliques on the *Da-Tacos-voice*, representing 299 tracks. It is interesting to report almost all cliques are homogeneous in terms of language. We obtain a MAP of 28% (2). Results demonstrate that even if the performance deteriorates for these cases, our system is often able to correctly classify these tracks. It can be explained as the chosen singing voice framework is transcribing something similar from one cover to another even for non-English lyrics. Considering the small quantity of non-English tracks and results on these tracks, we consider this issue to have a limited

Dataset	System	MAP (%)
Da-Tacos test	Fused	**62.7 (0.3)**
	Fused-wo-inst	50.2 (0.3)
	Tonal	50.6 (0.3)
Da-Tacos-voice	Fused	**80.4 (0.3)**

Table 9.2 – Results of fused, with and without instrumental detection, tonal and lyrics-recognition-based cover detection systems on various datasets. The standard errors are given in parentheses. The best performance on each evaluation dataset is displayed in bold.

impact on the performance in our evaluation setup.

9.4.2 Fused system results

Results for the fused system on the full *Da-Tacos test* and its *Da-Tacos-voice* subset are given in Table 9.2. The fused system significantly outperforms the results of the tonal-based or the lyrics-based ones on *Da-Tacos-voice*, displayed in Table 9.1, showing the validity of our assumption of both approaches being strongly complementary. The use of the instrumental detection module to inform the fusion strategy is empirically validated, with a major drop of the performance on *Da-Tacos test* occurring when it is not considered.

The fused system outperforms the tonal-based system by 12 absolute points for the MAP metric on the *Da-Tacos test* dataset. This improvement in performance is due to a rise in accuracy on the *Da-Tacos-voice* subset, where information from both branches is accessible, with the MAP measure reaching approximately 80%. To highlight this complementarity, similarities for sampled pairs of tracks from the *Da-Tacos-voice* are displayed in Figure 9.2. While the majority of same-clique pair lyrics and tonal similarities are significantly higher than non-matching pairs, there are multiple cases where one modality seems more indicative than the other. Level curves of s_{fus} are also displayed illustrating most pairs being linearly separable in the combined modality plane.

9.4.3 ANN results

We first evaluate the impact of pruning results to the first 100 candidates by computing the MAP@100 of the fused system on the *Da-Tacos test* dataset. When compared to the value displayed in Table 9.2, a small decrease is observed with a MAP@100 of 62.4% (0.3). After applying an ANN method to our fused approach, results remain the same with a MAP@100 of 62.4% (0.3) . This result can be explained as the recall of the HNSW method for both the tonal-based and lyrics-recognition systems being more than 99.5%. Thus, the scalability of our implementation is assured while maintaining the cover detection performance.

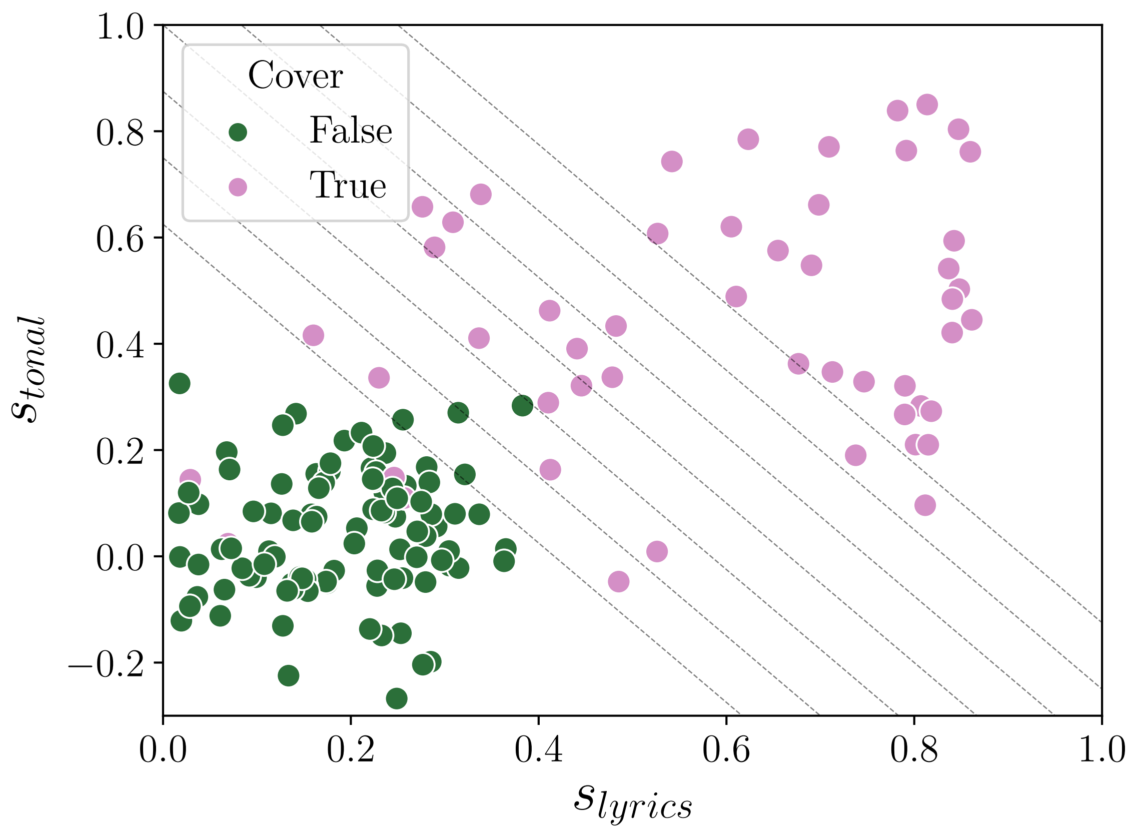


Figure 9.2 – Similarities of sampled pairs of tracks from the *Da-Tacos-voice*. Here, each point is a pair of tracks. Each color indicates a same-clique belonging status. Some level curves of s_{fus} are also displayed.

System	SVR	MAP (%)
Lyrics	CTC	40.3 (0.7)
	Ours	79.0 (0.6)
	Lyrics-informed	**89.7 (0.4)**
Fused	CTC	71.1 (0.6)
	Ours	88.5 (0.4)
	Lyrics-informed	**93.6 (0.4)**

Table 9.3 – Performance of lyrics-recognition and fused based cover detection systems on *Da-Tacos-lyrics* with various SVR frameworks. The lyrics-informed framework is informed by lyrics at test time. "Ours" SVR is the one used in the rest of the chapter. The standard errors are given in parentheses. The best performance of each system is displayed in bold.

9.4.4 Impact of the SVR framework

A detailed analysis of failing samples of the lyrics-recognition-based system shows the main cause for failure of the system is the low quality of the transcriptions. To further investigate this impact, we introduce two baselines by changing the SVR framework part of our approach.

In the first, an alternative *Connectionist Temporal Classification* (CTC) based SVR framework is used. The acoustic model used in this framework is the same as that giving the best multilingual generalization results for the lyrics-to-audio alignment task in Chapter 6. As a recall, this model is a *Recurrent Neural Network* (RNN) trained on multilingual data with a CTC algorithm and outputting *International Phonetic Alphabet* (IPA) phonemes. The language model used is the same as the one described in Section 9.2.1. Decoding is performed, after tuning the language model weight and the insertion penalty value using a validation dataset, with a CTC beam search decoding toolkit[5] as in Chapter 4. The transcription of the results obtained on the Jamendo dataset [SDE19] are significantly lower than our current SVR framework with a WER of 84.4%. We therefore expect this CTC-baseline to obtain results far below our system for the cover detection task.

In the second baseline, we simulate an 'ideal' SVR framework outputting an exact transcription. It can be considered as an oracle system, yielding an upper bound for the performance of lyrics-recognition-based approaches. To compare these three systems, we retrieve the lyrics information for part of the *Da-Tacos test*. To do so, audio from the dataset is matched to the Deezer catalog using fingerprinting. After keeping the mappings with a high confidence, we then retrieve lyrics from the corresponding tracks if existing. For the simplicity of this section, we discard all tracks detected as instrumental and do not consider the instrumental detector. The subset obtained is labeled *Da-Tacos-lyrics* and is composed of 3467 tracks. Again, tracks from single-song cliques are not queried and are used as noise songs.

The results obtained on *Da-Tacos-lyrics* are given in Table 9.3. These results confirm the

5. `https://github.com/parlance/ctcdecode`

intuition that the lyrics-recognition system's strength for covering detection tasks directly depends on the quality of the lyrics transcription. The ranking of the performances on *Da-Tacos-lyrics* for these approaches is conserved after fusing them with the tonal-based branch. In comparison to the oracle system, our fused approach shows excellent results even if there is still room for improvement.

With the transcription performance of our SVR framework being as low as 62% WER, it certainly indicates a perfect transcription is not needed for the cover detection task. Interestingly, even an oracle system informed by the true lyrics benefits from being fused with a tonal-based one. This, once again, demonstrates both branches are acutely complementary to address the cover song detection problem. Future work could extend our implementation to take into account cases where lyrics are available for a part of the dataset.

9.5 Conclusion

Using only audio, we have proposed a framework for cover detection explicitly leveraging two types of similarities, tonal and lyrics based, and reach high accuracy levels while remaining simple and scalable. The results presented support the use of lyrics information in cover identification as well as the complementarity of tonal and lyrics-based modalities. With that said, work on more diverse data still remains to be done, notably on non-English tracks where the performance seems to be limited.

Future work could include replacing the current monolingual lyrics recognition with a multilingual framework. A multilingual similarity, capable of detecting the similarity of two texts based on their semantic content, independently of their language, could also be defined and evaluated. More generally, the Da-Tacos dataset is quite biased towards popular western music. Additional experimentation on a wider range of genres, notably non-western music, and cover types (*e.g.* karaoke, renditions, etc.) remains to be conducted. Also, more elaborate fusion schemes could be explored, such as mid-level fusion, and possibly lead to improved performance.

Chapter 10

Conclusion

10.1 Summary

In this thesis, we explore various scientifically and industrially challenging *Music Information Retrieval* (MIR) tasks related to lyrics transcription. To do so, recent state-of-the-art *Singing Voice Recognition* (SVR) frameworks are successfully adapted for the considered tasks. Moreover, we take advantage of the recent release of the DALI dataset, a large public annotated singing voice database of great quality. Notably, a vanilla lyrics transcription pipeline is trained in Chapter 4 and serves as the basis of our SVR modules in most created systems. Four subjects are addressed: explicit content detection, lyrics-to-audio alignment, language identification and cover detection.

As claimed in the introduction, we aim at studying methods in realistic situations, as close to industrial cases as possible. For the language identification task, for example, we consider, notably, the first evaluation on out-of-set languages. In the case of cover detection, an *Approximated Nearest Neighbor* (ANN) algorithm is used to make the system scalable. Finally, for the lyrics-to-audio alignment task, efforts are made to generalize our approach to all types of languages, even when no resources are available. The created model, using phonemes as an intermediate representation, is able to work with any language as long as a pronunciation lexicon is available which is the case for a significant part of all spoken languages.

We also try evaluating our approaches on datasets as diversified and large as possible. Each system is noticeably evaluated on datasets of several thousands of tracks. For explicit content, the approaches are evaluated, for instance, on a English dataset of more than five thousand tracks. In the case of lyrics-to-audio alignment, the systems are rated on a variety of large datasets of various scripts and languages, one being notably composed of tracks with multilingual lyrics. The main research contributions of this thesis are described in the following paragraphs:

A first approach for explicit content detection from audio: A completely new method for explicit content detection based on lyrics information extracted from audio is developed. As described in Chapter 5, this method is based on various parts: an audio-to-character acoustic model, a keyword spotting module linked with a dictionary of keywords smartly constructed and a simple binary classifier to do the explicit detection. This method displays promising performances obtaining a F1-score of 67% on an industrial scale explicit content dataset.

It has multiple advantages. Firstly, novel keywords may be straightforwardly added into the dictionary without requiring to retrain the system. Secondly, its modular form makes it simple to deduce the reason for its decisions by looking at the keyword presence probabilities. This approach could therefore be ideal to be used as a tool to assist annotators in doing labeling. Results of these experiments, given the lyrics baseline and the highly subjective nature of the task, point towards the inadequacy of deploying explicit content detection systems without human oversight in a 'real-world' application.

A novel system to do multilingual lyrics-to-audio alignment: A first attempt to design a multilingual lyrics-to-audio system is presented in Chapter 6. The implementation is based on an acoustic model outputting sequences of *International Phonetic Alphabet* (IPA) phonemes trained with multilingual data. It shows great performance on a variety of languages of various scripts, with plenty to zero training resources available. Notably the first evaluation of a code-switching dataset is presented for the singing recognition field. In this example, gains on architectures trained with multilingual data over those trained with monolingual data are especially notable, with a relative increase of about ten percent for the *Percentage of Correct Onsets* (PCO) measure.

During alignment, additional information regarding phone similarity is also supplied, although without demonstrating any improvement. This model is then further developed in the form of an API which allows for automated lyrics-to-audio alignments for any requested track of the Deezer catalog. The results of these experiments show that it is advantageous to use phonemes as an intermediate representation when developing a lyrics-to-audio alignment system, even when trained and evaluated on one language. Indeed, this choice permits to easily improve multilingual generalization of the implemented system without degrading the performance on the target language.

First perception studies on lyrics and audio synchrony: Two online experiments inspired from karaoke were designed as described in Chapter 7. The asymmetrical perceptual thresholds of synchrony perception between lyrics and audio, as well as the effect of rhythmic variables on them, are the most important outcomes of these investigations. Unfortunately, despite collecting a vast pool of thousands of annotations, we were unable to demonstrate the effect of word local positions on the perception of lyrics-to-audio synchrony. These studies highlight the limits of current lyrics-to-audio metrics' perceptual validity as well as paving the way to the creation of more perceptually grounded ones.

A new method for lyrics language identification: As described in Chapter 8, a new approach for language identification is designed. This system is a modernized phonotactic approach composed of an acoustic model for phoneme recognition, and a language classifier module for language labeling. Both parts are made using recurrent networks. State-of-the-art results are achieved. Moreover, first results are described for the task on out-of-set languages.

The proposed acoustic model is demonstrated to greatly improve results on both closed-set and open-set scenarios. Notably, a F1-score of 92.39% is obtained for the proposed approach when evaluated on the closed-set dataset, a large evaluation database consisting of data from nine languages. However, the recurrent language classifier is shown to be limited, barely outperforming statistical modeling in the closed-set situation and strongly deteriorating over it in the out-of-set one. The findings of these studies suggest that the primary area of research that should be explored for the language identification task is the generalization of systems to out-of-set languages.

A novel cover detection approach using lyrics information from audio: As described in Chapter 9, we present the first cover detection system that explicitly leverages lyrics information from audio. It is based on the fusion of a SVR pipeline and a state-of-the-art tonal-based cover detection approach. The presented approach is simple, obtains high performances and is made scalable using an approximate nearest neighbor search algorithm.

The experimental results both validate the use of lyrics as a strong invariant to perform the cover detection task and the complementarity of both modalities used for fusion. On the Da-Tacos dataset, the fused system notably outperforms the tonal-based system by 12 absolute points on the *Mean Average Precision* (MAP) metric. Based on the results of these experiments, we affirm that using lyrics information to perform cover detection is an important direction of research for the task, with a lot of room for improvement, and should be more considered in the future.

Two reproducible benchmarks on the DALI dataset: The splits of the DALI dataset used to train and evaluate systems developed for both lyrics-to-audio alignment and language identification tasks are made available. The presented approaches are thus reproducible and future systems can then use these new large evaluation datasets to compare their performances.

10.2 Future work

In this thesis, we present a variety of systems, based on a lyrics recognition pipeline, performing lyrics-related tasks. One can reasonably assume that extracting transcripts, or posteriorgrams, of better quality will improve the performance of these approaches. When perfect transcription is given, it is notably shown to significantly improve the results for both explicit detection and cover detection tasks. All approaches presented in this thesis are modular and it is therefore straightforward to change the SVR implementation.

Such pipelines could be improved by enhancing the *Singing Voice Separation* (SVS) module, the acoustic model or the language model. However, two pipelines with identical performances can obtain vastly different results for a given lyrics-related task. An exhaustive study of what is significant to improve for the SVR implementation for each task is yet to be done. As tried in this thesis, the evaluation will need to keep on being as diverse as possible by incorporating more scripts, more languages and more genres of music in the test dataset.

From the limitations of the singing recognition implementation utilized in this thesis, we propose in the following paragraphs a number of future work and problems that could be investigated in order to enhance results on the lyrics transcription task. More broadly, we also suggest new research directions in the broader SVR field.

General limitations: In this thesis, two simplifying assumptions are implicitly made that we will discuss in this paragraph. Firstly, it is assumed that, in a given song, there is no superimposition of words (*i.e.* different lyrics at the same time). However, in practice, several people singing different lyrics at the same time frequently occurs. One can think of pop music where one rapper speaking during a chorus while another vocalist sings something else happens regularly. Although the approaches implemented in the thesis are capable of handling the scenario of several voices singing the same lyrics, they were not conceived to manage the situation where they are singing different lyrics.

If separation algorithms tend to extract all the voices together, the designed transcription systems do not provide for this case. This problem is known as monaural multi-speaker recognition and is a challenging issue still largely studied in the *Automatic Speech Recognition* (ASR) domain that is yet to be solved. One could take inspiration from the most recently developed systems in speech for this task to adapt them to the singing voice [CQYW19, CQY18].

Secondly, it is assumed that each word maps to a unique sequence of phonemes, i.e. pronunciation of each word is unique. Notably, the *Grapheme-To-Phoneme* (GTP) used to obtain phonetic transcription in Chapter 6 only outputs one pronunciation for each word. However, in practice, the pronunciation of a word is variable and the mapping transforming one word to a sequence of phonemes is thus not unique. Such a variability could seriously hinder system performance.

One can imagine lyrics to align being mapped to a different sequence of phonemes than what is actually pronounced in the audio, making the lyrics-to-audio alignment difficult to perform. This non-uniqueness can come from regional accents (*e.g.* "privacy" is pronounced /ˈpɹaɪvəsi/ in US English and /ˈpɹɪvəsi/ in UK English), or from the context of words (*e.g.* "read" is pronounced /ɹid/ or /ɹɛd/ whether it is the present or the past). Taking into account this non-uniqueness in singing voice recognition systems could be done during training, alignment and decoding to improve robustness of implemented systems.

Singing voice separation: Cases where systems fail often include source separation artifacts. Other than improving the SVS module, one potential avenue to solve these problems would be to jointly train source separation and acoustic model architectures. Still, the required training dataset is yet to be created. More generally, as described in Chapter 2, an exhaustive evaluation for polyphonic music of benefits of using a SVS module in a SVR pipeline, for both training and testing, still has to be done.

Acoustic model: Training efficient and robust acoustic models could be done by leveraging larger and more diverse datasets than DALI. Notably, as described in Section 2.1, it should particularly help the SVR systems employing *End-To-End* (E2E) acoustic models, requiring more data to train than traditional phoneme-based architectures. Moreover, systems trained on the DALI dataset obtain poor performance on certain genres of music such as rap. It may be explained as the dataset is largely rock oriented. Leveraging a more diverse dataset could help improve generalization of the trained systems on music genres. However, this sort of singing dataset is generally either incomplete or unannotated.

Taking as an example the Deezer dataset described in Section 2.4, the corresponding annotations are frequently of bad quality or incomplete, with some parts of lyrics missing for instance. In the case of an incomplete dataset, training could be achieved by taking inspiration from lightly supervised training [BKFI17] which has been successfully used for automatic broadcast subtitles.

The main concept behind this method is to detect which portions of the annotations are valid and only utilize these for training. More precisely, a first random subset of the training dataset is used to train a baseline recognition model. The transcripts obtained from all training data using this model are compared with their respective ground truths. Depending on the distance between the pairs of strings obtained, the segments are either corrected or removed. The new labeled data is used to train a novel baseline model. The steps are repeated iteratively until convergence of the performance of the trained system is achieved.

Unannotated datasets, on the other part, could be leveraged with active learning [YVDA10]. In this method, a first baseline system is trained on a small amount of manually labeled data from the corpus. This system is next used to automatically determine which missing annotations in the rest of the corpus would be the most beneficial for training. These annotations are then manually annotated, and a new system is trained using the newly available pool of labeled data. The process is repeated iteratively until the performance of the trained system converges.

Annotations can also be generated with the latest alignment algorithms as proposed in [DAD21]. Kruspe [Kru18] notably employs this method, but solely considers *a cappella* extracts. Another way to generate additional data could be data augmentation. It is, for example, considered in [SG15] for singing voice detection, employing classic augmentation methods such as pitch shifting, or in [BAGT21] using a more recent singing voice style transfer algorithm. In the latter, the authors leverage a large dataset of speech that they transform to a 'singing-like' dataset with a vocoder based speech synthesizer converting natural speech to singing voice. Finally, the architecture could be also improved using more recent approaches such as the *Connectionist Temporal Classification* (CTC)-attention joint one [KHW17] or the transformer [BAGT21] architectures.

Language modelling: The language models used in this thesis were trained using the DALI lyrics corpus. Training more effective language models could be achieved by leveraging larger corpora of lyrics such as in [DB20]. Another way to improve the quality of language modelling could be to perform transfer learning from models trained on text, using for example a linear interpolation as described in [GGL20]. Furthermore, while the architecture employed in this thesis is a simple n-gram model, utilizing more recent language models such as *Bidirectional Encoder Representations from Transformers* (BERT) might assist to increase performance [DCLT18].

Character level language models could also be considered. They possess many advantages over word-level ones. The main one being they do not suffer from *Out Of Vocabulary* (OOV) words. OOV phenomenons are especially studied in [LSC19]. The authors show the superiority of lexicon-free decoding character-based language models on utterances with OOV words over lexicon-based decoding with either character or word-based language models, where decoding is limited to words of a vocabulary, without losing performance on general transcription. Training the acoustic model and the character language model jointly, as explored in the ASR domain [SWW+19], could be done additionally to further improve performance on the lyrics recognition task.

Addressing new tasks: Taking inspiration from cover detection systems, multiple tasks could be performed leveraging lyrics information extracted explicitly from audio and fusing it with a model trained on melodic features. Here we present a few examples of such tasks, each having already been performed in the literature using both audio and pre-existing lyrics. A SVR pipeline could then be introduced to get rid of the need for such lyrics.

Firstly, SVS systems could be improved by employing lyrics information. Indeed, the authors in [JCL20] show extending a classic separation system with a highway network based lyrics encoder, taking as input preexisting aligned lyrics, significantly improves the performance. They further demonstrate improvements are not only due to the vocal activity information, but also to the phonetic contents of the inputted aligned lyrics.

For this task, SVR systems not utilizing a preprocessing of SVS algorithm, as Gupta20, should be employed. Indeed, SVS approaches are known to perform badly on some specific cases (e.g. when autotune is used). As a result, a SVR system using this sort of preprocessing also fails in these instances and will be unable to enhance SVS systems in these cases. For these situations, a system like Gupta20 could be more useful by providing complementary information to those captured by the SVS system.

Secondly, studying the music auto-tagging subject could also be an interesting direction for future work. In [PFOT19], the authors present a system using both lyrics and audio modality for detection of music emotion, which can be considered as a particular case of music auto-tagging. They investigate various ways of fusing both modalities and demonstrate its superiority over an unimodal system.

Finally, the structure extraction task could represent another interesting subject to explore. It is extensively studied in [Fel20], where the authors introduce an approach using both audio and textual features, the latter being extracted from preexisting lyrics, significantly improving state-of-the-art results over text only-based systems.

Bibliography

[AAA$^+$16] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 173–182, 2016.

[AB18] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

[And18] Catherine Anderson. In *Essentials of Linguistics*. 2018.

[ATS16] Tanel Alumäe, Stavros Tsakalidis, and Richard Schwartz. Improved multilingual training of stacked neural network acoustic models for low resource languages. In *Proc. of the Annual Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pages 3883–3887, 2016.

[BAGT21] Sakya Basak, Shrutina Agarwal, Sriram Ganapathy, and Naoya Takahashi. End-to-end lyrics recognition with voice to singing style transfer. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 266–270, 2021.

[BCC$^+$17] Eric Battenberg, Jitong Chen, Rewon Child, Adam Coates, Yashesh Gaur Yi Li, Hairong Liu, Sanjeev Satheesh, Anuroop Sriram, and Zhenyao Zhu. Exploring neural transducers for end-to-end speech recognition. In *Proc. of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 206–213, 2017.

[BKFI17] Ismael Bada, Juan Karsten, Dominique Fohr, and Irina Illina. Data selection in the framework of automatic speech recognition. In *Proc. of the Int. Conf. on Natural Language, Signal and Speech Processing (ICNLSSP)*, pages 1–5, 2017.

[BME11] Thierry Bertin-Mahieux and Daniel PW. Ellis. Large-scale cover song recognition using hashed chroma landmarks. In *Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 117–120, 2011.

[BN13] Leonid Boytsov and Bilegsaikhan Naidan. Engineering efficient and effective non-metric space library. In *Proc. of the Similarity Search and Applications Conf. (SISAP)*, 2013.

[Bor80] Marc H. Bornstein. The ecological approach to visual perception, 1980.

[CBL$^+$19] Jaejin Cho, Murali Karthick Baskar, Ruizhi Li, Matthew Wiesner, et al. Multilingual sequence-to-sequence speech recognition: Architecture, transfer learning, and language modeling. In *Proc. of the IEEE Spoken Language Technology Workshop (SLT)*, pages 521–527, 2019.

[CDHL19] Weicheng Cai, Cai Danwei, Shen Huang, and Ming Li. Utterance-level end-to-end language identification using attention-based cnn-blstm. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5991–5995, 2019.

[CFSC17] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2392–2396, 2017.

[CHA18] Albin Andrew Correya, Romain Hennequin, and Mickaël Arcos. Large-scale cover song detection in digital music libraries using metadata, lyrics and audio features. *arXiv:1808.10351*, 2018.

[CLX18] Ning Chen, Wei Li, and Haidong Xiao. Fusing similarity functions for cover song identification. *Multimedia Tools and Applications*, 77(2):2629–2652, 2018.

[con] Table of ipa pulmonic consonant. Found in https://commons.wikimedia.org/wiki/File:The_Int._Phonetic_Alphabet_(2015_version)_(cropped)_(only_pulmonic_consonants).svg, Int. Phonetic Association (2015), CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons. Accessed: 2021-06-04.

[CPH14] Guoguo Chen, Carolina Parada, and Georg Heigold. Small-footprint keyword spotting using deep neural networks. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4087–4091, 2014.

[CQY18] Xuankai Chang, Yanmin Qian, and Dong Yu. Monaural multi-talker speech recognition with attention mechanism and gated convolutional networks. In *Proc. of the Annual Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pages 1586–1590, 2018.

[CQYW19] Xuankai Chang, Yanmin Qian, Kai Yu, and Shinji Watanabe. End-to-end monaural multi-speaker asr system without pretraining. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6256–6260, 2019.

[CSR11] Vijay Chandrasekhar, Mehmet Sargin, and David Ross. Automatic language identification in music videos with low level audio and visual features. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5724–5727, 2011.

[DAD21] Emir Demirel, Sven Ahlbäck, and Simon Dixon. Low resource audio-to-lyrics alignment from polyphonic music recordings. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[DB08] Eirlys E. Davies and Abdelâli Bentahila. Translation and code switching in the lyrics of bilingual popular songs. *The Translator*, 14(2):247–272, 2008.

[DB19] Gerardo Roa Dabike and Jon Barker. Automatic lyric transcription from karaoke vocal tracks: Resources and a baseline system. In *Proc. of the Annual Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pages 579–583, 2019.

[DB20] Gerardo Roa Dabike and Jon Barker. The sheffield university system for the mirex 2020: Lyrics transcription task. *Music Information Retrieval Evaluation eXchange (MIREX)*, 2020.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018.

[DED08] Adrien Daniel, Valentin Emiya, and Bertrand David. Perceptually-based evaluation of the errors usually made when automatically transcribing music. In *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2008.

[DK16] Aliya Deri and Kevin Knight. Grapheme-to-phoneme models for (almost) any language. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 399–408, 2016.

[DKD$^+$11] Najim Dehak, Patrick Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-end factor analysis for speaker verification. *IEEE Trans. on Audio, Speech, and Language Processing (TASLP)*, 2011.

[DM80] Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, 1980.

[DP19] Guillaume Doras and Geoffroy Peeters. Cover detection using dominant melody embeddings. In *Proc. of the Int. Soc. for Music Information Retrieval (ISMIR)*, 2019.

[DS14] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6964–6968, 2014.

[DS15] Georgi Dzhambazov and Xavier Serra. Modeling of phoneme durations for alignment between polyphonic audio and lyrics. In *Proc. of the Int. Conf. in Sound and Music Computing (SMC)*, pages 281–286, 2015.

[DTCRD11] Najim Dehak, Pedro A. Torres-Carrasquillo, Douglas Reynolds, and Reda Dehak. Language recognition via i-vectors and dimensionality reduction. In *Proc. of the Annual Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pages 857–860, 2011.

[Dun10] Knight Dunlap. Reaction to rhythmic stimuli with attempt to synchronize. *Psychological Review*, 17:399–416, 1910.

[DYRS16] Georgi Dzhambazov, Yile Yang, Rafael Caro Repetto, and Xavier Serra. Automatic alignment of long syllables in a cappella beijing opera. In *Proc. of the Int. Workshop on Folk Music Analysis (FMA)*, pages 88–91, 2016.

[DYZ$^+$21] Xingjian Du, Zhesong Yu, Bilei Zhu, Xiaoou Chen, and Zejun Ma. Bytecover: Cover song identification via multi-loss training. In *Proc. of the*

IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), 2021.

[Dzh17] Georgi Dzhambazov. *Knowledge-Based Probabilistic Modeling For Tracking Lyrics In Music Audio Signals*. PhD thesis, Universitat Pompeu Fabra Barcelona, 2017.

[EA18] Bronwen G. Evans and Wafaa Alshangiti. The perception and production of british english vowels and consonants by arabic learners of english. *Journal of Phonetics*, 68:15–31, 2018.

[EP07] Daniel PW. Ellis and Graham E. Poliner. Identifying cover songs' with chroma features and dynamic programming beat tracking. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal (ICASSP)*, volume 4, 2007.

[ET12] Daniel PW. Ellis and Bertin-Mahieux Thierry. Large-scale cover song recognition using the 2d fourier transform magnitude. In *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2012.

[Faw04] Tom Fawcett. Roc graphs: Notes and practical considerations for researchers. *Machine learning*, 2004.

[FCCG19] Michael Fell, Elena Cabrio, Michele Corazza, and Fabien Gandon. Comparing automated methods to detect explicit content in song lyrics. In *Proc. of the Recent Advances in Natural Language Processing Conf. (RANLP)*, 2019.

[Fel20] Michael Fell. *Natural language processing for music information retrieval: deep analysis of lyrics structure and content*. PhD thesis, Université Côte d'Azur, 2020.

[FGO$^+$06] Hiromasa Fujihara, Masataka Goto, Jun Ogata, et al. Automatic synchronization between lyrics and music cd recordings based on viterbi alignment of segregated vocal signals. In *Proc. of the IEEE Int. Symposium on Multimedia (ISM)*, pages 257–264, 2006.

[FGO08] Hiromasa Fujihara, Masataka Goto, and Jun Ogata. Hyperlinking lyrics : A method for creating hyperlinks between phrases in song lyrics. In *Proc. of the Int. Conf. on Music Information Retrieval (ISMIR)*, pages 281–286, 2008.

[FGOO11] Hiromasa Fujihara, Masataka Goto, Jun Ogata, and Hiroshi G. Okuno. Lyric synchronizer: Automatic synchronization system between musical audio signals and lyrics. *IEEE Journal on Selected Topics in Signal Processing (JSTSP)*, 5(6):1252–1261, 2011.

[Fle07] Arthur Flexer. A closer look on artist filters for musical genre classification. In *Proc. of the Int. Conf. on Music Information Retrieval (ISMIR)*, pages 341–344, 2007.

[For73] David G. Forney. The viterbi algorithm. *Proc. of the IEEE*, 61(3):268–278, 1973.

[GBM$^+$11] Ondřej Glembek, Lukáš Burget, Pavel Matějka, Martin Karafiát, and Patrick Kenny. Simplification and optimization of i-vector extraction. In

Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), pages 4516–4519, 2011.

[GFGS06] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 369–376, 2006.

[GGL20] Xiaoxue Gao, Chitralekha Gupta, and Haizhou Li. Lyrics transcription and lyrics-to-audio alignment with music-informed acoustic models. *Music Information Retrieval Evaluation eXchange (MIREX)*, 2020.

[GJ14] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. *Proc. of the Int. Conf. on Machine Learning (ICML)*, 32, 2014.

[GKD+06] Fabien Gouyon, Anssi Klapuri, Simon Dixon, Miguel Alonso, George Tzanetakis, Christian Uhle, and Pedro Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Trans. on Audio, Speech, and Language Processing (TASLP)*, 14:1832–1844, 2006.

[GM12] Peter Grosche and Meinard Müller. Toward characteristic audio shingles for efficient cross-version music retrieval. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 473–476, 2012.

[GMH13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649, 2013.

[GTLW18] Chitralekha Gupta, Rong Tong, Haizhou Li, and Ye Wang. Semi-supervised lyrics and solo-singing alignment. In *Proc. of the Int. Conf. on Music Information Retrieval (ISMIR)*, pages 600–607, 2018.

[GYL20] Chitralekha Gupta, Emre Yılmaz, and Haizhou Li. Automatic lyrics transcription in polyphonic music: Does background music help? In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[Han12] Jens K. Hansen. Recognition of phonemes in a-cappella recordings using temporal patterns and mel frequency cepstral coefficients. In *Proc. of the Sound and Music Computing Conf. (SMC)*, 2012.

[Han17] Awni Hannun. Sequence modeling with ctc. *Distill*, 2017. Figure found in `https://distill.pub/2017/ctc/`, licensed under Creative Commons Attribution 4.0 Int. Accessed: 2021-06-04.

[HB19] Andre Holzapfel and Emmanouil Benetos. Automatic music transcription and ethnomusicology: a user study. In *Proc of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 678–684, 2019.

[HBF+01] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *A field guide to dynamical recurrent neural networks*, 2001.

[HDY$^+$12] Geoffrey Hinton, Li Deng, Dong Yu, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[HES00] Hynek Hermansky, Daniel PW Ellis, and Sangita Sharma. Tandem connectionist feature extraction for conventional hmm systems. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 3, pages 1635–1638, 2000.

[HK12] Xiao Hu and Noriko Kando. User-centered measures vs. system effectiveness in finding similar songs. In *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 331–336, 2012.

[HKVM20] Romain Hennequin, Anis Khlif, Félix Voituret, and Manuel Moussallam. Spleeter: a fast and efficient music source separation tool with pre-trained models. *Journal of Open Source Software*, 5(50):2154, 2020.

[HLS15] Kyuyeon Hwang, Minjae Lee, and Wonyong Sung. Online keyword spotting with a character-level recurrent neural network. *Arxiv:1512.08903*, 2015.

[HMN16] Hossein Hamooni, Abdullah Mueen, and Amy Neel. Phoneme sequence recognition via dtw-based classification. *Knowledge and Information Systems (KAIS)*, 48(2):253–275, 2016.

[Ho95] Tin Kam Ho. Random decision forests. In *Proc. of the Int. Conf. on Document Analysis and Recognition (ICDAR)*, volume 1, pages 278–282, 1995.

[HPR$^+$17] Yanzhang He, Rohit Prabhavalkar, Kanishka Rao, Wei Li, Anton Bakhtin, and Ian McGraw. Streaming small-footprint keyword spotting using sequence-to-sequence models. In *Proc. of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 474–481, 2017.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HSIM05] Toru Hosoya, Motoyuki Suzuki, Akinori Ito, and Shozo Makino. Lyrics recognition from a singing voice based on finite state automaton for music information retrieval. In *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2005.

[HW16] Allen Huang and Raymond Wu. Deep learning for music. *arXiv:1606.04930*, 2016.

[Ind19] British Phonographic Industry. Parental advisory guidelines. `https://www.bpi.co.uk/media/1047/parental-advisory-guidelines.pdf`, 2019. Online; accessed June 9, 2021.

[ISAL18] Saad Irtza, Vidhyasaharan Sethu, Eliathamby Ambikairajah, and Haizhou Li. End-to-end hierarchical language identification system. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

[JB89] Mari Riess Jones and Marilyn Boltz. Dynamic attending and responses to time. *Psychological Review*, pages 459–491, 1989.

[JCL20] Chang-Bin Jeon, Hyeong-Seok Choi, and Kyogu Lee. Exploring aligned lyrics-informed singing voice separation. In *Proc. of the IEEE Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2020.

[JGB+16] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. *arXiv:1612.03651*, 2016.

[JGBM16] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv:1607.01759*, 2016.

[JHM+17] Andreas Jansson, Eric Humphrey, Nicola Montecchio, et al. Singing voice separation with deep u-net convolutional networks. In *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2017.

[KAD14] Anna Kruspe, Jakob Abeßer, and Christian Dittmar. A gmm approach to singing language identification. *Journal of the Audio Engineering Soc. (AES)*, pages 140–148, 2014.

[KB15] Diederik P. Kingman and Jimmy Ba. Adam: A method for stochastic optimization. *Proc. of the Int. Conf. for Learning Representations (ICLR)*, 2015.

[KCH+19] Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, et al. A comparative study on transformer vs rnn in speech applications. In *Proc. of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 449–456, 2019.

[KHDM98] Josef Kittler, Mohamad Hatef, Robert PW Duin, and Jiri Matas. On combining classifiers. *IEEE transactions on pattern analysis and machine intelligence*, 20(3):226–239, 1998.

[KHW17] Suyoun Kim, Takaaki Hori, and Shinji Watanabe. Joint ctc-attention based end-to-end speech recognition using multi-task learning. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4835–4839, 2017.

[Kle57] Edmund T. Klemmer. Simple reaction time as a function of time uncertainty. *Journal of experimental psychology*, 54(3):195–200, 1957.

[KM19] Jayong Kim and Yi Y. Mun. A hybrid modeling approach for an automated lyrics-rating system for adolescents. In *Proc. of the European Conf. on Information Retrieval (ECIR)*, pages 779–786. Springer, 2019.

[KNH+05] Thomas R. Knösche, Christiane Neuhaus, Jens Haueisen, Kai Alter, Burkhard Maess, Otto W. Witte, and Angela D. Friederici. Perception of phrase structure in music. *Human Brain Mapping*, 24(4):259–273, 2005.

[KO13] Maksim Khadkevich and Maurizio Omologo. Large-scale cover song identification using chord profiles. In *Proc. of the Int. Soc. for Music Information Retrieval (ISMIR)*, volume 13, pages 233–238, 2013.

[Kru14a] Anna Kruspe. Keyword spotting in a-capella singing. In *Proc. of the Int. Soc. of Music Information Retrieval Conf. (ISMIR)*, 2014.

[Kru14b] Anna M. Kruspe. Improving singing language identification through I-vector extraction. In *Proc. of the Int. Conf. on Digital Audio Effects (DAFx)*, 2014.

[Kru16a] Anna Kruspe. Automatic B**** Detection. In *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 3–4, 2016.

[Kru16b] Anna Kruspe. Bootstrapping a system for phoneme recognition and keyword spotting in unaccompanied singing. In *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2016.

[Kru16c] Anna M. Kruspe. Phonotactic language identification for singing. In *Proc. of the Annual Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pages 3319–3323, 2016.

[Kru18] Annamaria Kruspe. *Application of Automatic Speech Recognition Technologies to Singing Doctoral Thesis.* PhD thesis, University Fraunhofer, 2018.

[LCC$^+$18] Yair Lakretz, Gal Chechik, Evan-Gary Cohen, Alessandro Treves, and Naama Friedmann. Metric learning for phoneme perception. *arXiv:1809.07824*, 2018.

[LCCL18] Juheon Lee, Sungkyun Chang, Sang Keun Choe, and Kyogu Lee. Cover song identification using song-to-song cross-similarity matrix with convolutional neural network. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 396–400, 2018.

[LJ96] Fred Lerdahl and Ray S. Jackendoff. *A Generative Theory of Tonal Music.* The MIT Press, 1996.

[LLY$^+$19] Ke Li, Jinyu Li, Guoli Ye, Rui Zhao, and Yifan Gong. Towards code-switching asr for end-to-end ctc models. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[LML13] Haizhou Li, Bin Ma, and Kong Aik Lee. Spoken language recognition: From fundamentals to practice. *Proc. of the IEEE*, 101(5):1136–1159, 2013.

[LP12] Carla Lopes and Fernando Perdigão. Broad phonetic class definition driven by phone confusions. *EURASIP Journal on Advances in Signal Processing*, 2012.

[LSC19] Tatiana Likhomanenko, Gabriel Synnaeve, and Ronan Collobert. Who needs words? lexicon-free speech recognition. In *Proc. of the Annual Conf. of the Int. Speech Communication Association (INTERSPEECH)*, 2019.

[MAD$^+$17] Miroslav Malik, Sharath Adavanne, Konstantinos Drossos, Tuomas Virtanen, Dasa Ticha, and Roman Jarina. Stacked convolutional and recurrent neural networks for music emotion recognition. *Sound and Music Computing Conf. (SMC)*, 2017.

[Mar08] Matija Marolt. A mid-level representation for melody-based retrieval in audio collections. *IEEE Transactions on Multimedia*, 10(8):1617–1625, 2008.

[MBCHP20] Gabriel Meseguer-Brocal, Alice Cohen-Hadria, and Geoffroy Peeters. Creating dali, a large dataset of synchronized audio, lyrics, and notes. *Trans. of the Int. Soc. for Music Information Retrieval (TISMIR)*, 3(1), 2020.

[MBP19] Gabriel Meseguer-Brocal and Geoffroy Peeters. Conditioned-u-net: Introducing a control mechanism in the u-net for multiple source separations. In *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2019.

[MCCD18] Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2018.

[Mes12] Annamaria Mesaros. *Singing Voice Recognition for Music Information Retrieval.* PhD thesis, Tampere university of technology, 2012.

[Mes13] Annamaria Mesaros. Singing voice identification and lyrics transcription for music information retrieval invited paper. In *Proc. of the Conf. on Speech Technology and Human - Computer Dialogue (SpeD)*, 2013.

[MFG12] Matthias Mauch, Hiromasa Fujihara, and Masataka Goto. Integrating additional chord information into hmm-based lyrics-to-audio alignment. *IEEE Trans. on Audio, Speech, and Language Processing (TASLP)*, 20(1):200–210, 2012.

[MGH$^+$14] Alvin F. Martin, Alvin F. Greenberg, John M. Howard, Doddington George R., and Godfrey John J. Nist language recognition evaluation - past and future. In *Proc. of the Odyssey The Speaker and Language Recognition Workshop*, pages 145–151, 2014.

[MH11] Mahnoosh Mehrabani and John Hansen. Language identification for singing. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4408–4411, 2011.

[MJC94] Yeshwant K. Muthusamy, Neena Jain, and Ronald A. Colen. Perceptual benchmarks for automatic language identification. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 1994.

[MKB$^+$10] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proc. of the Annual Conf. of the Int. Speech Communication Association (INTERSPEECH)*, 2010.

[MMC$^+$05] Jose P. G. Mahedero, Álvaro MartÍnez, Pedro Cano, Markus Koppenberger, and Fabien Gouyon. Natural language processing of lyrics. In *Proc. of the Annual ACM Int. Conf. on Multimedia*, pages 475—-478, 2005.

[MNR08] Rudolf Mayer, Robert Neumayer, and Andreas Rauber. Rhyme and style features for musical genre classification by song lyrics. In *Proc. of the Int. Soc. of Music Information Retrieval Conf. (ISMIR)*, pages 337–342, 2008.

[MS07] J. Devin McAuley and Miller N. S. Picking up the pace: Effects of global temporal context on sensitivity to the tempo of auditory sequences. *Perception & Psychophysics*, 69:709–718, 2007.

[MSW17] Markus Müller, Sebastian Stüker, and Alex Waibel. Language adaptive multilingual ctc speech recognition. In *Proc. of the Int. Conf. on Speech and Computer (SPECOM)*, pages 473–482. Springer, 2017.

[MV08] Annamaria Mesaros and Tuomas Virtanen. Automatic alignment of music audio and lyrics. In *Proc. of the Int. Conf. on Digital Audio Effects (DAFx)*, pages 1–4, 2008.

[MV10a] Annamaria Mesaros and Tuomas Virtanen. Automatic recognition of lyrics in singing. *EURASIP Journal on Audio, Speech, and Music Processing*, 2010.

[MV10b] Annamaria Mesaros and Tuomas Virtanen. Automatic recognition of lyrics in singing. *EURASIP Journal on Audio, Speech, and Music Processing*, 2010.

[MVK07] Annamaria Mesaros, Tuomas Virtanen, and Anssi Klapuri. Singer identification in polyphonic music using vocal separation and pattern recognition methods. In *Proc. of the IEEE Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2007.

[MY18] Yu A. Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

[NEK94] Hermann Ney, Ute Essen, and Reinhard Kneser. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech & Language*, 8(1):1–38, 1994.

[NTHAL16] Trung Ngo Trong, Ville Hautamäki, and Kong Aik Lee. Deep language: a comprehensive deep learning approach to end-to-end language recognition. In *Proc. of the Odyssey The Speaker and Language Recognition Workshop*, pages 109–116, 2016.

[oA20] Recording Industry Association of America. Us sales database. https://www.riaa.com/u-s-sales-database/, 2020. Online; accessed June 20, 2021.

[OEFD15] Julien Osmalsky, Jean-Jacques Embrechts, Peter Foster, and Simon Dixon. Combining features for cover song identification. In *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2015.

[otPI20] International Federation of the Phonographic Industry. Industry data. https://www.ifpi.org/our-industry/industry-data/, 2020. Online; accessed June 20, 2021.

[PCKSH13] Chen Pei-Chun, Lin Keng-Sheng, and Chen Homer. Emotional accompaniment generation system based on harmonic progression. *IEEE Trans. on Multimedia*, 15(7):1469–1479, 2013.

[PCPK15] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *Proc. of the IEEE Int. Conf. on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210, 2015.

[PCZ$^+$19] Daniel S. Park, William Chan, Yu Zhang, et al. Specaugment: A simple data augmentation method for automatic speech recognition. In *Proc. of the Annual Conf. of the Int. Speech Communication Association (INTERSPEECH)*, 2019.

[PDvG17] Ben Peters, Jon Dehdari, and Josef van Genabith. Massively multilingual neural grapheme-to-phoneme conversion. In *Proc. of the First Workshop on Building Linguistically Generalizable NLP Systems*, 2017.

[PFOT19] Loreto Parisi, Simone Francia, Silvio Olivastri, and Maria Stella Tavella. Exploiting synchronized lyrics and vocal features for music emotion detection. *arXiv:1901.04831*, 2019.

[PGB+11] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, et al. The kaldi speech recognition toolkit. In *Proc. of the IEEE Automatic Speech Recognition and Understanding (ASRU)*, 2011.

[PNP+17] Jordi Pons, Oriol Nieto, Matthew Prockup, et al. End-to-end learning for music audio tagging at scale. *arXiv:1711.02520*, 2017.

[PRWZ02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proc. of the annual meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, 2002.

[RÁA14] Pablo Ruiz, Aitor Álvarez, and Haritz Arzelus. Phoneme similarity matrices to improve long audio alignment for automatic subtitling. In *Proc. of the Int. Conf. on Language Resources and Evaluation (LREC)*, pages 437–442, 2014.

[Rab89] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[Rad19] National Public Radio. 'parental advisory' labels — the criteria and the history. https://www.npr.org/sections/therecord/2010/10/29/130905176/you-ask-we-answer-parental-advisory---why-when-how, 2019. Online; accessed June 9, 2021.

[RE10] Suman Ravuri and Daniel PW Ellis. Cover song detection: from high scores to general classification. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 65–68, 2010.

[Rep06] Bruno Repp. Sensorimotor synchronization: A review of the tapping literature. *Psychonomic bulletin & review*, 12:969–92, 2006.

[RMH+14a] Colin Raffel, Brian Mcfee, Eric Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel Ellis. mir_eval: A transparent implementation of common mir metrics. In *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2014.

[RMH+14b] Colin Raffel, Brian McFee, Eric J Humphrey, et al. mir_eval: A transparent implementation of common mir metrics. In *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2014.

[RP02] Bruno Repp and Amandine Penel. Auditory dominance in temporal processing: New evidence from synchronization with simultaneous visual and auditory sequences. *Journal of experimental psychology. Human perception and performance*, 28, 2002.

[SB01] Jayren J. Sooful and Elizabeth C. Botha. An acoustic distance measure for automatic cross-language phoneme mapping. *Pattern Recognition Association of South Africa (PRASA)*, 1:99–102, 2001.

[SBWH06] Jochen Schwenninger, Raymond Brueckner, Daniel Willett, and Marcus Hennecke. Language identification in vocal music. In *Proc. of the Int. Conf. on Music Information Retrieval (ISMIR)*, pages 377–379, 2006.

[SdCQS$^+$15] Haşim Sak, Félix de Chaumont Quitry, Tara Sainath, Kanishka Rao, et al. Acoustic modelling with cd-ctc-smbr lstm rnns. In *Proc. of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 604–609, 2015.

[SDE19] Daniel Stoller, Simon Durand, and Sebastian Ewert. End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 181–185, 2019.

[SG15] Jan Schlüter and Thomas Grill. Exploring data augmentation for improved singing voice detection with neural networks. In *Proc. of the Int. Conf. on Music Information Retrieval (ISMIR)*, pages 121–126, 2015.

[SGH08] Joan Serra, Emilia Gómez, and Perfecto Herrera. Transposing chroma representations to a common key. In *Proc. of the IEEE CS Conf. on The Use of Symbols to Represent Music and Multimedia Objects*, pages 45–48, 2008.

[SGH10] Joan Serra, Emilia Gómez, and Perfecto Herrera. Audio cover song identification and similarity: background, approaches, evaluation, and beyond. In *Advances in Music Information Retrieval*, pages 307–332. Springer, 2010.

[SGHS08] Joan Serra, Emilia Gómez, Perfecto Herrera, and Xavier Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Trans. on Audio, Speech, and Language Processing (TASLP)*, 16(6):1138–1151, 2008.

[SGLW19] Bidisha Sharma, Chitralekha Gupta, Haizhou Li, and Ye Wang. Automatic lyrics-to-audio alignment on polyphonic music using singing-adapted acoustic models. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 396–400, 2019.

[SHG17] Jordan BL. Smith, Masahiro Hamasaki, and Masataka Goto. Classifying derivative works with search, text, audio and video features. In *Proc. of the IEEE Int. Conf. on Multimedia and Expo (ICME)*, pages 1422–1427, 2017.

[SMH18] Miikka Silfverberg, Lingshuang Jack Mao, and Mans Hulden. Sound analogies with phoneme embeddings. In *Proc. of the Soc. for Computation in Linguistics (SCiL)*, pages 136–144, 2018.

[SP97] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[SRP18] Yann Soullard, Cyprien Ruffino, and Thierry Paquet. Ctcmodel: Connectionist temporal classification in keras. *arXiv:1901.07957*, 2018.

[SSA09] Joan Serra, Xavier Serra, and Ralph G. Andrzejak. Cross recurrence quantification for cover song identification. *New Journal of Physics*, 11(9), 2009.

[SvHP97] Agaath M. C. Sluijter, Vincent J. van Heuven, and Jos J. A. Pacilly. Spectral balance as a cue in the perception of linguistic stress. *The Journal of the Acoustical Soc. of America (JASA)*, 101(1):503–513, 1997.

[SVN37] Stanley Smith Stevens, John Volkmann, and Edwin Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America (JASA)*, 8(3):185–190, 1937.

[SW01] Tanja Schultz and Alex Waibel. Language-independent and language-adaptive acoustic modeling for speech recognition. *Speech Communication*, 35(1-2):31–51, 2001.

[SWW$^+$19] Changhao Shan, Chao Weng, Guangsen Wang, et al. Component fusion: Learning replaceable language model component for end-to-end speech recognition system. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5361–5635, 2019.

[TL18] Raphael Tang and Jimmy Lin. Deep residual learning for small-footprint keyword spotting. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5484–5488, 2018.

[TPM16] Timothy J. Tsai, Thomas Prätzlich, and Meinard Müller. Known artist live song id: A hashprint approach. In *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 427–433, 2016.

[Tra17] Christopher J. Tralie. Early mfcc and hpcp fusion for robust cover song identification. In *Proc. of the Int. Soc. for Music Information Retrieval (ISMIR)*, 2017.

[TSW$^+$18] Shubham Toshniwal, Tara N. Sainath, Ron J. Weiss, et al. Multilingual speech recognition with a single end-to-end model. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4904–4908, 2018.

[TW04] Wei-Ho Tsai and Hsin-Min Wang. Towards automatic identification of singing language in popular music recordings. In *Proc. of the Int. Conf. on Music Information Retrieval (ISMIR)*, pages 568–576, 2004.

[vEKJvdP08] Rob L. J. van Eijk, Armin Kohlrausch, James F. Juola, and Steven van de Par. Audiovisual synchrony and temporal order judgments: effects of experimental method and stimulus type. *Attention, Perception, & Psychophysics*, 70(6):955—968, 2008.

[VFMA18] Argiro Vatakis, Balci Fuat, Di Luca Massimiliano, and Correa Angel. *Timing and Time Perception: Procedures, Measures, & Applications.* Brill, 2018.

[VHM$^+$20] Andrea Vaglio, Romain Hennequin, Manuel Moussallam, Gael Richard, and Florence D'alché-Buc. Multilingual lyrics-to-audio alignment. In *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2020.

[Vin12] Emmanuel Vincent. Improved perceptual metrics for the evaluation of audio source separation. In *Proc. of the Int. Conf. on Latent Variable Analysis and Signal Separation (LVA/ICA)*, pages 430—437, Berlin, Heidelberg, 2012. Springer-Verlag.

[VMR08] Tuomas Virtanen, Annamaria Mesaros, and Matti Ryynänen. Combining pitch-based inference and non-negative spectrogram factorization in separating vocals from polyphonic music. In *Proc. of the Annual Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pages 17–22, 2008.

[vow] Ipa vowel chart. Founded in https://upload.wikimedia.org/wikipedia/commons/8/89/IPA_vowel_chart.svg, IPA_vowel_trapezium.svg: Moxfyre (talk)Ga_open_allophones.svg: Angr (talk)derivative work: Mr KEBAB, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons. Accessed: 2021-06-04.

[WHH17] Shinji Watanabe, Takaaki Hori, and John Hershey. Language independent end-to-end architecture for joint language and speech recognition. In *Proc. of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2017.

[WJ15] Chung-Che Wang and Jyh-Shing Roger Jang. Improving query-by-singing/humming by combining melody and lyric information. *IEEE/ACM Trans. on Audio, Speech, and Language Processing (TASLP)*, 23(4):798–806, 2015.

[WT97] Colin W. Wightman and David T. Talkin. The aligner: Text-to-speech alignment using markov models. In *Progress in speech synthesis*, pages 313–323. Springer, 1997.

[XDH+16] Wayne Xiong, Jasha Droppo, Xuedong Huang, et al. Achieving human parity in conversational speech recognition. *Arxiv:1512.08903*, 2016.

[XWA+18] Wayne Xiong, Lingfeng Wu, Fil Alleva, et al. The microsoft 2017 conversational speech recognition system. In *Proc. of the IEEE Int. Conf. on acoustics, speech and signal processing (ICASSP)*, pages 5934–5938, 2018.

[YHX13] Zhi-Jie Yan, Qiang Huo, and Jian Xu. A scalable approach to using dnn-derived features in gmm-hmm based acoustic modeling for lvcsr. In *Proc. of the Annual Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pages 104–108, 2013.

[YJMTTS07] Luo Yin-Jyun, Chen Ming-Tso, and Chi Tai-Shih. Singing voice correction using canonical time warping. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 396–400, 2007.

[YLBP20a] Adrien Ycart, Lele Liu, Emmanouil Benetos, and Marcus Pearce. Investigating the perceptual validity of evaluation metrics for automatic piano music transcription. *Trans. of the Int. Soc. for Music Information Retrieval (TISMIR)*, 3:68–81, 2020.

[YLBP20b] Adrien Ycart, Lele Liu, Emmanouil Benetos, and Marcus T. Pearce. Musical features for automatic music transcription evaluation. *arXiv:2004.07171*, 2020.

[YSG20a] Furkan Yesiler, Joan Serra, and Emilia Gómez. Accurate and scalable version identification using musically-motivated embeddings. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 21–25, 2020.

[YSG20b] Furkan Yesiler, Joan Serra, and Emilia Gomez. Less is more: Faster and better music version identification with embedding distillation. In *Proc. of the Int. Soc. for Music Information Retrieval (ISMIR)*, 2020.

[YTC+19] Furkan Yesiler, Chris Tralie, Albin Andrew Correya, Diego F. Silva, Philip Tovstogan, Emilia Gómez Gutiérrez, and Xavier Serra. Da-tacos: A dataset for cover song identification and understanding. In *Proc. of the Int. Soc. for Music Information Retrieval (ISMIR)*, 2019.

[YVDA10] Dong Yu, Balakrishnan Varadarajan, Li Deng, and Alex Acero. Active learning and semi-supervised learning for speech recognition: A unified framework using the global entropy reduction maximization criterion. *Computer Speech & Language*, 24(3):433–444, 2010.

List of Figures

List of Tables

Appendix

A Lyrics to audio alignment tables of results

Target	Source	Mean AAE (s)	Mean PCO (%)
English	English	0.61 (0.13)	**89 (1)**
	5lang	**0.58 (0.12)**	**89 (1)**
	5langb	0.79 (0.16)	85 (1)
German	English	0.89 (0.23)	83 (3)
	German	0.67 (0.22)	88 (3)
	5lang	**0.59 (0.15)**	**89 (3)**
	5langb	0.63 (0.20)	**89 (3)**
French	English	0.76 (0.34)	79 (4)
	French	0.78 (0.34)	83 (3)
	5lang	0.64 (0.32)	85 (3)
	5langb	**0.45 (0.16)**	**86 (3)**
Spanish	English	0.26 (0.06)	91 (3)
	Spanish	0.16 (0.02)	95 (1)
	5lang	0.13 (0.03)	**96 (1)**
	5langb	**0.12 (0.03)**	**96 (1)**
Italian	English	1.01 (0.50)	78 (5)
	Italian	0.32 (0.07)	**85 (5)**
	5lang	0.36 (0.07)	**85 (4)**
	5langb	**0.30 (0.04)**	**85 (4)**
Portuguese	English	0.75 (0.27)	84 (4)
	5lang	0.68 (0.36)	87 (4)
	5langb	**0.47 (0.21)**	**88 (3)**
Polish	English	0.66 (0.10)	81 (2)
	5lang	0.58 (0.12)	**88 (2)**
	5langb	**0.51 (0.09)**	86 (2)
Finnish	English	0.23 (0.03)	94 (1)
	5lang	0.18 (0.02)	**96 (1)**
	5langb	**0.15 (0.02)**	**96 (1)**
Dutch	English	0.48 (0.15)	87 (2)
	5lang	**0.28 (0.07)**	**91 (2)**
	5langb	0.38 (0.13)	90 (2)

Table A.1 – Lyrics-to-audio evaluation on DALI language subset datasets for character based architectures. Several training set design strategies are considered. Here 'source' refers to data language used to train the given model and 'target' refers to data language used to evaluate the trained model. The dataset 5langb is obtained by balancing the dataset 5lang with oversampling. AAE is better if smaller, PCO is better if larger. Standard errors over tested songs are given in parentheses.

Target	Source	Mean AAE (s)	Mean PCO (%)
English	English	0.58 (0.13)	86 (1)
	5lang	**0.52 (0.13)**	**91 (1)**
	5langb	0.65 (0.13)	88 (1)
German	English	0.62 (0.16)	85 (3)
	German	0.64 (0.21)	88 (3)
	5lang	**0.50 (0.16)**	**90 (3)**
	5langb	0.55 (0.20)	**90 (3)**
French	English	0.68 (0.31)	80 (3)
	French	0.46 (0.14)	86 (3)
	5lang	**0.38 (0.16)**	**88 (3)**
	5langb	0.57 (0.29)	87 (3)
Spanish	English	0.20 (0.02)	93 (1)
	Spanish	0.16 (0.02)	95 (1)
	5lang	**0.10 (0.03)**	**97 (1)**
	5langb	0.11 (0.03)	**97 (1)**
Italian	English	0.54 (0.09)	81 (4)
	Italian	0.41 (0.10)	84 (4)
	5lang	0.28 (0.05)	**86 (5)**
	5langb	**0.23 (0.04)**	**86 (4)**
Portuguese	English	**0.53 (0.20)**	85 (3)
	5lang	0.54 (0.29)	**88 (3)**
	5langb	0.54 (0.29)	87 (3)
Polish	English	0.40 (0.05)	86 (1)
	5lang	**0.23 (0.03)**	**92 (1)**
	5langb	0.35 (0.06)	91 (1)
Finnish	English	0.23 (0.05)	94 (1)
	5lang	**0.10 (0.01)**	**97 (1)**
	5langb	0.11 (0.01)	**97 (1)**
Dutch	English	0.54 (0.28)	87 (2)
	5lang	**0.24 (0.08)**	**93 (2)**
	5langb	0.34 (0.14)	91 (2)

Table A.2 – Lyrics-to-audio evaluation on DALI language subset datasets for phoneme based architectures. Several training set design strategies are considered. Here 'source' refers to data language used to train the given model and 'target' refers to data language used to evaluate the trained model. The dataset 5langb is obtained by balancing the dataset 5lang with oversampling. AAE is better if smaller, PCO is better if larger. Standard errors over tested songs are given in parentheses.

B Published papers

SINGING VOICE SEPARATION: A STUDY ON TRAINING DATA

Laure Prétet[⋆†] *Romain Hennequin*[⋆] *Jimena Royo-Letelier*[⋆] *Andrea Vaglio*[⋆†]

[⋆] Deezer R&D, Paris, France, research@deezer.com
[†] LTCI, Télécom ParisTech, Université Paris-Saclay, Paris, France

ABSTRACT

In the recent years, singing voice separation systems showed increased performance due to the use of supervised training. The design of training datasets is known as a crucial factor in the performance of such systems. We investigate on how the characteristics of the training dataset impacts the separation performances of state-of-the-art singing voice separation algorithms. We show that the separation quality and diversity are two important and complementary assets of a good training dataset. We also provide insights on possible transforms to perform data augmentation for this task.

Index Terms— source separation, supervised learning, training data, data augmentation

1. INTRODUCTION

Singing voice separation is the decomposition of a music recording into two tracks, the singing voice on one side, and the instrumental accompaniment on the other side. Typical applications are automatic karaoke creation, remixing, pitch tracking [1], singer identification [2], and lyrics transcription [3].

This is a highly popular topic in the Music Information Retrieval (MIR) literature and yearly competitions such as the SiSec MUS challenge gather an increasing number of teams (24 systems evaluated in 2016, 30 in 2018). The 2018 edition of the SiSec campaign [4] shows that the best current systems rely on supervised, deep-learning based models. In particular, Convolutional Neural Networks (CNN) seem to be especially adapted for this task. Recently, a U-Net [5] and several DenseNet-based systems [6] showed impressive performance: for the first time, state-of-the-art models performed similarly to oracle systems for the instrumental part [4].

However, despite these achievements, it is often difficult to identify what is the main success factor of these systems. Results are generally presented for a full procedure, including dataset building, data pre-processing and/or augmentation, architecture design, post-processing and sometimes a long engineering work to tune the hyperparameters of the models [7, 8, 5, 9].

In this work, we focus on the influence of the training dataset on the performances of a state-of-the-art deep-learning based separation systems. We investigate the impact of four different aspects of these (size, separation quality, use of data augmentation techniques and use of separated sources from several instruments to estimate voice separation) by training a same baseline model while varying the training dataset. In the previous literature [10, 11, 12, 13, 9] different architectures are usually compared using the same train/test datasets, but to the best of our knowledge, there are no previous works that study particularly the influence of these datasets. As opposed to the previous works, we use one single state-of-the-art architecture and train it on different datasets in order to reveal the effect of diverse characteristics of the training data on separation performances. We notably inspect the following aspect: data diversity and separation quality, data augmentation, and number of separated sources.

Diversity and Separation Quality. In the literature, data scarcity is often cited as one of the main limits for building efficient and scalable supervised singing voice separation algorithms [14, 15, 16]. Indeed, public training datasets have been regularly released (MIR-1K [17], MedleyDB [18], DSD100 [19] MUSDB [20]) and used to compare different methods, but they are rather small, and often lack diversity. We propose here to use several datasets of different sizes and separation qualities to evaluate the benefits of training systems with larger amounts of data. These include a relatively small public database (MUSDB), a large private dataset, and a large dataset with estimated separated tracks build from Deezer's music catalog following the technique presented in [21].

Data Augmentation. A common method used to artificially increase the size of a dataset for MIR tasks is data augmentation. For instance, in singing voice detection, some data augmentation like pitch shifting or the application of random frequency filters have proven to increase performance [22]. Also, in [8] the authors studied the use of other data augmentations (channels swapping, amplitude scaling or random chunking) with no improved results. We propose to study the influence of using several data augmentation techniques over a small sized dataset.

Several Sources. Finally, we study the influence of using several sources (the *bass*, *drums* and *other* parts available in MUSDB) for estimating the *instrumental* part. Indeed, when only estimating the *vocal* and *instrumental* parts, source separation systems tend to include in the vocals estimation residual parts from other instruments (in particular from *drums*). Hence, using the additional information included in multiple sources could lead to a better modeling of the *instrumental* part, and thus to a better separation.

The rest of the paper is organized as follows. In Section 2, we introduce the three datasets that we used for our experiments. In Section 3, we detail the methodology that we put in place to compare the performances on the different datasets. In Section 4, we expose our results and discuss possible interpretations. Finally, we draw conclusions in Section 5.

2. DATASETS

In this section, we present the three training datasets that we used in our experiments, along with their main characteristics. In addition to the total duration of audio, we define a *quality* criterion and a *diversity* criterion. The quality of the dataset reflects the quality of the source separation in the dataset's tracks: in two datasets (MUSBD and Bean), the separated tracks come from different recordings, while in the last one (Catalog), the vocal part was not available as separate track and had to be estimated. In the last case, the separated tracks being only estimates, residuals from other sources can be present in the ground truth tracks. This criterion does not account for the production quality, nor the audio quality. The diversity criterion reflects the variability of songs from which the dataset was built. It can be quantified by the number of different songs that are represented by one or more segments in the dataset. This information is summarized in Table 1.

2.1. MUSDB

MUSDB is the largest and most up-to-date public dataset for source separation. MUSDB is mainly composed of songs taken from DSD100 and MedleyDB datasets and was used as a reference for training and test data during the last singing voice separation campaign [4]. This dataset is composed of 150 professionally produced songs. 0nly western music genres are present, with a vast majority of pop/rock songs, along with some hip-hop, rap and metal songs. 100 songs belong to the training set and 50 to the test set.

For each song, five audio files are available: the mix, and four separated tracks (*drums*, *bass*, *vocal* and *other*). The original mix can be synthesized by directly summing the tracks of the four sources. To create the *instrumental* source, we add up the tracks corresponding to *drums*, *bass* and *others*. In our experiments, we consider both the *instrumental/vocals* dataset and the 4-stems dataset.

	MUSDB	Catalog	Bean
Diversity	150 songs	28,810 songs	24,097 songs
Quality	Separated recordings	Estimates	Separated recordings
Duration	10 hours	95 hours	79 hours
Train/val/test (%)	53/13/33	97/3/0	85/8/7

Table 1: Main characteristics of the three datasets.

2.2. Bean

In addition to MUSDB, we use a private multi-track dataset called Bean. The Bean dataset contains a majority of pop/rock songs and includes both vocal and instrumental tracks as separated recordings. Among the 24,097 available songs in this dataset, 21,597 were used for training, 2,000 for validation and the 500 remaining for test.

In total, the Bean dataset represents 5,679 different artists. We made the train/validation/test split in such a way that an artist cannot appear simultaneously in two parts of the split, as in MUSDB. This is an important precaution to ensure that the separation system will not overfit on the artists, an issue often raised in MIR [23]. We performed genre statistics on Bean, as presented in Figure 1 (green histogram). The genre distribution in Bean is mainly dominated by Pop and Rock songs, which is quite similar to MUSDB.

2.3. Catalog

To build this dataset, we took inspiration from [21], where a method is presented to build a dataset based on a music streaming catalog. We adapted this method to build a dataset from Deezer's catalog, by exploiting the *instrumental versions* that are released by some artists along with the original songs.

The first step is to find all possible *instrumental/mix* track pairs within the catalog. This matching is done using metadata and audio fingerprinting. Then, a few filtering and homogenization operations are performed: A pair is removed if both tracks have a duration difference greater than 2 seconds. Songs longer than 5 minutes are filtered out. Then, tracks within a pair are temporally re-aligned using autocorrelation. Finally, the loudness of both tracks is equalized.

To produce a triplet *(mix, instrumental, vocals)* from the pair *(mix, instrumental)*, we perform a half-wave rectified difference of both spectrograms. Eventually, 28,810 triplets were created. We split them into a training and a validation dataset, making sure that a given artist cannot appear simultaneously in both parts of the split. We refer this dataset as *Catalog A*.

Using metadata, we noticed an important genre bias towards kids music and hip-hop in this dataset compared to the genre distribution in Bean (and consequently in MUSDB), as represented in Figure 1. To overcome this issue, we built a second dataset by rebalancing the representation of each genre in a way that the final distribution matches the one of Bean. We refer to this dataset as *Catalog B*.

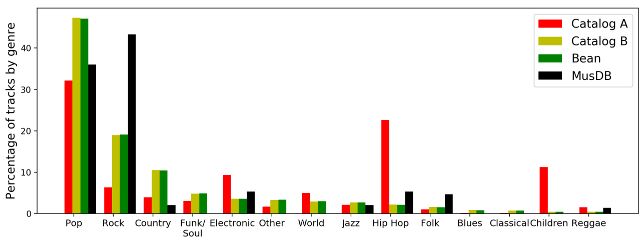


Fig. 1: Genre distribution for Bean, Catalog and MUSDB datasets.

Even though Catalog benefits from a very large volume compared to MUSDB, we must keep in mind that it was not professionally produced for separation purposes and is necessarily of a lower quality. The two main issues that we found in the dataset are:

- The half-wave rectified difference between the mix and the instrumental does not correspond exactly to the vocal part. This is because this operation is performed on magnitude spectrograms, for which source additivity is not ensured. Besides, the smallest misalignment between both tracks can produce instrumental residuals in the vocals. An informal listening test on a small subset (40 tracks) reveals that this happens in almost 50% of the tracks.
- If the metadata matching is not perfect, there may be songs with no singing voice in the mix. In this case, the *vocals* part is only a residual noise. Reversely, some *instrumental* tracks contain choirs. These cases are difficult to detect by automatic systems.

Accordingly, we may say that the Catalog database forms a large amount of weakly labeled training data. The instrumental part is professionally-produced, while the vocals are only estimates.

3. METHODOLOGY

3.1. Network architecture

In this paper, we focus on deep neural networks to perform the separation. The baseline model that we chose is the U-Net, as proposed in [5]. After some pilot experiments with other architectures (the DenseNet and MMDenseNet from [6]), we selected the U-Net, which could train in a reasonable amount of time even on large datasets. This architecture showed state-of-the-art results on the DSD100 dataset [5] and in the last SiSeC campaign [4]. It is also a simple, general architecture that can be applied in a variety of domains [24].

The U-Net shares the same architecture as a convolutional autoencoder with extra skip-connections that bring back detailed information lost during the encoding stage to the decoding stage. It has 5 strided 2D convolution layers in the encoder and 5 strided 2D deconvolution layers in the decoder.

The main modification compared to [5] was to integrate stereo processing: we used 3D tensors (channels, time steps, frequency bins) as input and output of the network. The other layers were not modified.

3.2. Data preparation

In the original datasets, all songs are stereo and sampled at 44100Hz. To reduce computational cost, we resample them to 22050Hz . We split all songs into segments of 11.88 seconds. For Catalog and Bean, we randomly select one segment from each song in the training and validation sets, avoiding the intro (first 20s) and the outro (last 20s), where vocals are often missing. We also constructed a second test dataset using 500 tracks from Bean, from which we were able to extract 1,900 segments. We made sure to balance its genre distribution over the 10 most represented genres of Figure 1. The final split proportions can be seen in Table 1.

Similarly to [5], we used Short Time Fourier Transform (STFT) as input and output features for our network. The window size is 2048 and the step size is 512. We chose these settings such that after removing the highest frequency band, the dimensions of the spectrograms are a power of 2: (channels, time steps, frequency bins) = (2, 512, 1024). This is necessary, because the network architecture that we use reduces the dimensions of the spectrograms by a factor which is a power of two.

3.3. Training

For each source (*vocals* and *instrumental*), we trained a U-Net to ouptut the corresponding magnitude spectrogram from the magnitude spectrogram of the mixture. We trained each network for 500 epochs using Keras with Tensorflow backend. We define one epoch as 800 gradient descent steps. To limit overfitting, we use the validation split of each dataset for early stopping. The training loss is the L_1 norm of the difference between the target spectrogram and the masked output spectrogram, as described in [5]. The optimizer is ADAM and the learning rate is 0.0001. The batch size is set to 1 after a short grid search.

3.4. Reconstruction

Once the training is finished, we perform an inference pass on the test dataset, equally cut into 11.88 second segments. The complex spectrograms of each source are reconstructed by computing a ratio mask from both estimates and applying it to the original mixture spectrogram. This way, the output phase is that of the mixture. The ratio mask of a source is obtained by dividing the spectrogram estimate of a source (output of the corresponding U-Net) by the sum of both the estimates. For the particular case of 4-stems separation, the *instrumental* spectrogram estimates is obtained by summing the spectrogram estimates of the 3 non-vocals stems. The STFT are inverted and full songs are reconstructed by simply concatenating the different segments. The audio is finally upsampled back to 44100Hz.

3.5. Evaluation

We use the Museval [20] toolbox to compute the standard source separation measures: Signal to Distortion Ratio (SDR), Signal to Interference Ratio (SIR) and Signal to Artifact Ratio (SAR). We aggregate these metrics using a median over all 1-second frames to keep one single metric per song and per source, as in [4]. We run the evaluation process on both the MUSDB and Bean test datasets.

To compare the performance of the different methods, we also conducted a paired Student t-test on the per songs metrics. This step was motivated by the observation that the variance was high in the metric distributions, making it sometimes difficult to assess whether a method performed significantly better than another one or not. Even though two methods may produce very similar distributions of the metrics, these metrics may vary in a dependent way (e.g. with a small but constant difference). The paired t-test helps revealing this phenomenon.

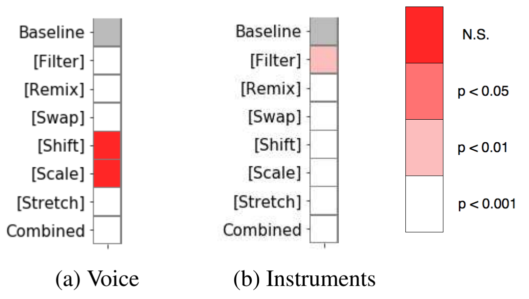


Fig. 2: Data augmentation experiment: Results of the Student's paired t-test for the SDR on the MUSDB Test dataset.

4. EXPERIMENTS AND RESULTS

4.1. Data augmentation

When training on a small dataset like MUSDB, data augmentation is regularly cited as a way to improve separation performances [8]. In this experiment, we try to figure out to what extent data augmentation can improve separation performances. For selecting data transformation to be performed, we took inspiration from [22], in which the author uses a set of transformations on the spectrograms and tests the effect on a singing voice detection task. We set up a similar set of experiments to evaluate the impact of various forms of data augmentation on separation results. We adapted the transforms proposed by Schülter (pitch shifting, time stretching, loudness modification and filtering) for source separation and added channel swapping (following [8]) and source remixing. The specificity of data augmentation in the context of source separation is that both the target and the inputs must be processed with the exact same transformation. Here is the detail of the various transformations we used:

Channel swapping [Swap]: The left and right channels are swapped with a probability of 0.5.

Time stretching [Stretch]: We linearly scale the time axis of the spectrograms by a factor $\beta_{stretch}$ and keep the central part. $\beta_{stretch}$ is drawn randomly from a uniform distribution between 0.7 and 1.3 (± 30%) for each sample. Note that this is an approximation compared to an actual modification of the speed of the audio.

Pitch shifting [Shift]: We linearly scale the frequency axis of the spectrograms by a factor β_{shift} and keep the bottom part, such that the lowest frequency band stays aligned with 0 Hz. β_{shift} is drawn randomly from a uniform distribution between 0.7 and 1.3 (± 30%) for each sample. Note that this is an approximation compared to an actual pitch shifting of the audio.

Remixing [Remix]: We remix the *instrumental* and *vocals* part with random loudness coefficients, drawn uniformly on a logarithmic scale between -9dB and $+9$dB.

Inverse Gaussian filtering [Filter]: We apply to each sample a filter with a frequency response of $f(s) = 1 - e^{-(s-\mu)^2/2\sigma^2}$ with μ randomly chosen on a linear scale from 0 to 4410Hz and σ randomly chosen on a linear scale from 500Hz to 1000Hz.

Loudness scaling [Scale]: we multiply all the coefficients of the spectrograms by a factor β_{scale}. β_{scale} is drawn uniformly on a logarithmic scale between -10dB and $+10$dB.

Combined: We perform simultaneously the channel swapping, pitch shifting, time stretching and remixing data augmentations.

Median source separation metrics (SDR, SAR, SIR) are reported in Table 2. To get an idea of the significance of the metric differences, we performed a paired Student t-test between data augmented training and the not data augmented baseline: we report p-values for this test applied to SDR on the MUSDB test set in Figure 2.

Table 2 shows that data augmentation may have a positive impact on separation metrics in some case: notably on the Bean dataset, channel swapping, pitch shifting and time-stretching seems to quite consistently improve most of the metrics. However it must be noted that even when the improvement is statistically significant for the test we performed, the improvement is very limited and hardly exceeds 0.2dB in SDR, which is very low and might not even be audible. Thus, the various data augmentation types we tested seem to have quite a low impact on separation results while being commonly used in the literature.

		Voice			Instruments		
Test	Transform	SDR	SIR	SAR	SDR	SIR	SAR
MUSDB	*Baseline*	*4.32*	*12.62*	*4.1*	*10.65*	*13.46*	*11.51*
	[Filter]	3.9	**13.35**	3.33	10.27	12.57	11.66
	[Remix]	3.75	12.89	3.6	10.45	11.81	**12.05**
	[Swap]	4.37	**13.01**	4.08	**10.69**	13.08	**11.74**
	[Shift]	4.0	**15.3**	3.5	10.58	12.46	**12.11**
	[Scale]	4.05	12.6	3.64	10.68	12.38	**11.85**
	[Stretch]	4.19	**13.44**	3.57	10.96	12.76	**12.09**
	Combined	3.76	**13.86**	3.3	10.48	12.35	**11.72**
Bean	*Baseline*	*5.91*	*9.23*	*5.73*	*9.33*	*12.43*	10.9
	[Filter]	5.58	**10.8**	5.2	9.18	11.53	10.75
	[Remix]	5.7	**10.18**	5.44	9.43	11.1	**11.4**
	[Swap]	**5.98**	**9.94**	**5.83**	**9.5**	12.25	**11.24**
	[Shift]	**6.06**	**11.53**	5.82	**9.57**	11.67	**11.63**
	[Scale]	5.87	**9.55**	5.66	9.42	11.71	**11.32**
	[Stretch]	**6.12**	**10.68**	**5.94**	**9.64**	12.18	**11.35**
	Combined	5.98	**11.45**	**5.99**	9.4	11.1	**11.07**

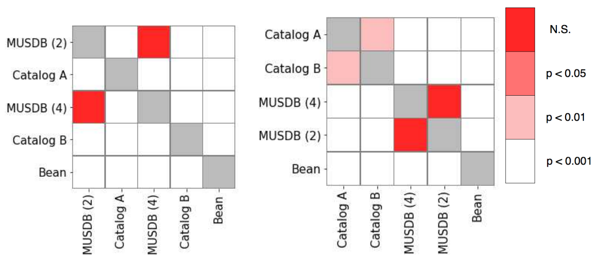
Table 2: Data augmentation experiment: Results of the U-Net trained on MUSDB with data augmentation. In bold are the results that significantly improve over the baseline ($p < 0.001$).

4.2. Impact of the training dataset

In this experiment, we evaluate the impact of the training dataset on the performances of the selected separation system. The system is trained with the 5 datasets presented in Section 2: *Catalog A*, *Catalog B*, Bean, MUSDB with two stems (*accompaniment* and *vocals*) and MUSDB with four stems (*vocals*, *drums*, *bass* and *other*). After training the system on each dataset, we evaluate its performances on the two test datasets: MUSDB and Bean. Medians over all tracks of source separation metrics are reported in Table 3 and p-values for the paired Student t-test between SDR obtained on the MUSDB test dataset are reported in Figure 3.

As expected, training on the Bean dataset yields the highest scores for most metrics on both the vocals and the accompaniment parts and on both test datasets. It is worth noting that the SDR values on the *vocals* part for the system trained on Bean are higher than the ones for all other systems by more than 1dB on the MUSDB test set and 1.5dB on the Bean test set, which is quite important (and is perceptually very noticeable). This confirms that having large datasets with clean separated tracks is a good way of improving performances of source separation systems. More surprisingly, all other training datasets provide quite similar performances from one to another. In particular, training on 4 stems instead of 2 did not improve significantly the metrics on MUSDB: then on this particular setup, adding extra information to help modelling the accompaniment spectrogram actually did not result in improved performance.

We also notice that training the system with both *Catalog* datasets has a very limited impact on the separation performances. Compared to MUSDB alone, it yields in higher SAR, but lower SIR, resulting in a similar SDR. The effect is particularly visible on the vocals. This makes sense with the way the Catalog training dataset was built: the recordings are professionally produced, so the mixture quality is good, but significant leaks remain in the vocal target. Moreover, training with *Catalog A* or *Catalog B* seems to



(a) Voice (b) Instruments

Fig. 3: Training dataset comparison experiment: Results of the Student's paired t-test for the SDR on the MUSDB Test dataset. SDR increases from top left to bottom right.

provide very similar results, which means that the difference of genre distribution between *Catalog A* and *Bean* is not responsible for the high differences of performance and the actual reason for lower performance is probably the lower quality of the separated tracks of the dataset.

Hence, training a system on a large and diverse dataset with low quality semi-automatically obtained sources seems to have a very limited impact on the performance metrics compared to using a large clean dataset such as Bean. This comes in contradiction to what was suggested in [5], where the impact of the size of the dataset was assumed to be important (even though this aspect was not tested with all other factor being fixed).

		Voice			Instruments		
Test	Train	SDR	SIR	SAR	SDR	SIR	SAR
MUSDB	MUSDB (2 stems)	4.32	12.62	4.1	10.65	13.46	11.51
	MUSDB (4 stems)	4.44	12.26	4.2	10.61	13.7	11.48
	Catalog A	4.2	7.6	**7.44**	10.47	12.84	12.03
	Catalog B	4.34	8.04	7.05	10.6	12.8	12.12
	Bean	**5.71**	**14.82**	5.19	**11.99**	**16.04**	**12.21**
Bean	MUSDB (2 stems)	5.91	9.23	5.73	9.33	12.43	10.9
	MUSDB (4 stems)	5.88	8.56	5.71	9.3	12.87	10.92
	Catalog A	5.85	7.26	7.16	9.56	11.68	12.3
	Catalog B	6.05	7.62	6.79	9.74	11.85	**12.42**
	Bean	**7.67**	**12.33**	**7.51**	**11.09**	**15.35**	12.17

Table 3: Training dataset comparison experiment: Results of the U-Net system trained on the 5 different datasets. The best results on each test dataset are displayed in bold.

5. CONCLUSION

In this study, we consider what aspects of training datasets have an impact on separation performances for a particular state-of-the-art source separation system (U-Net). In this setup, we showed that data augmentation, while quite frequently used in the literature, has a very limited impact on the separation results when performed on a small training dataset. We also showed that the extra information brought by having access to more sources than needed for performing the separation task (4 stems instead of *vocals* and *accompaniment* only) does not improve the system performances. Besides, we showed that, as opposed to what was assumed in the literature, a large dataset with semi-automatically obtained vocal sources does not help much the studied system compared to a smaller dataset with separately recorded sources. At last, we confirmed a common belief that having a large dataset with clean separated sources improves significantly separation results over a small one.

In future works, we may try to generalize these results to other state-of-the-art sources separation systems. Moreover, we focused on objective source separation metrics that are known to poorly account for perceptual differences between system. Then, assessing the impact of data with a stronger focus on the perceptual impact would be a relevant continuation of this work.

6. REFERENCES

[1] Emanuele Pollastri, "A pitch tracking system dedicated to process singing voice for music retrieval," in *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*. IEEE, 2002, vol. 1, pp. 341–344.

[2] Annamaria Mesaros, Tuomas Virtanen, and Anssi Klapuri, "Singer identification in polyphonic music using vocal separation and pattern recognition methods.," in *ISMIR*, 2007, pp. 375–378.

[3] Annamaria Mesaros, "Singing voice recognition for music information retrieval," *Tampereen teknillinen yliopisto. Julkaisu-Tampere University of Technology. Publication; 1064*, 2012.

[4] Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito, "The 2018 signal separation evaluation campaign," in *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2018, pp. 293–305.

[5] Andreas Jansson, Eric J Humphrey, Nicola Montecchio, Rachel Bittner, Aparna Kumar, and Tillman Weyde, "Singing voice separation with deep u-net convolutional networks," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2017, pp. 323–332.

[6] Naoya Takahashi and Yuki Mitsufuji, "Multi-scale multi-band densenets for audio source separation," in *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2017 IEEE Workshop on*. IEEE, 2017, pp. 21–25.

[7] Daniel Stoller, Sebastian Ewert, and Simon Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation," *arXiv preprint arXiv:1806.03185*, 2018.

[8] Stefan Uhlich, Marcello Porcu, Franck Giron, Michael Enenkl, Thomas Kemp, Naoya Takahashi, and Yuki Mitsufuji, "Improving music source separation based on deep neural networks through data augmentation and network blending," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 261–265.

[9] Daniel Stoller, Sebastian Ewert, and Simon Dixon, "Adversarial semi-supervised audio source separation applied to singing voice extraction," *arXiv preprint arXiv:1711.00048*, 2017.

[10] Naoya Takahashi, Nabarun Goswami, and Yuki Mitsufuji, "Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation," *arXiv preprint arXiv:1805.02410*, 2018.

[11] Aditya Arie Nugraha, Antoine Liutkus, and Emmanuel Vincent, "Multichannel music separation with deep neural networks," in *Signal Processing Conference (EUSIPCO), 2016 24th European*. IEEE, 2016, pp. 1748–1752.

[12] Yi Luo, Zhuo Chen, John R Hershey, Jonathan Le Roux, and Nima Mesgarani, "Deep clustering and conventional networks for music separation: Stronger together," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 61–65.

[13] Zhe-Cheng Fan, Yen-Lin Lai, and Jyh-Shing Roger Jang, "Svsgan: Singing voice separation via generative adversarial network," *arXiv preprint arXiv:1710.11428*, 2017.

[14] Pritish Chandna, Marius Miron, Jordi Janer, and Emilia Gómez, "Monoaural audio source separation using deep convolutional neural networks," in *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2017, pp. 258–266.

[15] Stylianos Ioannis Mimilakis, Konstantinos Drossos, Tuomas Virtanen, and Gerald Schuller, "A recurrent encoder-decoder approach with skip-filtering connections for monaural singing voice separation," *arXiv*, vol. 1709, 2017.

[16] Andrew JR Simpson, Gerard Roma, and Mark D Plumbley, "Deep karaoke: Extracting vocals from musical mixtures using a convolutional deep neural network," in *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2015, pp. 429–436.

[17] Chao-Ling Hsu and Jyh-Shing Roger Jang, "On the improvement of singing voice separation for monaural recordings using the mir-1k dataset," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 2, pp. 310–319, 2010.

[18] Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello, "Medleydb: A multitrack dataset for annotation-intensive mir research.," in *ISMIR*, 2014, vol. 14, pp. 155–160.

[19] Antoine Liutkus, Fabian-Robert Stöter, Zafar Rafii, Daichi Kitamura, Bertrand Rivet, Nobutaka Ito, Nobutaka Ono, and Julie Fontecave, "The 2016 signal separation evaluation campaign," in *Latent Variable Analysis and Signal Separation - 12th International Conference, LVA/ICA 2015, Liberec, Czech Republic, August 25-28, 2015, Proceedings*, Petr Tichavský, Massoud Babaie-Zadeh, Olivier J.J. Michel, and Nadège Thirion-Moreau, Eds., Cham, 2017, pp. 323–332, Springer International Publishing.

[20] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner, "The MUSDB18 corpus for music separation," Dec. 2017.

[21] Eric Humphrey, Nicola Montecchio, Rachel Bittner, Andreas Jansson, and Tristan Jehan, "Mining labeled data from web-scale collections for vocal activity detection in music," in *Proceedings of the 18th ISMIR Conference*, 2017.

[22] Jan Schlüter, *Deep Learning for Event Detection, Sequence Labelling and Similarity Estimation in Music Signals*, Ph.D. thesis, Johannes Kepler University Linz, Austria, July 2017, Chapter 9.

[23] Arthur Flexer, "A closer look on artist filters for musical genre classification," *World*, vol. 19, no. 122, pp. 16–17, 2007.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

AUDIO-BASED DETECTION OF EXPLICIT CONTENT IN MUSIC

Andrea Vaglio[†⋆] *Romain Hennequin*[†] *Manuel Moussallam*[†]
Gaël Richard[⋆] *Florence d'Alché-Buc*[⋆]

[†]Deezer Research & Development
[⋆] LTCI, Télécom Paris, Institut Polytechnique de Paris
research@deezer.com

ABSTRACT

We present a novel automatic system for performing explicit content detection directly on the audio signal. Our modular approach uses an audio-to-character recognition model, a keyword spotting model associated with a dictionary of carefully chosen keywords, and a Random Forest classification model for the final decision. To the best of our knowledge, this is the first explicit content detection system based on audio only. We demonstrate the individual relevance of our modules on a set of sub-tasks and compare our approach to a lyrics-informed oracle and an end-to-end naive architecture. The results obtained are encouraging with a F1-score of 67% on a industrial scale explicit content dataset.

***Index Terms*—** Explicit content detection, keyword spotting, lyrics transcription, CTC training, music information retrieval

1. INTRODUCTION

For over three decades [1], a parental advisory label has been found on musical recordings when they include explicit content (*e.g.* lyrics potentially unsuitable for children). As of today, this labelling is mainly done manually following guidelines [2]. This process is slow and hard to scale to industrial-size catalogs. Existing automatic approaches are scarce and rely on the availability of the lyrics in text format.

Lyrics transcriptions could be obtained from audio using singing voice recognition algorithms. However, although *Automatic Speech Recognition* (ASR) methods have recently shown impressive progress [3], singing voice transcription still raises challenging issues [4]. First, in comparison to speech, singing voice characteristics are often more varied. The pitch, the pronunciation and the vowels duration can fluctuate greatly. Second, the accompaniment can be considered as a highly correlated noise with a level comparable to the signal of interest. A pre-processing step of voice separation is generally used to improve results [5], although still not on par with those obtained on a capella singing [6]. Third, until recently [7], no open dataset was available to train statistical models at scale.

When lyrics are available, explicit content detection can be approximately achieved through assessing the presence of words from a fairly small specialized dictionary. In fact, as proven in [8], state-of-the-art deep neural network algorithms perform just slightly better than dictionary-based methods with suitable keywords. This suggests that detecting a set of carefully chosen keywords directly on the audio signal is a good proxy to perform the explicit detection task in the general case. Keyword Spotting in audio is an actively studied task, achieving high performances on speech signals [9]. A few attempts to transfer them to singing voice have been proposed [10, 11]. One existing approach relies on a keyword-filler *Hidden Markov Model* (HMM) algorithm [11]. This method presents several limitations. First, expert knowledge is required for the tedious task of creating a pronunciation dictionary. Second, model training requires synchronized annotations, at the phoneme level, between audio and text. Since no readily accessible dataset exists for polyphonic music, such annotations are generated with an acoustic model trained on speech, by aligning textual lyrics to music, leading to sub-optimal model performance.

In this paper, we address the task of explicit musical content detection from audio only. Namely, given a piece of music, we aim at classifying an audio recording as either *explicit* or *non-explicit*. Our proposed work is, to the best of our knowledge, the first audio-based detection system of explicit content in music. Our approach is based on an *Audio-To-Character* (A2C) recognition model recently proposed for singing voice transcription [12] and a keyword spotting model associated with a dictionary of carefully chosen keywords in relation to the explicit detection task. In [13], authors have demonstrated the better performance of this architecture over the keyword filler. To the best of our knowledge, it is the first time that such an architecture is used for *Keyword Spotting* (KWS) on singing voice. The key advantages of this method are that it requires no expert knowledge and is usable with unsynchronized annotations. The decoding is directly performed on the output of the A2C and, contrary to end-to-end KWS approach like in [14], new keywords can be added easily to the dictionary without retraining the model. Finally,

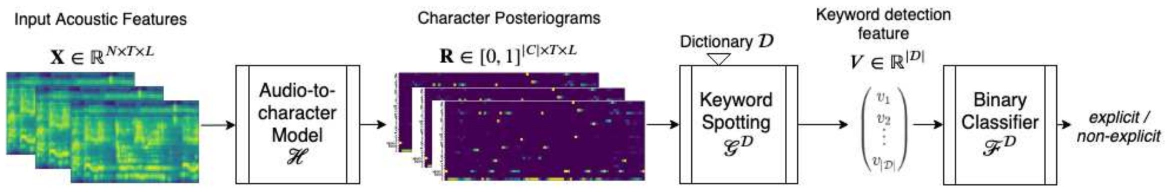

Input Acoustic Features
$\mathbf{X} \in \mathbb{R}^{N \times T \times L}$
Audio-to-character Model $\mathscr{H}$
Character Posteriograms
$\mathbf{R} \in [0,1]^{|C| \times T \times L}$
Dictionary $\mathcal{D}$
Keyword Spotting $\mathscr{G}^{\mathcal{D}}$
Keyword detection feature
$V \in \mathbb{R}^{|\mathcal{D}|}$
v_1 v_2 $v_{|\mathcal{D}|}$
Binary Classifier $\mathscr{F}^{\mathcal{D}}$
explicit / non-explicit

Fig. 1. General overview of the proposed modular explicit content detection system

the *explicit* label is inferred by a binary classifier using the output of the keyword spotting system. In this study, we restrict ourselves to recordings with English lyrics.

2. PROPOSED METHOD

A general overview of our system is given in Figure 1. A monophonic song S is sliced into L segments of equal size. It is worth noting that L may be different for each song since it depends on the song duration. The system takes as input an acoustic feature tensor $\mathbf{X} \in \mathbb{R}^{N \times T \times L}$ built from S, with T the number of temporal frames per segment and N the features dimension. Given a dictionary $\mathcal{D}$, the predictive model $\mathscr{L}^{\mathcal{D}}$ is the composition of three modules:

$$\mathscr{L}^{\mathcal{D}}(\mathbf{X}) = \mathscr{F} \circ \mathscr{G}^{\mathcal{D}} \circ \mathscr{H}(\mathbf{X}) \tag{1}$$

For a given input tensor $\mathbf{X}$, the audio-to-character module $\mathscr{H}$ outputs a 3D-tensor $\mathbf{R}$. Given X_ℓ, the matrix of acoustic features extracted from the ℓ^{th} segment, each coefficient $r_{i,j,\ell}$ provides an estimation of the posterior probability of c_i, the i^{th} character being uttered at t_j, the j^{th} frame. :

$$r_{i,j,\ell} = h_{i,j}(X_\ell) = \hat{P}(c_i, t_j \mid X_\ell), \tag{2}$$

for $1 \leq i \leq |C|$, $1 \leq j \leq T$, $1 \leq \ell \leq L$, C being the set of characters outputted by the model. It consists of 26 lowercase letters of the Latin alphabet, plus the word-boundary "space" token, the instrumental token "I", the apostrophe and the CTC blank token ϵ introduced in section 2.2. Note that $\mathscr{H}$ relies on the design of $h : \mathbb{R}^{N \times T} \rightarrow [0,1]^{|C| \times T}$, that takes a segment represented by an acoustic feature matrix and predicts a posteriogram on characters.

For a given input tensor $\mathbf{R} = \mathscr{H}(\mathbf{X})$, a vector V is outputted by the keyword spotting module $\mathscr{G}^{\mathcal{D}}$, whose each coefficient v_n gives an estimate of the (log) posterior probability of k_n, the n^{th} keyword of the dictionary $\mathcal{D}$, by averaging on all segments:

$$v_n = \underset{\ell=1,\dots,L}{\text{mean}} \log \hat{P}(k_n \mid R_\ell), \tag{3}$$

where R_ℓ is the ℓ^{th} matrix in tensor $\mathbf{R}$.

For a given input vector $V = \mathscr{G}^{\mathcal{D}} \circ \mathscr{H}(\mathbf{X})$, an *explicit* label $\mathscr{L}^{\mathcal{D}}(\mathbf{X})$ is outputted by the binary classifier $\mathscr{F}$ discriminating the content of the song S as *explicit* or *not-explicit*. Among these three modules, only modules $\mathscr{H}$ and $\mathscr{F}$ require to be learned. Learning $\mathscr{H}$ boils down to learning model h which can be done using a dataset of annotated segments $\{(X^i, u^i)_{i=1}^{n_{seg}}\}$ where X^i is a matrix of acoustic features describing a segment and u^i is the corresponding sequence of characters. Learning $\mathscr{F}$ requires to apply the pre-processing $\mathscr{G}^{\mathcal{D}} \circ \mathscr{H}$ to the training dataset $\{(\mathbf{X}^i, y^i)_{i=1}^{n_{songs}}\}$ containing songs annotated by *explicit/non explicit* labels.

2.1. Audio-to-character recognition

The model h on which A2C module $\mathscr{H}$ relies is implemented with a *Convolutional Recurrent Neural Network* (CRNN) trained with a *Connectionist Temporal Classification* (CTC) algorithm. The *Recurrent Neural Network* (RNN) CTC has been successfully applied to ASR [3]. To reduce the dimension of features and accelerate training, we use additional convolutional layers. For RNN layers, we choose bidirectional *Long Short-Term Memory* (LSTM), so that outputs at each frame depend of the entire input sequence [15].

The CTC algorithm [16] allows to train RNN models without aligned annotations. To do that, a "blank" symbol (noted ϵ) is introduced to represent a non-emission token. Any character, including ϵ, can be emitted at each frame by the model. The total probability of the output character sequence is maximized using the CTC algorithm by marginalizing over all possible alignments for a given input. The objective function being differentiable, the network is trained with back-propagation through time. More details for CTC algorithm are given in [16].

2.2. Keyword spotting

Following the work of [13], we implement $\mathscr{G}^{\mathcal{D}}$ as a CTC-based decoding function. For a given searched keyword k, we consider k' which is the keyword k with an ϵ at the beginning, end, and between every character to allow the use of ϵ during decoding. A decoding network of size $|k'| \times T$ is constructed from k'. The goal of the decoding function is to find the path in the network that maximize the CTC scoring for the keyword k'. To do that, we define network's node $\alpha_{s,j}$ as the CTC score of the sub-sequence $k'_{1:s}$ after j frame. A forward-backward algorithm can be used to compute efficiently $\alpha_{s,j}$ scores, by merging together paths that reach the same node. $\alpha_{s,j}$ is then computed recursively from α's of the previous frame. Only transitions between blank and non-blank characters, and between pair of distinct non-blank characters are allowed. As ϵ at the beginning and end of the sequence is optional, there are two valid starting nodes and two final nodes. The coefficients α's are initialized as follows:

$$\alpha_{s,0} = P(k'_s, t_0 \mid X_\ell) \text{ for } s \in \{0,1\} \text{ and } \alpha_{s,0} = 0, \forall s > 1$$

Recursion is given by:

$$\begin{aligned} \alpha_{s,j} &= (\sum_{\tau=0}^{\tau=1} \alpha_{s-\tau,j-1}) P(k'_s, t_j \mid X_\ell), \text{if } k'_s \in \{\epsilon, k'_{s-2}\} \\ \alpha_{s,j} &= (\sum_{\tau=0}^{\tau=2} \alpha_{s-\tau,j-1}) P(k'_s, t_j \mid X_\ell), \quad \text{otherwise} \end{aligned} \tag{4}$$

Finally, keyword probability is given at each step j by:

$$P(k, t_j) = \alpha_{|k'|-1,j} + \alpha_{|k'|-2,j} \tag{5}$$

We consider the detection score s of the keyword k to be the maximum of the keyword probability over all time step:

$$s(k) = \max_{j=1,\ldots,T} P(k, t_j) \tag{6}$$

We found empirically that with these computation rules, we can only find keywords at the beginning of segments. In practice, keyword probabilities after the first sung word in the recording are artificially low. To prevent this and allow the keyword detector to be fired at any time, we choose to reinitialize the first node at each time step:

$$\alpha_{0,j} = P(k'_0, t_j \mid X_\ell) \tag{7}$$

As naive CTC scoring is numerically unstable, computations are done in log-space using the log-sum-exp trick [17].

3. EXPERIMENTS

3.1. Recognition model

The main component h of the A2C model $\mathscr{H}$ is trained with the DALI dataset introduced in [7]. DALI contains 5358 audio tracks with time-aligned lyrics at paragraph, line and word levels. It is composed of varied western genres (*e.g.* rock, rap and electronic). Tracks are downsampled to 16 kHz and converted to mono. Vocals from each song are then extracted using Spleeter [18]. For each song, training samples are generated by segmenting the track using a window of 5 seconds with a hop size of 2.5 seconds. The character sequence associated with a segment is created by concatenating all words whose start position are within the segment. In case no words start within a segment, we generate a token "I" (for "Instrumental"). Character sequences are transformed to fit the set of characters C outputted by the model: each character sequence is converted to lower-case and non-valid characters are discarded. Finally, we make an artist aware split [19] between train, validation and test dataset of 70%-1%-14%. We respectively obtain datasets of 384, 5 and 63 hours of music.

The model h is composed of 2 convolutional layers, followed by 3 layers of bidirectional LSTM and a dense layer. For each input sample, values of mel-scale log filterbanks coefficient and energy plus deltas and double-deltas are extracted. A Hann window of 32 ms with a step size of 16 ms is used. For the convolutional part, the size of filters and max-pooling are respectively 3×3 and 2×3. The number of features map of each layer is 32. Each recurrent layer has a dimension equal to 256. A dropout of 0.1 is applied between recurrent layers. The last layer is a single affine transformation followed by a softmax function which outputs the probabilities of characters from vocabulary C. The model is trained using the CTC loss implementation of [20]. The loss is minimized using ADAM with a learning rate of 10^{-4}, a batch size of 32, 4000 training epochs with 250 steps per epoch. We use validation-based early stopping. For transcription, classic beam search decoding [15] is used, using a beam width of 100.

Finally, we obtain a *Character Error Rate* (CER) of 47.41% on the test dataset. This is on par with results reported by [12] on a different dataset. An example of posteriogram R_ℓ inferred using the trained network is pictured in Figure 2. R_ℓ consists of a sequence of spikes, associated with detected characters, separated by ϵ character.

3.2. Keyword spotting

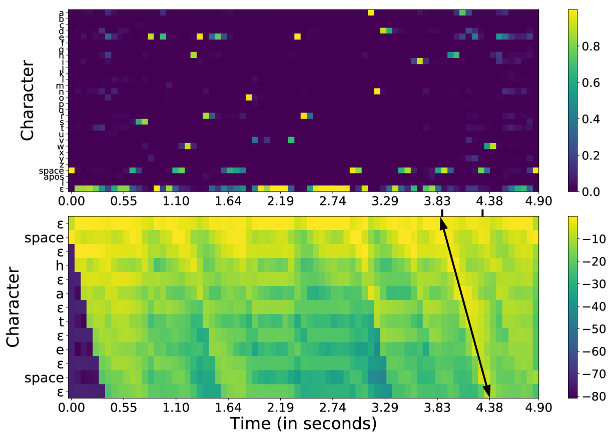


Fig. 2. A positive sample for keyword "hate". Top: Posteriogram R_ℓ inferred by acoustic model $\mathscr{H}$. Bottom: Decoding matrix composed of coefficients α. Ground truth: "to see we're over and i hate when". Transcription with beam search: "e se where over and i hae we". Decoding line and ground truth position of keyword are displayed in the figure

Following the work of [21] we select the keywords of $\mathcal{D}$

Metrics	Audio baseline	Our system	Lyrics baseline
Precision	.61 (.02)	.63 (.02)	.65 (.02)
Recall	.59 (.02)	.71 (.02)	.84 (.02)
F1-score	.60 (.02)	.67 (.02)	.73 (.02)

Table 1. Results for explicit detection task on the test set (standard deviation in parenthesis)

based on the explicit and not-explicit lyrics word distributions. To generate $\mathcal{D}$, we use the importance I defined in [8]. For a chosen keyword, I is computed as the ratio between frequency of the word in explicit and non-explicit lyrics. As in [8], we manually discard stop words, too common words, too rare words, onomatopoeia and abbreviations. The dictionary is constructed with 128 words with the highest importance.

Performance of decoding with dictionary $\mathcal{D}$ are assessed on DALI test set. 75% of $\mathcal{D}$ keywords, with at least one occurrence in the test dataset, have a *Area Under Curve* (AUC) of *Receiver Operating Characteristic* (ROC) curve greater than 0.81. Being the first time such metrics are computed for keyword spotting in the singing case, we cannot compare it to other results. Since these values are significantly higher than random, the feature vector V carry some information on the presence of keywords in $\mathcal{D}$. An example of decoding is displayed in Figure 2. The example is "positive", as the searched keyword is indeed present in the ground truth character sequence. A decoding line is visible in the figure. Position of "space" character delimiting the decoding line (3.94s to 4.28s) are quite close to the ground truth position of the word (3.9s to 4.29s). This result suggests that the acoustic model $\mathscr{H}$ correctly uses the "space" character and is able to find words position. This is consistent with results found for lyrics-to-audio alignment [12].

3.3. Explicit lyrics detection

To train $\mathscr{F}$, we use a private dataset. Songs are either labelled *explicit* or *non-explicit*. We discard tracks also present in DALI to avoid overfitting. We notice that the music genre distribution of explicit tracks and non-explicit tracks, is very different: rap is strongly over-represented in explicit tracks (40% of all tracks), but not in the non-explicit ones (few percents). To avoid creating an explicit content detection that rely mostly on the genre information, we sample both explicit and non-explicit tracks to obtain same genre distribution for the two types of songs. The complete dataset then consists of 2600 non-explicit tracks and 2530 explicit ones. Finally, we make an artist aware split [19] between training, validation, test of respectively 70%, 15%, 15%. We create another dataset the same way for dictionary creation. The dataset consists of 24250 non-explicit tracks and 24250 explicit ones. No songs are common between the two datasets.

Our model is compared to two baseline systems. The first one is a classic CRNN audio classifier. This architecture was successfully used in a variety of music classification tasks, such as genre recognition [22] or music emotion recognition [23]. Unlike our system, this classifier tries to directly infer *explicit* labels from audio in an end-to-end manner. The model is composed of 4 convolutional layers, followed by 1 gated recurrent units layer and a dense layer. For each input sample, values of mel-scale log filterbanks coefficient are extracted using a Hann Window of 48 ms with a step size of 48 ms. The model is trained using a binary cross-entropy loss which is optimized using an Adadelta optimizer and a batch size of 1. The model is trained for 3000 epochs with 450 steps per epoch. We use validation-based early stopping. The second baseline is a dictionary lookup based on lyrics as in [8]. Given the dictionary $\mathcal{D}$, this method classifies a song as *explicit* if its lyrics contain at least one of the keywords in $\mathcal{D}$ and as *non-explicit* otherwise. Unlike our system, this baseline is informed by lyrics at test time. As such, this baseline can be considered as an oracle (*e.g.* providing an upper bound for performance) for our task of detecting explicit content from audio only.

For $\mathscr{F}$, we use a Random Forest classifier [24]. Hyperparameters of the classifier are tuned using a first step of random search and a second step of grid search. Number of keywords of the dictionary, for our model and for the lyrics baseline, are tuned on the validation dataset. We report precision, recall and F1-score for the explicit class. Since explicit content might be sensitive to certain audiences, emphasis is put, at highest F1-score, on the system maximizing the recall. We use this rule to choose our "best" parameters on the validation dataset. The "best" number of keywords is 128 for the lyrics baseline and 32 for our model. Baselines and our system with their "best" parameters are evaluated on the test dataset. Results are reported in Table 1. Scores reached by the lyrics baseline are similar to those found in [8]. Performance of a naive audio baseline on this challenging task is significantly outperformed by our modular approach. While yet not equivalent to a lyrics-informed scenario, these results are encouraging and show the validity of the proposed method. Performances of these systems are still insufficient to be deployed without human oversight. In [8], authors argue that explicit detection is an inherently hard task. They propose using these systems as tools to help annotators making the final labelling.

4. CONCLUSION

We address the novel task of explicit musical content detection from audio only. Despite the task being challenging, our proposed modular approach yield promising results. Moreover, the system's decisions can be explained in terms of specific keyword presence probability which is a desirable property given the sensitivity of the task. Future works will investigate keyword decoding augmentation with a character level language model as in [25].

5. REFERENCES

[1] National Public Radio, "'parental advisory' labels — the criteria and the history," `https://www.npr.org/sections/therecord/2010/10/29/130905176/you-ask-we-answer-parental-advisory---why-when-how`, 2019, [Online; accessed October 18, 2019].

[2] British Phonographic Industry, "Parental advisory guidelines," `https://www.bpi.co.uk/media/1047/parental-advisory-guidelines.pdf`, 2019, [Online; accessed October 18, 2019].

[3] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al., "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International Conference on Machine Learning (ICML)*, 2016, pp. 173–182.

[4] Anna Mesaros, *Singing Voice Recognition for Music Information Retrieval*, Ph.D. thesis, Tampere university of technology, 2012.

[5] Annamaria Mesaros, Tuomas Virtanen, and Anssi Klapuri, "Singer identification in polyphonic music using vocal separation and pattern recognition methods," in *In Proc. IEEE International Society for Music Information Retrieval Conference (ISMIR)*, 2007, pp. 375–378.

[6] Annamaria Mesaros and Tuomas Virtanen, "Automatic recognition of lyrics in singing," *EURASIP Journal on Audio, Speech, and Music Processing*, 2010.

[7] Gabriel Meseguer-brocal, Alice Cohen-hadria, and Geoffroy Peeters, "Dali : a Large Dataset of Synchronized Audio , Lyrics and Notes , Automatically Created Using Teacher-Student Machine Learning Paradigm," in *IEEE International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[8] Michael Fell, Elena Cabrio, Michele Corazza, and Fabien Gandon, "Comparing Automated Methods to Detect Explicit Content in Song Lyrics," in *Recent Advances in Natural Language Processing (RANLP)*, Varna, Bulgaria, Sept. 2019.

[9] Raphael Tang and Jimmy Lin, "Deep residual learning for small-footprint keyword spotting," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5484–5488.

[10] Hiromasa Fujihara, Masataka Goto, and Jun Ogata, "Hyperlinking lyrics : A method for creating hyperlinks between phrases in song lyrics," *Proceedings of the International Conference on Music Information Retrieval*, pp. 281–286, 2008.

[11] Anna Marie Kruspe, *Application of Automatic Speech Recognition Technologies to Singing Doctoral Thesis*, Ph.D. thesis, University Fraunhofer, apr 2018.

[12] Daniel Stoller, Simon Durand, and Sebastian Ewert, "End-to-end Lyrics Alignment for Polyphonic Music Using an Audio-to-Character Recognition Model," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[13] Kyuyeon Hwang, Minjae Lee, and Wonyong Sung, "Online Keyword Spotting with a Character-Level Recurrent Neural Network," *Arxiv*, pp. 1–10, 2015.

[14] Guoguo Chen, Carolina Parada, and Georg Heigold, "Small-footprint keyword spotting using deep neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. may 2014, pp. 4087–4091, IEEE.

[15] Alex Graves and Navdeep Jaitly, "Towards End-To-End Speech Recognition with Recurrent Neural Networks," *Internation Conference on Machine Learning (ICML)*, vol. 32, no. 1, pp. 1764–1772, 2014.

[16] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," *ACM/IEEE Supercomputing Conference (SC)*, vol. 148, pp. 369–376, 2006.

[17] Awni Hannun, "Sequence modeling with ctc," *Distill*, vol. 2, no. 11, pp. e8, 2017.

[18] Romain Hennequin, Anis Khlif, Felix Voituret, and Manuel Moussallam, "Spleeter: A fast and state-of-the art music source separation tool with pre-trained models," Late-Breaking/Demo ISMIR 2019, November 2019, Deezer Research.

[19] Arthur Flexer, "A closer look on artist filters for musical genre classification," *World*, vol. 19, no. 122, pp. 16–17, 2007.

[20] Yann Soullard, Cyprien Ruffino, and Thierry Paquet, "CTC-Model: Connectionist Temporal Classification in Keras," 2018.

[21] Jayong Kim and Y Yi Mun, "A hybrid modeling approach for an automated lyrics-rating system for adolescents," in *European Conference on Information Retrieval (ECIR)*. Springer, 2019, pp. 779–786.

[22] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho, "Convolutional recurrent neural networks for music classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 2392–2396.

[23] Miroslav Malik, Sharath Adavanne, Konstantinos Drossos, Tuomas Virtanen, Dasa Ticha, and Roman Jarina, "Stacked convolutional and recurrent neural networks for music emotion recognition," *arXiv preprint arXiv:1706.02292*, 2017.

[24] Tin Kam Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*. IEEE, 1995, vol. 1, pp. 278–282.

[25] Yanzhang He, Rohit Prabhavalkar, Kanishka Rao, Wei Li, Anton Bakhtin, and Ian McGraw, "Streaming small-footprint keyword spotting using sequence-to-sequence models," in *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 474–481.

MULTILINGUAL LYRICS-TO-AUDIO ALIGNMENT

Andrea Vaglio[12] **Romain Hennequin**[1] **Manuel Moussallam**[2]
Gaël Richard[2] **Florence d'Alché-Buc**[2]
[1] Deezer R&D
[2] LTCI, Télécom Paris, Institut Polytechnique de Paris
research@deezer.com

ABSTRACT

Lyrics-to-audio alignment methods have recently reported impressive results, opening the door to practical applications such as karaoke and within song navigation. However, most studies focus on a single language - usually English - for which annotated data are abundant. The question of their ability to generalize to other languages, especially in low (or even zero) training resource scenarios has been so far left unexplored. In this paper, we address the lyrics-to-audio alignment task in a generalized multilingual setup. More precisely, this investigation presents the first (to the best of our knowledge) attempt to create a language-independent lyrics-to-audio alignment system. Building on a *Recurrent Neural Network* (RNN) model trained with a *Connectionist Temporal Classification* (CTC) algorithm, we study the relevance of different intermediate representations, either character or phoneme, along with several strategies to design a training set. The evaluation is conducted on multiple languages with a varying amount of data available, from plenty to zero. Results show that learning from diverse data and using a universal phoneme set as an intermediate representation yield the best generalization performances.

1. INTRODUCTION

Lyrics-to-audio alignment aims at synchronizing lyrics text units such as paragraphs, lines or words to the timed position of their appearance in the audio signal. Tools dedicated to this task have many practical applications: they can be applied to generate new annotated data to train more robust singing voice recognizers [1]; or be used as building blocks in specific applications such as karaoke [2], navigation within songs [3] or explicit lyrics removal [4]. Lyrics alignment methods are typically inspired from text-to-speech methods. Although text-to-speech alignment is a mature [5] and widely studied task [6], lyrics-to-audio alignment remains a challenging problem with specific limitations. First, the musical accompaniment acts as a loud background "noise", potentially highly correlated with the signal of interest since vocalists usually sing in harmony and rhythm with instruments. A singing voice separation algorithm pre-processing step is often used to partially overcome this problem [7]. Second, singing voice exhibits more variety than speech with potentially large phonemes pronunciation variations between songs and extended tessitura. Recent studies have proposed efficient alignment methods for singing voice [8,9], but only for the English language, for which annotated data is abundant. The question of their ability to generalize to other languages, especially in low (or even zero) training resource scenarios, has not been properly addressed.

Arguably a monolingual evaluation is unrepresentative of the variety of music recordings available in large scale collections. Commercial streaming services commonly serve content in hundreds of languages and a non-negligible number of popular songs even have multilingual lyrics [10]. However, annotated data on this type of content are scarce. A source of inspiration comes from the related field of multilingual speech recognition [11]. Transfer learning methods [12] have been shown to improve performance on language with few to zero training data. However, this improvement on low-resource languages can sometimes be detrimental to performances on languages with more resources [11].

The goal of this paper is to evaluate and extend state of the art lyrics-to-audio alignment methods to a language-independent setup. First, we review the fitness of these systems to the multilingual framework. Then, we focus on one architecture and study two key features likely to allow generalization to several languages: 1) the intermediate representation space (character versus phoneme) and 2) the design of the training dataset. Evaluation is performed on multiple languages with various amounts of data available, from plenty to zero. The paper is organised as follows: related works are presented in Section 2. We then describe the proposed method in Section 3. The experimental setup and results are described respectively in Section 4 and Section 5. Finally, conclusions are drawn in Section 6 and future works are discussed.

2. RELATED WORKS

Singing voice alignment methods are typically inspired from text-to-speech alignment systems. Classically, an acoustic model is trained and used to force text to audio alignment using a Viterbi algorithm [5]. These models are usually trained using alignment annotations, at the frame level, between audio and text. However, the availability of such annotations is very limited for polyphonic music where they are traditionally generated by employing an intermediate model [1], leading to suboptimal performances [8]. More generally, the development of such approaches for singing voice was slowed down by the lack of publicly available annotated dataset at word or even line level. Some models were trained on speech and adapted to singing using speaker adaptation technique and a small singing dataset. For instance, in [13], a monophone *Hidden Markov Model* (HMM) is trained on speech and adapted on a small corpus of manually annotated *a cappella* songs with *Maximum Likelihood Linear Regression* (MLLR). The alignment is then performed on polyphonic songs after extracting the singing voice with a melody transcription and a sinusoidal modeling technique. Other models were trained with "low quality" automatic annotations generated with forced alignment using an *Automatic Speech Recognition* (ASR) system. In [1], a speech recognizer is used to generate a large amount of singing annotations by aligning a large corpus of *a cappella* singing to their corresponding lyrics. Annotations are then used to train a new acoustic model. This new model is used to align 19 vocal tracks from English language pop songs: the phoneme sequence is estimated for each track and its Levenshtein distance to the ground truth sequence from the lyrics is computed to find the alignment path. To help alignment, multiple approaches tried extending speech recognizers with external information such as chords [14], score [15] or note onsets [16].

The recent release of the DALI dataset [17] has led to significant progress in lyrics-to-audio alignment. This dataset is the first publicly annotated singing voice dataset available. It contains 5358 audio tracks with time-aligned lyrics at paragraph, line and word levels. These annotations are created from manual annotations and are considered to be very good. It is composed of varied western genres (*e.g.* rock, rap and electronic) in several languages. Novel singing voice separation algorithms displayed impressive results [18] and were also shown to improve significantly lyrics-to-audio alignment systems performances [7]. State-of-the-art approaches for lyrics alignment were compared in the MIREX 2019 challenge [1] . Two submitted systems showed particularly strong performances. The first one was SDE2, described in [8]. It is based on an end-to-end audio-to-character architecture, more precisely a wave-U-net. A preprocessing step of singing voice separation is performed, during training and inference, using a U-net convolutional network. The acoustic model is trained on a private English dataset of 40000 songs using a CTC algorithm. The second one was GYL1, described in [9], which obtained the best results on the challenge. It is based on a *Time Delay Neural Network* (TDNN) which is trained on the English subpart of the DALI dataset. It uses an extended lexicon to cope with long vowels duration in singing and genre labelling information (phoneme units are annotated with genres) but does not rely on a preprocessing step of singing voice separation.

Although it achieved the best performances in the MIREX challenge, GYL1 can not be straightforwardly used in a multilingual setup: it is composed of multiple parts, some of them, such as the pronunciation dictionary and the language model, being specific to English. To be able to use it on a new language, it would require to modify, or retrain, these parts. In comparison, SDE2 is based on an end-to-end acoustic model, trained with CTC algorithm, that directly outputs characters. It is more suitable to perform multilingual lyrics-to-audio alignment as it can be theoretically applied to any languages being based on the same script (writing system) as the training language.

Employing characters may not be optimal for multilingual lyrics-to-audio alignment: [8] suggest that using phoneme as an intermediate representation could be more relevant for aligning song in other languages. They argue that, for phoneme based systems, only the pronunciation dictionary has to be replaced for a new language, while a character based system is limited by the set of characters that the acoustic model outputs. For instance, SDE2 can only be used to align songs in Latin script languages. The output of the acoustic model could be extended with characters from scripts of new languages, as in [19], but it would require retraining the acoustic model each time a new script is added in the language pool. Using phoneme as an intermediate representation, any language can be theoretically aligned for any trained model if a pronunciation dictionary is available. In this work, we study a system inspired from [8] using either a character or a phoneme intermediate representation.

3. PROPOSED METHOD

A general overview of the proposed system is described in Figure 1. It is composed of three parts: a singing voice separation model, an acoustic model and a lyrics-to-audio alignment procedure. It takes as input a song x, its corresponding lyrics y and output the synchronized lyrics $\hat{y}$. Vocals are extracted from the song using a singing voice separation module. The acoustic model processes features extracted from the isolated vocal signal. The acoustic model consists in an RNN trained with a CTC algorithm. The set of outputs of the acoustic model is either characters of the Latin alphabet or phonemes of an universal phoneme set. Lyrics-to-audio alignment is performed on outputs of the acoustic model by a CTC-based alignment decoding function.

3.1 Acoustic model

The acoustic model of our system is a RNN trained with a CTC algorithm. CTC-based acoustic models were successfully used for multilingual speech recognition [19,20]. The RNN part is composed of bi-directional *Long Short-*

[1] `https://www.music-ir.org/mirex/wiki/2019:Automatic_Lyrics-to-Audio_Alignment`

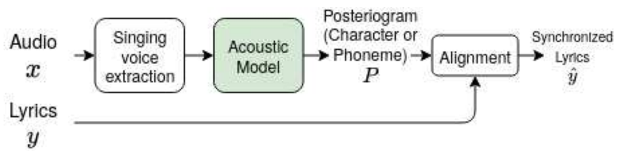


Figure 1. Overview of the lyrics-to-audio alignment system. Our study focuses on the training of the acoustic model (section 3.1) and the design of intermediate posteriogram representation spaces (section 3.2). The alignment block is described in section 3.3

Term Memory (LSTM) layers. Authors in [21] argue that such models can give reliable alignments given that outputs at each frame depend on the entire input sequence. In contrast, uni-directional CTC acoustic model suffers from alignment delay [22].

CTC makes it possible to directly train RNN models using weakly aligned annotations, e.g. at word or line level. To do that, the CTC algorithm introduces a new symbol called "blank" (noted ϵ) which represent a non-emission token. The total probability of the output label sequence is then marginalized over all possible alignment for a given input. In our case, the output label sequence is a sequence of character or phoneme. Since the objective function is differentiable, the network can be trained with standard back-propagation through time. The CTC algorithm is more extensively described in [23].

3.2 Character vs Phoneme

We consider two different intermediate representations for our architecture. The first one is a character set, here the Latin alphabet. This representation does not need any kind of expert linguistic knowledge as the acoustic model directly outputs characters probability. However, such a representation is not suitable to perform alignments of songs in a language with a different script. To process those, the acoustic model would need to be retrained with new data on the given script. Moreover, even for languages sharing the same script, a character-based representation is suboptimal for transferring knowledge between languages, as characters pronunciation can significantly differ from one language to another. Our approach relies on the following remarks: all languages share some common phonemes and phonemes are considered to be language independent [24], i.e. to be pronounced the same way across languages. Therefore, using a universal phoneme set as an intermediate representation makes it possible to leverage similarity between sounds across languages. The idea is to use consistent phonemes across languages used for training and that most phonemes from an unseen language appear in the languages used for training.

It can be achieved using *international phonetic alphabet* (IPA) symbols. The IPA is a set of phonetic notations which is a standardized representation of sounds of all spoken language. IPA Pronunciations of words from all languages can be obtained using *Grapheme-To-Phoneme* (GTP) tools. Such tools are available for most common languages. The universal phoneme set is created by concatenating and merging the union of phoneme sets of all languages based on their IPA symbols.

3.3 Lyrics to audio alignment

In order to align a song to its corresponding lyrics y, the audio is sliced into segments of 5 seconds with a step size of 2.5 seconds. A posteriogram is generated by the trained acoustic model for each segment. To obtain the final posteriogram, all segments posteriograms are concatenated, cropped to half of their duration centered in their middle. We obtain a posteriogram $P \in [0,1]^{|C| \times T}$, C being the set of symbols supported by our acoustic model, either characters of phonemes, and T the number of temporal frames of the song. This matrix provides an estimation of the posterior probabilities of each symbol through time. Alignment annotations are then predicted, using the generated posteriogram P and lyrics y, with a CTC-based alignment function inspired from the CTC-based decoding function presented in [25] and is akin to a Viterbi forced alignment [26]. Viterbi forced alignment is a simpler version of Viterbi decoding where the possible paths are limited to the lyrics symbol sequence. To allow the use of ϵ during decoding, y is extended to z by adding a ϵ at the beginning, end, and between every unit. A decoding network of size $|z| \times T$ is then constructed from z. The goal of the decoding function is to find the path in the network that give the most probable alignment $\hat{y}$ of y given the posteriogram P. More precisely:

$$\hat{y} = \underset{B(\hat{y})=y}{\arg\max} \prod_{j=1}^{T} P(\hat{y}_j, j) \tag{1}$$

where B is an operator that removes blanks and repetitions from a sequence $\hat{y}$. To do that, network's node $\alpha_{s,j}$ is defined as the probability of the best alignment of the subsequence $z_{1:s}$ after j frames. $\alpha_{s,j}$ scores can be calculated efficiently using a forward-backward algorithm, by merging together paths that reach the same node. $\alpha_{s,j}$ is then computed recursively from the values of α in the previous frame. Only transitions between blank and non-blank characters, and between pairs of distinct non-blank characters are allowed. ϵ at the beginning and end of the sequence being optional, there are two valid starting nodes and two final nodes. The coefficients α are initialized such as:

$$\alpha_{s,1} = P(z_s, 1) \text{ for } s \in \{1,2\} \text{ and } \alpha_{s,1} = 0, \forall s > 2 \tag{2}$$

Recursion is given by:

$$\begin{aligned} \alpha_{s,j} &= \max_{\tau\in\{0,1\}} (\alpha_{s-\tau,j-1}) P(z_s, j), \text{if } z_s \in \{\epsilon, z_{s-2}\} \\ \zeta_{s,j} &= \underset{\tau\in\{0,1\}}{\arg\max} (\alpha_{s-\tau,j-1}) \\ \alpha_{s,j} &= \max_{\tau\in\{0,1,2\}} (\alpha_{s-\tau,j-1}) P(z_s, j), \qquad \text{otherwise} \\ \zeta_{s,j} &= \underset{\tau\in\{0,1,2\}}{\arg\max} (\alpha_{s-\tau,j-1}) \end{aligned} \tag{3}$$

Then, the probability of the best alignment is given by:

$$P(\hat{y}) = \max_{\tau \in \{0,1\}} (\alpha_{|z|-\tau,T}) \tag{4}$$

Alignment $\hat{y}$ can finally be computed with an inverse recursion. The initial unit is initialized such as:

$$\hat{y}_T = |z| - \arg\max_{\tau \in \{0,1\}} (\alpha_{|z|-\tau,T}) \tag{5}$$

Inverse recursion is given by:

$$\hat{y}_{j-1} = \hat{y}_j - \zeta_{\hat{y}_j,j} \tag{6}$$

Calculations are performed in log-space using the log-sum-exp trick [27] to avoid numerical instabilities. As some phonemes from target languages can be unseen in the training languages, the acoustic model will be unable to predict them, resulting in all alignment having a probability of zero. To get rid of this problem, a small amount of uniformly distributed noise is added to all entries of the posteriogram, as suggested in [8].

4. EXPERIMENTAL SETUP

4.1 Dataset

For this study, we consider several language subsets of the DALI dataset. They are described in Table 1. Experiments are conducted using 5 source languages for the initial multilingual system development. These source languages are: English, German, French, Spanish and Italian. English is considered as a high-resource language. The 4 others languages are considered as low-resource languages in this study. The split between train, validation and test datasets for the first five languages is an artist aware split [28]. We also consider 4 additional target zero-resource languages: Portuguese, Polish, Finnish and Dutch. Data from these languages are only used for evaluation. The split of the different language data, i.e. dali ids belonging to each dataset, are made publicly available at `https://github.com/deezer/MultilingualLyricsToAudioAlignment`. One dataset, that we named *5lang*, is created for multilingual training. The training and validation sets of this dataset are generated by simply concatenating respectively the training and validation sets of the 5 source languages. This dataset is largely unbalanced, English data dominating the corpus. Balancing the dataset with oversampling was tested without modification on performances of the multilingual model on low-resource and zero-resource languages. Similar results were also found for speech [29]. Worse, it significantly degrades results for the English language. These results were expected as the quantity of English data being far superior in comparison to other languages in the multilingual dataset, diminishing their importance could only degrade results for the multilingual model when tested on English dataset. Results of multilingual models trained with balanced dataset are displayed in supplementary materials.

Language	# Phonemes	Train (h)	Test (h)
English (en)	44 (5)	192.7	31.5
German (de)	44 (1)	17.4	2.3
French (fr)	42 (0)	8.9	0.9
Spanish (es)	35 (3)	8.4	1.1
Italian (it)	33 (0)	8.5	1.2
Portuguese (pt)	37 (0)	X	1.8
Polish (pl)	31 (2)	X	4.2
Finnish (fi)	25 (0)	X	3.1
Dutch (nl)	41 (2)	X	3.1

Table 1. Description of DALI language subset datasets and corresponding phoneme dictionary sizes. In parenthesis are displayed the number of phonemes only occurring in the given language and its equivalent ISO 639.1 code

The procedure to generate training samples and corresponding labels for the acoustic model is similar to the one described in [25]. To recall, Spleeter [18] is used to isolate vocals from each song. Training samples are then computed by segmenting extracted vocals. The character sequence associated with a segment is created from word level annotations of DALI by concatenating all words whose start position is within the segment. An instrumental token is generated if no words are present in the segment. For phoneme models, the phoneme sequence associated with a segment is generated from his corresponding character sequence using Phonemizer[2]. Phonemizer includes GPT tools for most common languages. It decomposes each word into a sequence of IPA symbol. To create the phoneme dictionary of one given language, we collect all IPA phonemes present in the corresponding dataset. For simplicity, we did not consider IPA symbols others than vowels and consonants. Sizes of dictionaries of phoneme of each language are given in Table 1. After concatenating and merging all dictionaries, we obtain a universal phoneme set of 62 phonemes. The language sharing factor [24] for the nine languages we used in this study is 5.35. It means that, on average, one unit of the universal phoneme set is shared by 5 to 6 languages of the language pool which supports the fact that IPA phonemes are rather consistent across languages that we consider in this study.

4.2 Parameters of acoustic models

We use the same architecture for all acoustic models. Several sets of regularisation and architecture's size parameters were tested without a clear impact on performances. Parameters of architecture are similar to those used in [25]. The model has 3 layers of bidirectional LSTM and a dense layer. It takes as input mel-scale log filterbanks coefficients and energy plus deltas and double-deltas. The acoustic model output is the probabilities of characters or IPA phonemes. In the first case, the set of outputs is the concatenation of the Latin alphabet, the apostrophe, the instrumental token, the space token and the CTC blank symbol

[2] `https://github.com/bootphon/phonemizer`

ϵ. A set of size 30 is obtained. In the second case, it is constituted of the universal phoneme set, plus the instrumental token, the space token and the CTC blank symbol ϵ. A set of size 65 is obtained. Parameters of training are the same as those used in [25].

4.3 Evaluation

To evaluate our system, we use the *Average Absolute Error* (AAE) [13]. For its calculation, the absolute difference between the actual start of the word timestamp and its estimation for each word is calculated. The final error score for a song is obtained by averaging over all word-level errors. A known issue of this metric is its perceptive dependence on tempo. In fact, one absolute error will not be perceived the same if the tempo is fast or slow. The *Percentage of correct onsets* (PCO) [14] was proposed to mitigate this effect. It is computed as the percentage of start of the word timestamps whose estimation are below a certain distance from the ground truth. This metric considers that errors bellow a certain threshold fall within human listeners perceptive tolerance. We use 0.3 seconds as the tolerance window. Both metrics are classic metrics of MIREX lyrics-to-audio alignment challenge. They are computed using the same evaluation script as the one used for the challenge [30] [3].

5. RESULTS AND DISCUSSION

5.1 State of the art comparison

To validate our implementation, We first compare our system with two state-of-the-art ones. Results are collected from the 2019 MIREX lyrics-to-audio alignment challenge. For this comparison, we use characters as intermediate representation space and only English for training. We use three standard evaluation datasets for lyrics-to-audio task. Hansen [31] and Mauch [14] are constituted of respectively 9 and 20 English pop music songs. Jamendo [8] is made of 20 English music songs of several western genres. All three datasets are annotated with start-of-word timestamps. Results are summarized in Table 2.

Our system performances are close to those of GYL1, with no significant differences for PCO metric on the three evaluation datasets. Although we use an architecture somewhat similar to SDE2 (i.e. a CTC based approach with a pre-step of singing voice separation), we report significantly better performances. It is worth noting that GYL1 and our system both use the English part of DALI as training dataset, while SDE2 uses a private dataset of unknown quality. We can postulate that the DALI dataset annotation quality is higher, which would explain the better performances reached by our implementation despite using a much smaller train set than SDE2.

[3] `https://github.com/georgid/AlignmentEvaluation`

Dataset	System	Mean AAE (s)	Mean PCO (%)
Hansen	SDE2 [8]	0.39 (0.12)	88 (3)
	GYL1 [9]	**0.10 (0.03)**	**97 (1)**
	Ours	0.18 (0.05)	95 (2)
Mauch	SDE2 [8]	0.26 (0.04)	87 (2)
	GYL1 [9]	**0.19 (0.03)**	**91 (2)**
	Ours	0.22 (0.03)	**91 (1)**
Jamendo	SDE2 [8]	0.38 (0.11)	87 (3)
	GYL1 [9]	**0.22 (0.06)**	**94 (2)**
	Ours	0.37 (0.05)	92 (2)

Table 2. Comparison between our character based architecture trained with the English part of DALI and state-of-the-art systems on standard lyrics-to-audio alignment evaluation datasets. Mean AAE is better if smaller, mean PCO is better if larger. Standard errors over tested songs are given in parenthesis

5.2 Multilingual generalization

Results of multilingual generalization experiments are displayed in Figure 2. Precise numerical values are reported in supplementary materials. Several conclusions can be drawn:

- Using a multilingual training set helps For both character and phoneme based architectures, the model exhibiting the best multilingual generalization is trained with multilingual dataset. In fact, this model significantly outperforms the ones trained on English on low-resource and zero-resource languages without degrading performances on English. With phoneme as intermediate representation, it even improves results on English. On low-resource languages, multilingual trained model obtains results on par with models trained only on the target language (e.g. French trained model on French dataset). It is worth noticing that the multilingual training dataset is only marginally larger than the English one. Performances differences are to be attributed to the additional information the model was able to extract from the diversity of languages seen during training.

- Use phonemes over characters as an intermediate representation has better performances Performances of phoneme based architectures are almost always better than those of their character based counterparts in all our experimental setups. The gap is bigger for models trained on the multilingual dataset than for those trained on monolingual ones. The only models that are not improved are the ones trained and tested on the same languages. Such results show that the use of phoneme as an intermediate representation enables transfer knowledge between language better than character representation.

- Training on multilingual data and a phoneme internal representation yields the best results in all considered cases Training the acoustic model on multilingual data and use a universal phoneme set is a relevant way for improving the generalization capacity of the considered lyrics-to-audio alignment architecture even to zero-

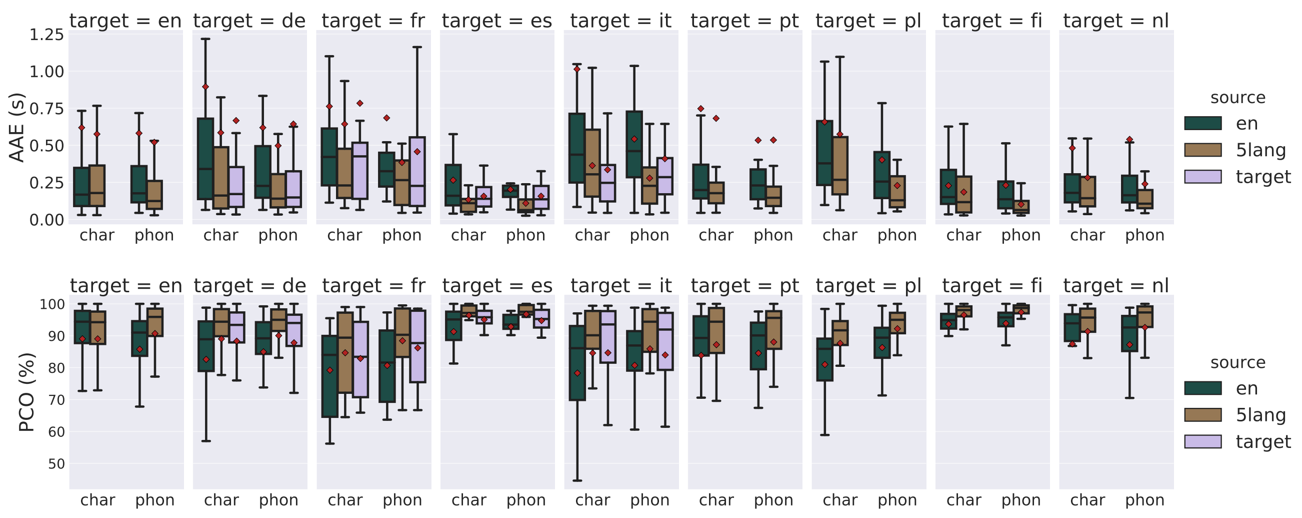


Figure 2. Lyrics-to-audio evaluation on DALI language subset datasets for phoneme and character based architectures. Several training set design strategies are considered. Languages are given by their ISO 639.1 code. Here "source" refers to data language used to train the given model and "target" refers to data language used to evaluate the trained model. When source is equal to target, architectures are trained and tested on the same language. Mean AAE is better if smaller, mean PCO is better if larger. Mean values are displayed using squares

resource scenarios.

6. CONCLUSION

In this paper, we investigated extending state-of-the-art methods in the multilingual context. Focusing on one architecture that seemed fit for generalization, we demonstrated that design choices regarding the training dataset and the acoustic representation space are salient factors. We have shown that using many languages to train the acoustic model and a universal phoneme set improves the multilingual generalization of such architecture. In this work, we have built a dataset using the language distribution found in DALI, which resulted in a largely unbalanced dataset. For comparison, we also conducted experiments with a balanced dataset, in which all 5 languages were equally present. The performance was similar, except for English, when it was significantly degraded. This raises the issue of how to design training sets in a setting where several high-resource languages are available. Although there are no publicly available datasets exhibiting such characteristics, future work should investigate this case. Existing works on multilingual speech processing [11] point towards increasing model complexity to circumvent this. Also, only a small set of languages were considered in this study. Additional experiments on a wider, more diverse set of songs remain to be conducted. Finally, future works should consider the specific case of songs with multilingual lyrics. This problem, known as code-switching, has been studied for speech [21] but never for music. Such a phenomenon is however not uncommon in popular music [10], thus it should be addressed too.

7. REFERENCES

[1] A. M. Kruspe, "Application of Automatic Speech Recognition Technologies to Singing Doctoral Thesis," Ph.D. dissertation, University Fraunhofer, apr 2018. [Online]. Available: http://ieeexplore.ieee.org/document/7178348/

[2] A. Mesaros, "Singing Voice Recognition for Music Information Retrieval," Ph.D. dissertation, Tampere university of technology, 2012. [Online]. Available: https://dspace.cc.tut.fi/dpub/handle/123456789/21404

[3] H. Fujihara, M. Goto, J. Ogata, and H. G. Okuno, "Lyric synchronizer: Automatic synchronization system between musical audio signals and lyrics," *IEEE Journal on Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1252–1261, 2011.

[4] A. Kruspe, "Automatic B**** Detection," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, no. September, 2016, pp. 3–4.

[5] C. W. Wightman and D. T. Talkin, "The aligner: Text-to-speech alignment using markov models," in *Progress in speech synthesis*. Springer, 1997, pp. 313–323.

[6] A. Haubold and J. R. Kender, "Alignment of speech to highly imperfect text transcriptions," in *Proc. IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2007, pp. 224–227.

[7] B. Sharma, C. Gupta, H. Li, and Y. Wang, "Automatic lyrics-to-audio alignment on polyphonic music using singing-adapted acoustic models," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 396–400.

[8] D. Stoller, S. Durand, and S. Ewert, "End-to-end Lyrics Alignment for Polyphonic Music Using an Audio-to-Character Recognition Model," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019. [Online]. Available: http://arxiv.org/abs/1902.06797

[9] C. Gupta, E. Yılmaz, and H. Li, "Automatic lyrics transcription in polyphonic music: Does background music help?" *arXiv preprint arXiv:1909.10200*, 2019.

[10] E. E. Davies and A. Bentahila, "Translation and code switching in the lyrics of bilingual popular songs," *The Translator*, vol. 14, no. 2, pp. 247–272, 2008.

[11] S. Watanabe, T. Hori, and J. Hershey, "Language independent end-to-end architecture for joint language and speech recognition," in *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2017.

[12] J. Cho, M. K. Baskar, R. Li, M. Wiesner, S. H. Mallidi, N. Yalta, M. Karafiat, S. Watanabe, and T. Hori, "Multilingual Sequence-to-Sequence Speech Recognition: Architecture, Transfer Learning, and Language Modeling," in *Proc. IEEE Spoken Language Technology Workshop (SLT)*, no. 1, 2019, pp. 521–527.

[13] A. Mesaros and T. Virtanen, "Automatic alignment of music audio and lyrics," in *Proc. Int. Conference on Digital Audio Effects (DAFx)*, 2008, pp. 1–4. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.212.6683{&}rep=rep1{&}type=pdf

[14] M. Mauch, H. Fujihara, and M. Goto, "Integrating Additional Chord Information Into HMM-Based Lyrics-to-Audio Alignment," in *Proc. IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, jan 2012, pp. 200–210. [Online]. Available: http://ieeexplore.ieee.org/document/5876304/

[15] G. Dzhambazov and X. Serra, "Modeling of phoneme durations for alignment between polyphonic audio and lyrics," in *Proc. of the 12th International Conference in Sound and Music Computing (SMC)*, 2015, pp. 281–286.

[16] G. Dzhambazov and A. Srinivasamurthy, "On the Use of Note Onsets for Improved Lyrics-To-Audio Alignment in Turkish Makam Music," in *Proc. 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 716–722.

[17] G. Meseguer-brocal and A. Cohen-hadria, "Dali : a Large Dataset of Synchronized Audio , Lyrics and Notes , Automatically Created Using Teacher-Student Machine Learning Paradigm," in *Proc. International Society on Music Information Retrieval Conference (ISMIR)*, 2018.

[18] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: A fast and state-of-the art music source separation tool with pre-trained models," in *Proc. Late-Breaking/Demo of International Society of Music Information Retrieval Conference (ISMIR)*, November 2019, deezer Research.

[19] S. Toshniwal, T. N. Sainath, R. J. Weiss, B. Li, P. Moreno, E. Weinstein, and K. Rao, "Multilingual speech recognition with a single end-to-end model," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4904–4908.

[20] M. Müller, S. Stüker, and A. Waibel, "Language adaptive multilingual ctc speech recognition," in *Proc. International Conference on Speech and Computer (SPECOM)*. Springer, 2017, pp. 473–482.

[21] K. Li, J. Li, G. Ye, R. Zhao, and Y. Gong, "Towards code-switching asr for end-to-end ctc models," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6076–6080.

[22] H. Sak, F. de Chaumont Quitry, T. Sainath, K. Rao *et al.*, "Acoustic modelling with cd-ctc-smbr lstm rnns," in *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015, pp. 604–609.

[23] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ACM International Conference Proceeding Series*, vol. 148, 2006, pp. 369–376.

[24] T. Schultz and A. Waibel, "Language-independent and language-adaptive acoustic modeling for speech recognition," *Speech Communication*, vol. 35, no. 1-2, pp. 31–51, 2001.

[25] A. Vaglio, R. Hennequin, M. Moussallam, G. Richard, and F. d'Alché-Buc, "Audio-based detection of explicit content in music," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 526–530.

[26] D. G. Forney, "The viterbi algorithm," *Proc. of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[27] A. Hannun, "Sequence modeling with ctc," *Distill*, vol. 2, no. 11, p. e8, 2017.

[28] A. Flexer, "A closer look on artist filters for musical genre classification," in *Proc. of the 8th International Conference on Music Information Retrieval (ISMIR)*, no. 122, 2007, pp. 16–17.

[29] T. Alumäe, S. Tsakalidis, and R. Schwartz, "Improved multilingual training of stacked neural network acoustic models for low resource languages," in *Proc. Interspeech*, 09 2016, pp. 3883–3887.

[30] G. Dzhambazov, “Knowledge-Based Probabilistic Modeling For Tracking Lyrics In Music Audio Signals,” Ph.D. dissertation, Universitat Pompeu Fabra Barcelona, 2017. [Online]. Available: http://www.tdx.cat/bitstream/handle/10803/404681/tgd.pdf?sequence=1{%}0Ahttp://mtg.upf.edu/node/3751

[31] J. K. Hansen, “Recognition of Phonemes in A-cappella Recordings using Temporal Patterns and Mel Frequency Cepstral Coefficients,” in *Proc. 9th Sound and Music Computing Conference (SMC)*, 2012, pp. 494–499.

SINGING LANGUAGE IDENTIFICATION USING A DEEP PHONOTACTIC APPROACH

Lenny Renault**, Andrea Vaglio**†*, Romain Hennequin**

*Deezer Research, `research@deezer.com`
†LTCI, Télécom Paris, Institut Polytechnique de Paris

ABSTRACT

Extensive works have tackled *Language Identification* (LID) in the speech domain, however their application to the singing voice trails and performances on *Singing Language Identification* (SLID) can be improved leveraging recent progresses made in other singing related tasks. This work presents a modernized phonotactic system for SLID on polyphonic music: phoneme recognition is performed with a *Connectionist Temporal Classification* (CTC)-based acoustic model trained with multilingual data, before language classification with a recurrent model based on the phonemes estimation. The full pipeline is trained and evaluated with a large and publicly available dataset, with unprecedented performances. First results of SLID with out-of-set languages are also presented.

Index Terms— Singing language identification, CTC training, phonotactic approach, music information retrieval

1. INTRODUCTION

Having semantically meaningful and accurate descriptions of songs is crucial for organizing and retrieving relevant tracks in a large musical catalog [1]. Language tags are of particular interest for characterizing vocal music. This information can easily be extracted using the song lyrics with a text-based language identifier [2]. However, lyrics are not ubiquitously available for consequent musical collections. One then may want to estimate the song language from the frequently accessible metadata (e.g. song title, artist name). Yet, this method is limited as the metadata language can differ from the song language, and metadata may not contain enough information for retrieving the language [3]. A more robust approach would be to extract the language information from the audio content. Such data is indeed always available, but the task is arguably more challenging.

The speech community has long tackled language recognition from audio data, notably with the *Language Recognition Evaluation* (LRE) series [4]. However, the task was scarcely transposed in the music domain and most of the techniques used for spoken LID have yet to be adapted ffor SLID. The latter task is more difficult, as the prosodic specificities of languages are disturbed by the greater variabilities of the singing voice, in terms of pitch, pronunciation and vowels duration [5]. The musical accompaniment can be framed as noise and assumed to be loud and highly correlated with the signal of interest, being the voice.

Previous works on SLID include acoustic-phonetic systems which characterize language-specific acoustic events and their distribution with carefully chosen acoustic features, such as *Mel Frequency Cepstral Coefficients* (MFCC) [6], *Stabilized Auditory Images* (SAI) [7] and *Temporal Patterns* (TRAP) [8]. Statistical modeling and supervised classification are then applied to identify the language. In particular, Kruspe's system using the i-vector extraction technique obtains the current best performances on SLID [9], with 78% accuracy on a capella performances in 3 languages.

Phonotactic approaches, on the other hand, try to identify phonemes from the audio and examine their combinations and sequences, which are distinctive from one language to another [10]. These approaches are more resource-demanding as phoneme recognizers, or acoustic models, have to be trained. Mehrabani et al. [11] use multiple language-specific phoneme recognizers trained on speech data, to then compute language likelihoods with n-gram models for each target language. While the performances are on par with Kruspe's i-vector-based approach [9], it is more complex to train and it is hardly scalable to a large set of languages. In [12], the author simplifies the approach by using a unique *Deep Neural Network* (DNN)-based English phoneme recognizer and identifies the language from phoneme statistics. While singing data are included in the acoustic model training, the frame-wise phoneme annotations are obtained by a forced-alignment step, which leads to poorly annotated data. Also, this statistics-based language modeling overlooks the information contained in the phoneme transitions.

Recent works have trained new acoustic models with singing data and show great results in lyrics transcription [13], lyrics-to-audio alignment [14, 15] and explicit content detection [16], using more recent DNN techniques. In this work, we propose to apply these advances to a phonotactic SLID system: in particular, the usage of the CTC algorithm allows the acoustic model to be trained with DALI, a large multilingual singing dataset [17], while alleviating the need for frame-level aligned lyrics. For language modeling, we use a recurrent architecture that can capture temporal information

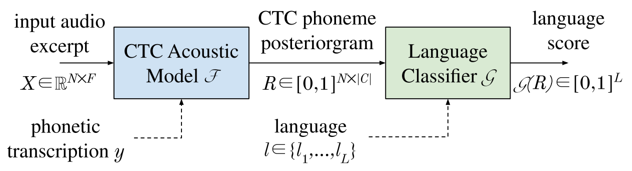

input audio excerpt
$X\in\mathbb{R}^{N\times F}$
CTC Acoustic Model $\mathcal{F}$
CTC phoneme posteriorgram
$R\in[0,1]^{N\times|C|}$
Language Classifier $\mathcal{G}$
language score
$\mathcal{G}(R)\in[0,1]^L$
phonetic transcription y
language $l\in\{l_1,...,l_L\}$

Fig. 1. Overview of the proposed SLID system. During training, the phonetic transcription of the lyrics y and the corresponding language label l of the excerpt X are provided.

in phoneme estimation sequences. We show that our system outperforms the previous state-of-the-art in a standard closed-set scenario, obtaining a 91.7% balanced accuracy score on polyphonic songs in 5 languages. We also investigate a harder setup with out-of-set languages where we can acknowledge the limits of our model. In Section 2, we describe the key aspects of our deep phonotactic SLID approach. The dataset, baselines and implementation details are given in Section 3. Results are presented in Section 4, providing a first reproducible benchmark on the public DALI dataset[1].

2. PROPOSED SYSTEM

As in previous SLID works, we frame the problem as a multi-class classification task. The system takes as input audio features of a musical excerpt $X \in \mathbb{R}^{N\times F}$, with N the number of time frames and F the feature dimensionality. The system should then estimate the language l used in the musical excerpt, among a set of L languages $\{l_1, l_2, ..., l_L\}$.

Our deep phonotactic system, as illustrated by Figure 1, is composed of two main models: an acoustic model $\mathcal{F}$ for phoneme estimation, followed by a language classifier $\mathcal{G}$. The acoustic model estimates the occurring phonemes in the input audio X by producing a $|C|$-dimensional vector of probabilities at each time frame, with C the set of characters supported by $\mathcal{F}$. Here, C encompasses the *International Phonetic Alphabet* (IPA) symbols appearing in the training excerpts, a word-boundary "space" token, an instrumental "I" token and the blank token ϵ introduced by the CTC algorithm in Section 2.1. The sequence of phoneme probability vectors is referred as the posteriorgram $R := \mathcal{F}(X) \in [0, 1]^{N\times|C|}$.

The language classifier $\mathcal{G}$ then produces a language probability vector score $\mathcal{G}(R) \in [0, 1]^L$ from the posteriorgram R. The language decision is finally taken from this vector score:

$$\hat{l} = \underset{l\in\{l_1,l_2,...,l_L\}}{\arg\max} \; [\mathcal{G}(\mathcal{F}(X))]_l. \tag{1}$$

Previous phonotactic approaches on SLID made distinct training of the two models, with different training sets [11, 12]. $\mathcal{F}$ needs the phonetic transcription y of each training excerpt X, whereas $\mathcal{G}$ needs the posteriorgram representation R and the language label l of its training excerpts. We use the same dataset of musical excerpts for both model training. Following the works on joint *Automatic Speech Recognition* (ASR) and LID [18], we train both models simultaneously. The joint loss function can be expressed as:

$$\mathcal{L}_{Joint}(R, \hat{l}, y, l) = \mathcal{L}_{\text{CTC}}(R, y) + \lambda\mathcal{L}_{\text{LID}}(\hat{l}, l), \tag{2}$$

with λ the weight given to the cross-entropy LID loss with regard to the CTC loss. The balance between the two losses is decisive for the system performance: various strategies are considered and described in Section 3.5.

2.1. Phoneme recognition

For the acoustic model $\mathcal{F}$, we rely on a *Convolutionnal Recurrent Network* (CRNN) trained with the CTC algorithm, as in [16]. CTC-based acoustic models were successfully implemented for singing-related tasks, such as lyrics-to-audio alignment [13, 15] and keyword spotting [16]. Following their works, we also employ a singing voice separation preprocessing step during training and inference, which improves performance over using polyphonic data in [13]. For the recurrent layers, we choose bidirectional *Long Short-Term Memory* (LSTM) cells to take the full sequence into account when predicting characters at each time frame.

The CTC algorithm enables to train a *Recurrent Neural Network* (RNN) with weakly aligned data, e.g. at word or line level, by introducing a blank token ϵ in the set C of characters supported by the model. The associated loss function computes the probability of an output sequence by marginalizing over all possible alignments with the input. Following the work in [15], the output sequences are composed of multilingual phonemes according to the IPA. As the CTC loss function is differentiable, network training can be done with any gradient descent algorithm, by providing the phonetic transcription y of the segment lyrics. Further details on the CTC algorithm can be found in [19].

2.2. Language Classifier

The language classifier $\mathcal{G}$ is built upon a RNN. The usage of recurrent architectures has been successful for end-to-end spoken LID [20, 21]. Here, the phoneme posteriorgram representation is given as input, instead of the raw acoustic features extracted from the audio excerpt. Bidirectional LSTM layers are chosen with the last layer only outputting a single probability vector for the whole input segment. This architecture takes the combination of phonemes into account, as in n-gram modeling [11], but with confidence scores on the phoneme predictions by using the full probability vectors, as in statistical modeling [12]. To avoid vanishing gradient on very long input sequence [22], we choose to perform SLID on fixed-length segments for a given song. Song language is then inferred from the mean of language scores output by the system on all segments.

[1]The dataset split can be found at `https://github.com/deezer/SingingLanguageIdentification`

3. EXPERIMENTAL SETUP

3.1. Dataset

All versions of our system are trained and tested on tracks from the DALI dataset [17]. This dataset contains 5358 songs of various western genres with the lyrics annotations at word-level and song-level language labels. All tracks are downsampled to 16kHz and converted to mono. Musical accompaniments are removed by vocal extraction with Spleeter [23].

We design two language sets from this dataset: a *closed-set* scenario and an *open-set* scenario. The closed-set retains languages with more than 10 hours of data each: English, French, German, Italian and Spanish. The open-set also adds a sixth label "Others" regrouping low-resource languages (Dutch, Finnish, Portuguese, Polish). Train, validation and test sets are obtained with a 80%-10%-10% language-wise and artist-aware split [24]. Songs in neither target nor low-resource languages are also labelled as "Others" and added to the open-set test set only. These out-of-domain samples help monitoring the generalization of out-of-set modeling learned from the subset of in-domain "Others" languages. As English is over-represented in the dataset, all systems and baselines are trained with a class-weighted LID objective function.

Our system performs SLID at segment-level. Each song is split into 20s segments with a 0.5 overlapping factor between two consecutive segments. Segment lyrics are retrieved using the word-level annotations from DALI and decomposed into IPA symbols using Phonemizer [25]. Collecting all phonemes occurring in the training segments and adding the space, instrumental and blank tokens, the total number of characters obtained is $|C| = 66$ in the closed-set scenario and $|C| = 71$ in the open-set scenario. For the segment language label, the FastText language identifier [26] is used on the segment lyrics. A segment is labeled *instrumental* when it has less than 3 words, or *ambiguous* when the lyrics repetitiveness or the FastText non-confidence score is above an empirically found threshold. During inference, *ambiguous* and *instrumental* scores are not taken into account when estimating the song language from segment language scores.

3.2. Baseline systems

Two baseline systems are implemented for comparison with our system. The *Metadata* baseline is a text-based language identifier using the artist name and song title metadata provided with the DALI dataset. The language is extracted using the FastText language identifier [26].

The *i-vector* baseline is an acoustic-phonetic i-vector-based system, as in [9]. Implemented with the Kaldi LRE receipt [27], this system computes 600-dimensional i-vector per vocal-isolated song. Sequences of 20 MFCC feature vectors are extracted then modeled by a *Gaussian Model Mixtures* (GMM)-based *Universal Background Model* (UBM). Supervised language classification is performed from the i-vector representation of the song using *Support Vector Machines* (SVM) with a cosine kernel, as in [9, 28].

3.3. Acoustic model architecture

40 Mel-scale log filterbanks coefficients and energy features, plus deltas and double-deltas are computed from the extracted vocals using a 32ms Hann window with 0.5 overlap. The input feature sequences are downsampled by two sub-modules each composed of a 2D-convolutional layer (32 filters with kernel size 3×3), a ReLU activation function and a 2×3 max-pooling layer: sequence length is thus divided by 4.

The recurrent part of the acoustic model is composed of 3 bidirectional LSTM layers with 256-dimensional hidden states. Dropout and recurrent dropout of 0.1 each is applied. Finally a time-distributed dense layer and a softmax activation function are applied for obtaining per-frame character probability vectors from C. The CTC layer and objective function implementations are taken from [29].

3.4. Language classifier

Inputted posteriorgrams are pre-processed by a deterministic cleaning module: frames with ϵ-emission probability $p(\epsilon) > 95\%$ are removed, to account only for frames with actual phoneme predictions.

The language classifier model is composed of 2 bidirectionnal LSTM layers with 64-dimensional hidden states each. The second layer outputs a single vector per segment, which is processed by a dense layer with a softmax activation function to produce one language probability vector. Recurrent layers have a 0.1 recurrent dropout factor and 0.2 dropout is applied between each layer. A class-weighted categorical cross-entropy loss function is used for training the model given the one-hot encoded language labels.

3.5. Training strategies for our approach

We test two strategies for training our system, implemented in Tensorflow. Each training variant relies on the ADAM optimization algorithm [30] with a learning rate of 10^{-3}, a batching size of 32 and validation-based early stopping.

The *2-step* variant first trains the acoustic model $\mathcal{F}$ alone. The language classifier $\mathcal{G}$ is then trained for SLID from the posteriorgrams of the training segments computed by $\mathcal{F}$. The *Joint* variant trains both models at the same time from scratch. With hyper-parameter tuning, we found that training the system with a loss balance $\lambda = 0.1$, then fine-tuning it with $\lambda = 100$ yields the best performances on the validation set.

3.6. Ablation study

We evaluate the relevance of our system parts by designing two simplified systems for comparison. The *E2E* system is an end-to-end approach to SLID with the same architecture as the *Joint* variant, except for the CTC component which is removed from the loss function. The phoneme recognition task is ignored as the model is solely trained to identify the language in song segments.

The *Statistics* system is a modified *2-step* variant. Instead of the recurrent layers, the language classifier is a pooling step of the mean and variance statistics of each phoneme class over the full song length. Song language is directly predicted from these statistic vectors using SVM. This system is analogous to a modernized version of [12], with a CTC-based acoustic model instead of the DNN-based one.

4. RESULTS

4.1. Performances in the closed-set scenario

The results of the evaluation of our systems on the test songs in a closed-set scenario are reported Table 1.

System	bAccuracy (%)	F1-score (%)
Metadata	76.48 (3.98)	76.71 (3.45)
i-vector	77.26 (3.88)	67.78 (3.57)
E2E	59.90 (4.33)	65.43 (4.47)
Statistics	88.46 (3.04)	89.00 (2.95)
2-step	88.62 (3.03)	90.75 (2.62)
Joint	**91.74 (2.70)**	**92.39 (2.31)**

Table 1. Systems evaluation in the closed-set scenario. Measured by balanced accuracy (bAccuracy) and macro-averaged F1-score (with standard errors in parenthesis).

All phonotactic approaches (*Statistics*, *2-step* and *Joint*) outperform the *Metadata* baseline, on the contrary of the *E2E* system. The phonetic information contained in the audio data is thus better suited for estimating the language than common metadata. Reliable estimations from the raw audio can not be achieved with a naive end-to-end approach and seems to require more refined techniques. Our deep phonotactic system also significantly outperforms the re-implemented state-of-the-art *i-vector* system. In particular, joint training of the acoustic model and language classifier further improves the system performance, as the *Joint* variant yields the best overall scores, with 91.7% of balanced accuracy.

Regarding the efficiency of each system part, the *Statistics* system has better performances that the *i-vector* baseline, which was not the case between the two analog approaches from Kruspe [9, 12]. Hence, our CTC-based acoustic model seems to offer better modeling capability than the DNN-based model from [12]. The *2-step* variant does not significantly outperform the *Statistics* system, which implies that the language classifier can be improved. Finally, even though the side phoneme recognition task requires more detailed information for training, it proves to be profitable for SLID since the *2-step* and *Joint* systems outperform the *E2E* baseline.

System	bAccuracy (%)	F1-score (%)	Target (%)	Others (%)
Metadata	70.16 (3.46)	70.52 (3.08)	73.30 (3.45)	56.60 (4.73)
i-vector	70.87 (3.16)	54.79 (2.90)	58.74 (3.18)	35.06 (4.92)
E2E	39.57 (3.17)	35.78 (2.57)	42.93 (3.07)	0.00 (0.00)
Statistics	**83.30 (2.83)**	**80.28 (2.79)**	**81.70 (3.03))**	**73.14 (3.79)**
2-step	78.49 (2.77)	74.35 (2.92)	79.89 (3.14)	46.62 (5.40)
Joint	72.89 (2.86)	64.46 (3.14)	72.02 (3.51)	26.67 (5.35)

Table 2. Systems evaluation in the open-set scenario. Measured by balanced accuracy (bAccuracy) and macro-averaged F1-score. Macro-averaged F1-score on the target languages and the F1-score on the "Others" class are also presented. Standard errors are in parenthesis.

4.2. Performances in the open-set scenario

The results of the evaluation of our systems on the test songs in the open-set scenario are reported Table 2. All phonotactic systems still outperform the *Metadata*, *E2E* and *i-vector* approaches. However, both variants of our deep phonotactic system are less robust to the introduction of the "Others" class than the simpler *Statistics* system. Indeed, they seem to overfit on the "Others" training data. It can be explained as this class has a greater linguistic variability than other classes but has the same amount of data as a low-resource target language. This effect is further demonstrated in Table 3 as only the *Statistics* system can generalize the out-of-set modeling to out-of-domain languages unseen during training.

System	In-domain "Others" (%)	Out-of-domain "Others" (%)
i-vector	50.00 (10.66)	20.00 (4.46)
E2E	0.00 (0.00)	0.00 (0.00)
Statistics	**86.36 (7.29)**	**56.25 (5.58)**
2-step	63.64 (10.28)	21.25 (4.61)
Joint	31.82 (9.87)	11.25 (3.56)

Table 3. Performances comparison on "Others" labelled test songs in in-domain and out-of-domain languages cases. Measured by accuracy (with standard errors in parenthesis).

5. CONCLUSION

We investigate modernized phonotactic systems for SLID on polyphonic music, using recurrent models for both phoneme recognition and language classification. Trained on a publicly available multilingual dataset, the proposed system outperforms metadata-based and the previous state-of-the-art SLID approaches. The CTC-based acoustic model greatly contributes to the performance increase, both in closed-set and open-set scenarios. However, the proposed language classifier hardly exceeds statistical modeling in a closed-set scenario, and deteriorates with out-of-set languages. Future works would focus on exploring hierarchical language modeling techniques for SLID with out-of-set languages, taking inspiration from the speech literature [31].

6. REFERENCES

[1] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet, "Semantic annotation and retrieval of music and sound effects," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 467–476, 2008.

[2] Jose P. G. Mahedero, Álvaro MartÍnez, Pedro Cano, Markus Koppenberger, and Fabien Gouyon, "Natural language processing of lyrics," in *Proc. of the Annual ACM International Conference on Multimedia*, 2005, p. 475–478.

[3] Wei-Ho Tsai and Hsin-Min Wang, "Towards Automatic Identification of Singing Language In Popular Music Recordings," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2004, pp. 568–576.

[4] Alvin F. Martin, Craig S. Greenberg, John M. Howard, George R. Doddington, and John J. Godfrey, "Nist language recognition evaluation - past and future," in *Proc. of the Odyssey The Speaker and Language Recognition Workshop*, 2014, pp. 145–151.

[5] Annamaria Mesaros and Tuomas Virtanen, "Automatic recognition of lyrics in singing," *EURASIP Journal on Audio, Speech, and Music Processing*, 2010.

[6] Jochen Schwenninger, Raymond Brueckner, Daniel Willett, and Marcus Hennecke, "Language identification in vocal music," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2006, pp. 377–379.

[7] Vijay Chandrasekhar, Mehmet Emre Sargin, and David A. Ross, "Automatic language identification in music videos with low level audio and visual features," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 5724–5727.

[8] Anna M. Kruspe, Jakob Abesser, and Christian Dittmar, "A GMM approach to singing language identification," *Journal of the Audio Engineering Society*, pp. 140–148, 2014.

[9] Anna M. Kruspe, "Improving singing language identification through I-vector extraction," in *Proc. of the International Conference on Digital Audio Effects (DAFx)*, 2014.

[10] Haizhou Li, Bin Ma, and Kong A. Lee, "Spoken language recognition: From fundamentals to practice," *Proc. of the IEEE*, vol. 101, no. 5, pp. 1136–1159, 2013.

[11] Mahnoosh Mehrabani and John H. L. Hansen, "Language identification for singing," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 4408–4411.

[12] Anna M. Kruspe, "Phonotactic language identification for singing," in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2016, pp. 3319–3323.

[13] Daniel Stoller, Simon Durand, and Sebastian Ewert, "End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 181–185.

[14] Chitralekha Gupta, Emre Yilmaz, and Haizhou Li, "Acoustic modeling for automatic lyrics-to-audio alignment," in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2019, pp. 2040–2044.

[15] Andrea Vaglio, Romain Hennequin, Manuel Moussallam, Gaël Richard, and Florence d'Alché-Buc, "Multilingual lyrics-to-audio alignement," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2020.

[16] Andrea Vaglio, Romain Hennequin, Manuel Moussallam, Gaël Richard, and Florence d'Alché-Buc, "Audio-based detection of explicit content in music," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 526–530.

[17] Gabriel Meseguer-Brocal, Alice Cohen-Hadria, and Geoffroy Peeters, "DALI: A Large Dataset of Synchronized Audio, Lyrics and notes, Automatically Created using Teacher-student Machine Learning Paradigm," in *Proc. of the International Society for Music Information Retrieval (ISMIR)*, 2018, pp. 431–437.

[18] Shinji Watanabe, Takaaki Hori, and John R. Hershey, "Language independent end-to-end architecture for joint language identification and speech recognition," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2017, pp. 265–271.

[19] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. of the International Conference on Machine Learning (ICML)*, 2006, pp. 369–376.

[20] Trung Ngo Trong, Ville Hautamäki, and Kong Aik Lee, "Deep language: a comprehensive deep learning approach to end-to-end language recognition," in *Proc. of the Odyssey The Speaker and Language Recognition Workshop*, 2016, pp. 109–116.

[21] Weicheng Cai, Danwei Cai, Sheng Huang, and Ming Li, "Utterance-level end-to-end language identification using attention-based cnn-blstm," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5991–5995.

[22] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al., "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," *A field guide to dynamical recurrent neural networks*, 2001.

[23] Romain Hennequin, Anis Khlif, Félix Voituret, and Manuel Moussallam, "Spleeter: a fast and efficient music source separation tool with pre-trained models," *Journal of Open Source Software*, vol. 5, no. 50, pp. 2154, 2020.

[24] Arthur Flexer, "A closer look on artist filters for musical genre classification," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2007, pp. 341–344.

[25] M. Bernard, "Phonemizer (version 2.2.1)," `https://github.com/bootphon/phonemizer`, 2015, [Online; accessed 15-October-2020].

[26] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov, "Fasttext.zip: Compressing text classification models," *arXiv preprint arXiv:1612.03651*, 2016.

[27] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely, "The kaldi speech recognition toolkit," in *Proc. IEEE Automatic Speech Recognition and Understanding (ASRU)*, 2011.

[28] Najim Dehak, Pedro A. Torres-Carrasquillo, Douglas Reynolds, and Reda Dehak, "Language recognition via i-vectors and dimensionality reduction," in *Proc. of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2011, pp. 857–860.

[29] Yann Soullard, Cyprien Ruffino, and Thierry Paquet, "CTCModel: Connectionist Temporal Classification in Keras," 2018.

[30] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *International Conference for Learning Representations (ICLR)*, 2015.

[31] Trung Ngo Trong, Ville Hautamaki, and Kristiina Jokinen, "Staircase network: structural language identification via hierarchical attentive units," in *Proc. of the Odyssey The Speaker and Language Recognition Workshop*, 2018, pp. 60–67.

USER-CENTERED EVALUATION OF LYRICS-TO-AUDIO ALIGNMENT

Ninon Lizé Masclef[1] **Andrea Vaglio**[1 2] **Manuel Moussallam**[1]

[1] Deezer Research
[2] LTCI, Télécom Paris, Institut Polytechnique de Paris
research@deezer.com

ABSTRACT

The growing interest for Human-centered *Music Information Retrieval* (MIR) motivates the development of perceptually-grounded evaluation metrics. Despite remarkable progress of lyrics-to-audio alignment systems in recent years, one thing which remains unresolved is whether the metrics employed to assess their performance are perceptually grounded. Even if a tolerance window for errors was fixed at 0.3s for the *Music Information Retrieval Evaluation eXchange* (MIREX) challenge, no experiment was conducted to confer psychological validity to this threshold. Following an interdisciplinary approach, fueled by psychology and musicology insights, we consider the lyrics-to-audio alignment evaluation from a user-centered perspective. In this paper, we call into question the perceptual robustness of the most commonly used metric to evaluate this task. We investigate the perception of audio and lyrics synchrony through two realistic experimental settings inspired from karaoke, and discuss implications for evaluation metrics. The most striking features of these results are the asymmetrical perceptual thresholds of synchrony perception between lyrics and audio, as well as the influence of rhythmic factors on them.

1. INTRODUCTION

Nowadays, the machine learning community is raising the question of how to design explainable [1] and human-grounded algorithms [2]. Especially in the field of MIR, user studies and evaluation metrics plays a pivotal role in this shift towards Human-centered MIR. Subjective listening tests [3–5] and ethnomusicological studies [6] previously demonstrated the feasibility of including tasks in the real setting context of user experience. Regarding metrics, we are witnessing the transition from exclusively system-centered evaluation to user-aware evaluation. In the reference toolkit mir_eval, the lack of Human-centered metrics was justified by the complexity and cost required to develop robust subjective evaluation methods [7]. However there are a few limitations of system-based evaluation, such as their inability to capture the inherently subjective experience of MIR and the absence of necessary correlation between system-centered evaluation and users' perceptions [8]. One could, and indeed should, ask what is the meaning of the effectiveness of an algorithm without the presence of an embodied experience of human perception? In epistemic terms, how is the distance to the ground truth translated into an error measurement without the mediation of an individual? Since the advent in 2005 of the system-centered evaluation approach by the MIREX, there were several attempts at creating perceptually grounded metrics, notably among the field of music transcription [9–11], source separation [12] and audio similarity [8].

One application at the frontier of music perception and human machine interaction is karaoke. Currently, the majority of alignments used by karaoke systems are fully manually achieved, or partially corrected by human annotators. Obtaining manual annotations of lyrics-to-audio alignment is costly and time-consuming. To obtain such annotations automatically, one could turn to automatic lyrics to audio alignment system. Such system takes as input lyrics text and outputs timed position of their appearance in the audio signal, at the word, line, or paragraph level. Several recent automatic lyrics-to-audio alignment systems have achieved high performance taking inspiration from automatic speech recognition [13–15] and using large public singing voice annotation dataset like DALI [16]. Among the metrics developed for the MIREX challenge to evaluate lyrics-to-audio alignment, the most commonly used is the *Percentage of correct onsets* (PCO) ρ_τ^k, illustrated in [17], using a tolerance window for the perception of lyrics-to-audio alignment errors defined by a threshold τ [17].

$$\rho_\tau^k \;=\; \frac{1}{N_k} \sum_{word\ i} 1_{|\hat{t_i} - t_i| < \tau} \times 100 \qquad (1)$$

where N_k is the number of words in the track k, t_i the ground truth start of the word timestamp of the lyrics unit and $\hat{t_i}$ the predicted timestamp. It suggests that listeners tolerate errors falling within this window, and still perceive as synchronous lyrics and audio whose onsets are separated by this offset. A tolerance window for errors was fixed at 0.3s for the MIREX, albeit no psychology experiment was conducted to confer validity to this threshold.

Additionally, while spectacular progress has been made in the past years, the gap between state-of-the-art systems, as measured in the MIREX competition, has narrowed, with many systems achieving close to perfect PCO scores on the test sets. Therefore, it might now be important to make room for qualitative rather than quantitative metrics. In this work, we are interested in challenging the PCO metric from a user-centric perspective, focusing on how humans perceive asynchrony to derive stricter metrics for the task. To this aim, we expose the design of two perceptual experiments in Section 3 and their respective results in Section 4. We then propose a PCO adaptation in Section 5 and conclude in Section 6.

2. RELATED WORKS

Singing karaoke engages coordination of articulatory movements, music and language processing systems, as well as crossmodal integration of audio and visual stimuli. It is thus a rich context of perception involving complex stimuli. As a consequence, we briefly consider the research on all the domains outlined above to illustrate paradigms and hypotheses relevant to lyrics-to-audio alignment perception. When presented with a pair of audiovisual stimuli, individuals reported an asymmetric perception of asynchrony, with audio lagging preferred over visual lagging [18, 19]. This asymmetry has been correlated with faster transmission of the visual signal over the audio signal [18] or with the auditory dominance in temporal processing [20]. The latter hypothesis asserts that, when emitting a judgment of synchrony, audio would provide individuals a more accurate sensory information in the case of dynamic event such as music, and also a more stable internal representation of periodicity, contrary to the visual modality [20]. The listening experience is a continuous production of rhythmic expectancies [21]. In the case of sensorimotor synchronisation experiment, one effect induced by rhythmic expectancies is the anticipation of the stimuli in a sequence, also called Negative Mean Asynchrony (NMA). First reported by Dunlap [22], it states that the reaction to an audio stimuli tends to precede rather than follow the stimuli. Repp [23] discovered that individuals anticipate audio events up to 100ms ahead of time. Klemmer [24] revealed that the anticipation effect varies with the tempo of the rhythmic stimuli, usually measured in terms of InterOnset Interval (IOI) duration. He found that the reaction time of individuals when attempting to stay in phase with an isochronous stimulus, is a function of the IOI between stimuli. The reaction time was greater for shorter IOI, suggesting that individuals have less sensibility in slow tempo. These observations were further formalized as a function of local and global rhythmic context by McAuley [25]. Besides global rhythmic factors, the listening experience is punctuated by local variations. Metric events are periodic peaks of attention organized into nested hierarchies that coordinate attention to events on various time-scales, allowing for grouping and accentuation of notes [26]. Musical stresses are the cues to infer a general rhythmic pattern [26]. Among the significant factors of stress reported were the duration of syllables [26], loudness [27], alignment with beats [21] and sequence boundaries [21, 28].

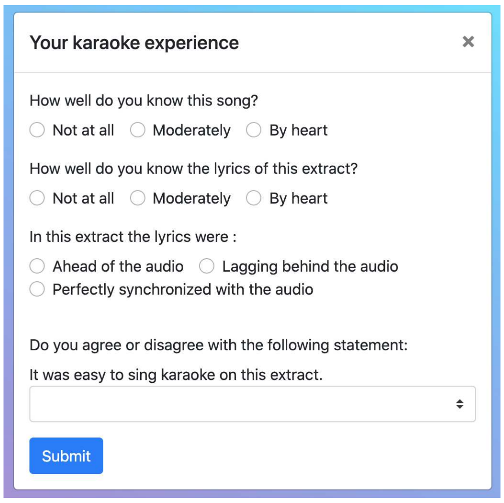
Given previous studies, our theoretical hypothesis is based on two points. Firstly, we expect individuals to tolerate more audio lagging than lyrics lagging. Secondly, we expect perception of lyrics-to-audio synchrony to rely both on global and local rhythmic context.

3. METHOD

To investigate the perception of lyrics-to-audio alignment, we designed two psychological experiments inspired from the main application of this task, karaoke. We chose karaoke as it is a popular practice where the participants' rhythmical precision is important, requiring attention to the displayed lyrics as much as to the audio. The first experiment is designed to test the influence of global parameters on human perception of audio/displayed lyrics synchrony and to investigate its symmetrical properties. The second experiment intends to explore local factors influences. To run both experiments we developed a karaoke application prototype, whose displayed textual lyrics were intentionally misaligned with the background audio according to various, controlled conditions, thereby creating an audiovisual offset. The stimuli were presented to individuals who then annotated their perceived quality of alignment in different error scenarios. A snippet of this interface is displayed in Figure 1.

Both experiments were run online, through a web interface that was designed to be correctly displayed on both computer and phone screens, for a total duration of two weeks each, between January and April 2021. The first experiment was conducted only with Deezer employees while the second experiment was public and hence involving a larger and more diverse set of participants. Before engaging in karaoke, participants are asked to fill out a questionnaire allowing us to determine their level of musical expertise and familiarity with the practice of karaoke. We collect, with their consent, a range information of their age, declared gender and native language. We do not have control on their external environment when performing karaoke (external noise) or any other factor which might disturb the readability of the karaoke (low light, uncorrected vision problem). Nevertheless, the instructions of the experiment encourage the participants to use headphones and favor a quiet environment.

In both experiments the dependent variable measured is the perceived synchrony and the amount of offset between lyrics and audio is a within subjects factor. In order to prevent from order effect, the values of audiovisual offset are presented in random order. These two experiments are akin to the Simultaneity Judgment task (SJ) widely used in the literature for studying the synchrony perception of audiovisual stimuli [18, 19].


Your karaoke experience

×

How well do you know this song?

Not at all ○ Moderately ○ By heart ○

How well do you know the lyrics of this extract?

Not at all ○ Moderately ○ By heart ○

In this extract the lyrics were :

Ahead of the audio ○ Lagging behind the audio ○ Perfectly synchronized with the audio ○

Do you agree or disagree with the following statement:

It was easy to sing karaoke on this extract.

Submit

Figure 1. Questionnaire used to evaluate lyrics-to-audio alignment.

3.1 Dataset

Since the measured effects should be valid irrespective of the song, we allow participants to choose their song for karaoke within a set of 80 songs from various genre (pop, rock, rap and metal) and language (English, French, German). We selected popular songs in the DALI dataset [16] with alignment done at word level. The first criterion for the choice of songs was their popularity, so that we can expect a large proportion of participants to be knowledgeable of their lyrics and melody. Other important point guiding our choice was the correct lyrics-to-audio alignment and the absence of syntactical problems. We manually controlled the alignment quality of this subset by visualizing their lyrics in the karaoke prototype and eliminated poorly aligned songs from our selection. To avoid a learning effect of the song, each song can be selected once for a trial and can only be listened to twice during a trial. Moreover, the order of the songs in the selection menu for karaoke is randomized for each trial.

3.2 Influence of global factors

3.2.1 Experiment design

In this experiment, each participant is asked to choose 14 songs from the dataset from which karaoke excerpts are presented. Each audio extract lasts 35 seconds and consists of a sequence of words within lyrical lines, highlighting each word subsequently according to their aligned onset times. A lyrics-to-audio alignment error is generated for each user-song pair randomly from a set of positive and negative offsets between the audio and the lyrics displayed on screen. The offset is fixed for the whole sequence, which means all words in the stimulus are shifted by the same amount. At the end of each trial, participants are asked to report whether they perceive an asynchrony between lyrics and audio with a ternary response ("lyrics ahead", "lyrics lagging", "synchronous"). This experiment has a repeated measure design, with lyrics-to-audio synchrony perception as a dependent variable, and the lyrics-to-audio error offset as the independent variable having 14 modalities. It aims to measure an overall threshold of lyrics-to-audio synchrony perception and to study the influence of global rhythmic factors on this threshold, such as the tempo and word rate. If our theoretical hypothesis is confirmed, we expect to observe a greater proportion of "synchronous" responses for lyrics ahead than lyrics lagging, as well as a modulation of the perceptual threshold with the global rhythmic context (tempo, word rate).

3.2.2 Choice of offsets

In order to precisely define a threshold, we use a wide range of 14 offsets from -1s to 1s with negative offsets corresponding to lyrics ahead and reversely positive offsets mean lyrics lagging behind audio. We intentionally keep this number as small as possible, since this value is equal to the number of annotated songs required for each participant. Meanwhile, we wish to highlight effects around the commonly used threshold of 0.3s and -0.3s. Thus we use smaller steps around these values. We also included larger offsets (1s, 0.75s) as control values, to test that individuals systematically report those as asynchronous. In the same spirit, we expect the offset value 0 to trigger "synchronous" answers. The full experimental protocol was carefully tested beforehand with user testing sessions on six people. Based on these test results, we evaluated that completing the annotation required approximately 12 minutes per participant. Overall, the experiment involved 53 participants who completed the task.

3.3 Influence of local factors

In this second experiment, we make some changes in the karaoke interface. This time, we require each participant to choose one song from the dataset from which 10 audio excerpts are presented with different audiovisual offsets. Each sample is composed of three lyrical lines from the given song. The experience can be repeated multiple times with additional songs if desired. Each song takes around 3 to 5 minutes to annotate. Whilst in the first experiment the alignment errors were located on all the words of the sentence, in the second experiment, the position of the error may be located on the first, the last word of the sentence, or close to a beat. These choices are driven by some of the significant factors of stress described in Section 2 namely alignment with beats [21] and sequence boundaries [21,28]. We decided to discount the influence of long syllables [26] and loudness [27] for this study. In fact, long syllables and loud words are found to be overlapping respectively with the last word of the sentence and words closed to beats. The perceived synchrony is reported as a binary response ("yes", "no") with confidence on a 5-point Likert scale. This experiment intends to quantify the interaction of the error location in the sentence and the offset on the perceived alignment. It has a factorial design with the lyrics-to-audio offset and the position of the error as within subject factors. If our theoretical hypothesis is confirmed, we expect to observe a modulation of the perceptual threshold with the location of the error in the sequence.

Proximity of a word to a beat is defined as at a distance less than a *sixteenth note* from the beat, computed as 𝅘𝅥𝅯 = 15/ Beats Per Minute (BPM). The tempo estimation relies on Anssi Klapuri's algorithm, which showed 80% accuracy with constant tempo during the International Society for Music Information Retrieval (ISMIR) 2004 tempo induction challenge [29]. Starting from the baseline threshold of synchrony perception established in the previous experiment, the second experiment focuses only on lyrics lagging with 3 offsets (0.25, 0.5, 0.75) and a control sample with no offset. We chose only positive offsets because of practical constraints. Indeed, applying a negative offset at the word level can (and does frequently) result in overlapping with previous words, at least for beat-aligned and end words. Filtering out cases of overlapping words resulted in an important selection bias toward very slow songs. To avoid that, we could apply linearly decreasing offsets to precedent words until no overlap remains, as a naive way to "catch-up" with the true annotation. Such behaviour is consistent with what is observed in errors made by lyrics-to-audio alignment systems, multiple errors on consecutive words being recurrent. The concern was that we would not control which first offsetted word the participant would be confronted with. We decided to not consider negative offsets in this experiment but the problem of overlapping words for positive offset remains. However, after applying linearly decreasing offsets to consecutive words until no overlap occurs, the first offsetted word to which each participant is confronted remains the word of interest. Ultimately, we collected 2458 annotations from 193 participants.

4. RESULTS

As we intend to compute an overall threshold of synchrony perception, we perform the analysis at the level of the aggregated results, considering all annotations from all users. We removed all the trials from participants who did not answer correctly to our control levels i.e. "non synchronous" at 1s and "synchronous" at 0s. This represented precisely 11% of answers for the first experiment. We conducted a similar cleaning phase for users of the second experiment using the control offset of 0 and removed 8% of answers.

4.1 Asymmetry of Lyrics-to-Audio Alignment Perception

Using the data collected in the first experiment, we compute an aggregated proportion of respondents who indicate that lyrics and audio are "synchronous", and display it as a function of the lyrics offset in Figure 2. We see that synchrony perception is typically asymmetric, positive offsets being more easily detected than negative ones. This was expected as it resonates with previous findings [18, 19]. Beyond aggregated data, we also looked at individual responses and found that the thresholds were indeed asymmetric for 72% of individuals.

To give perspective, we plot the window function that

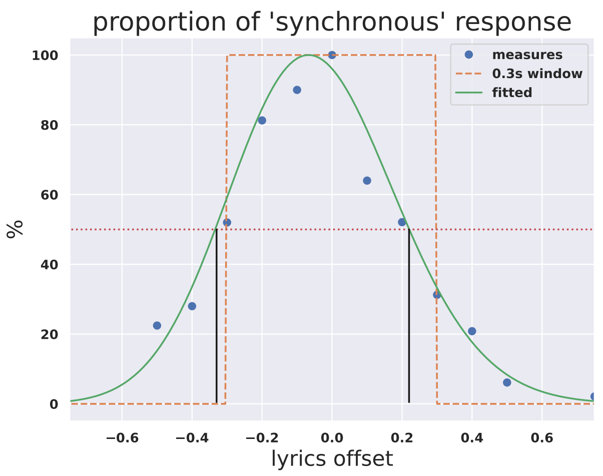


Figure 2. Aggregated results of *synchronous* judgment as a function of the lyrics/audio offset.

correspond to the PCO metric scoring as used in MIREX, with an absolute threshold value of 0.3s. We also fit a function akin to a scaled skew normal distribution function to the data points. Among several attempts with asymmetrical continuous functions, this was the best fit we obtained, although it does not respect the maximality at 0. Parameters of the fitted function are a skewness factor of 1.12, a location of −0.22 and a scale of 0.29, a multiplicative factor is also applied in order to have a value of 1. at the maximum. Using this function we can derive new perceptive thresholds for synchrony using a simple rule of 50% of respondents being able to detect the offset. For lyrics ahead and lyrics lagging we respectively identify the offsets −0.33s and 0.22s. Given the amount of noise in the data, we can reduce these to −0.3s and 0.2s and examine if the differences of perception are significant for these values. Indeed, pairwise tests revealed a significant difference of proportions of response "synchronous" on the levels −0.3 and 0.3s ($\chi^2(1)$ = 4.26, p = .038), while proportions on the levels −0.3s and 0.2 are not statistically different ($\chi^2(1)$ = 0.04, p = .08).

4.2 Sensitivity to global rhythmic context

In order to assess whether there is an influence of the global rhythmic context on lyrics-to-audio alignment perception, we compared the distribution of "synchronous" responses at each offset for two rhythmic factors: tempo and Words Per Second (WPS). We split our dataset of songs into two classes of tempo, defined as the upper and lower quartiles of the distribution of tempo, respectively fast (≥138BPM) and slow (≤93BPM). Although it is correlated with tempo, we also consider the average WPS rate of songs as a meaningful global factors. Again, we look at the first and last quartiles as Low (≤1.16WPS) and respectively High WPS (≥1.2WPS) classes. Figures 3 and 4 show the aggregated reported synchrony profiles for the negative offsets for the derived tempo and WPS classes. On both metrics, we observed no threshold discrepancy between the two classes for positive offsets.

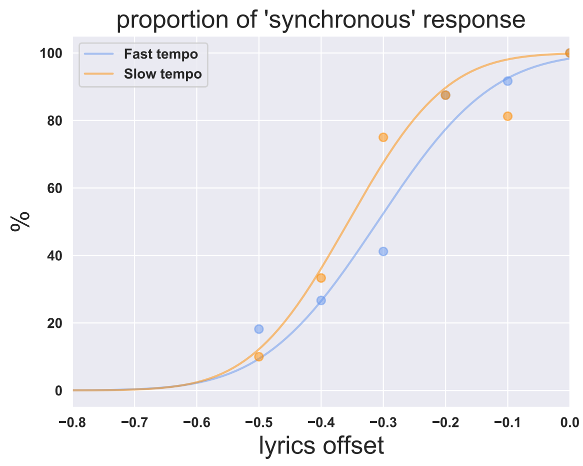


Figure 3. Proportion of response "synchronous" by tempo class.

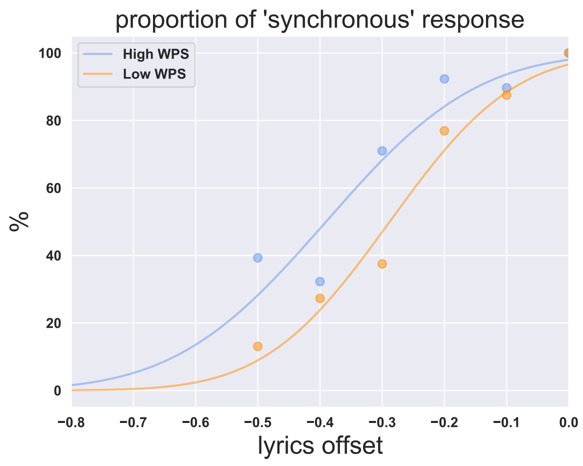


Figure 4. Proportion of response "synchronous" by WPS class.

To highlight the differences between classes, we fit a relatively simple sigmoid function to the data points. Among several candidates, a Gauss error function seemed most appropriate. Fitted functions are also displayed on Figures 3 and 4 and emphasize the different synchrony slopes. As before, we particularly consider the offset value intersecting with an average of 50% of "synchronous" responses as an indicator of participants' sensitivity to temporal asynchronies. Interestingly, the 50% threshold for the perception of synchrony is located at a larger offset (-0.36) for slow tempo than in fast tempo (-0.31). These results show that individuals report more frequently lyrics ahead as synchronous with slow tempo than with fast tempo. The lower sensitivity to lyrics-to-audio alignment errors in slow tempo is consistent with the results of [24]. Significance of these results are tested. The proportions of "synchronous" response at the offset -0.3s show significant difference between songs with high and low tempo ($\chi^2(1) = 5.44$, and $p < .02$).

Analogously, we found out that subjects are more tolerant to lyrics ahead (audio lagging) in high word rate than in low word rate. The 50% threshold for the perception of synchrony is indeed located at a larger offset for high word rate (-0.39) than in low word rate (-0.28) (Figure 4). These results show that subjects are more tolerant to lyrics ahead (audio lagging) in high word rate than in low word rate. We again tested the significance of these results. The proportions of "synchronous" response at the offset -0.3s are significantly different between songs with high and low WPS ($\chi^2(1) = 16.86$, and $p < .00004$).

4.3 Interaction between offset and word position

We designed the second experiment to distinguish perception of asynchrony as a function of the words position in the sentence. As explained in Section 3.3, we are only able to test for positive offsets. As insights from the previous experiment, we can assume that user sensibility is less affected by global factors for positive offsets. As a result, we expected it to be challenging for local factors too. For this reason, we aimed at collecting a much larger set of annotation, with a reduced set of tested offsets.

Figure 5 presents an overview of the results. There is a fairly large amount of noise in the collected data points, and few clear differences between synchrony perceptions for the three classes of word positions. The noise is particularly clear from the displayed level of confidence of participants who were unable to detect the asynchrony even for large values of the offset, but still were quite confident about their choice (average around 3.8). Regarding the location of the alignment error within the sentence, Cochran's Q test did not indicate a notable difference among the proportions of synchrony responses reported for the three error positions, $\chi^2(2) = 5.77$, $p = .056$.

The only visible effect seems to be for words aligned on *beats*, for which the confidence in the "asynchronous" answer at the 0.25 level is markedly higher than for the *end* class. More precisely, a Wilcoxon signed-rank test revealed that lyrics-to-audio alignment comparing error located on the beat with those on the last word did elicit a statistically significant change in the reported confidence of perception of error in individuals at the 0.25 level ($Z = 2.756$, $p < 0.006$). Indeed, mean confidence rating was 4.1 for error on the beat and 3.5 for error on the last word of the sentence. Such phenomenon is not observed for the synchronous case.

5. DISCUSSION

5.1 General discussion

Building on psychological theory and previous studies, we had hypothesized that a perceptual evaluation of lyrics/audio alignment quality would be asymmetrical and depend on both global and local factors. Using a first experiment we did find strong evidence for asymmetry and, to some extent for global factors influence. Despite a much larger experimental setup which involved hundreds of participants, we were not able to exhibit a clear influence of the local factors we tested. This negative result could mean

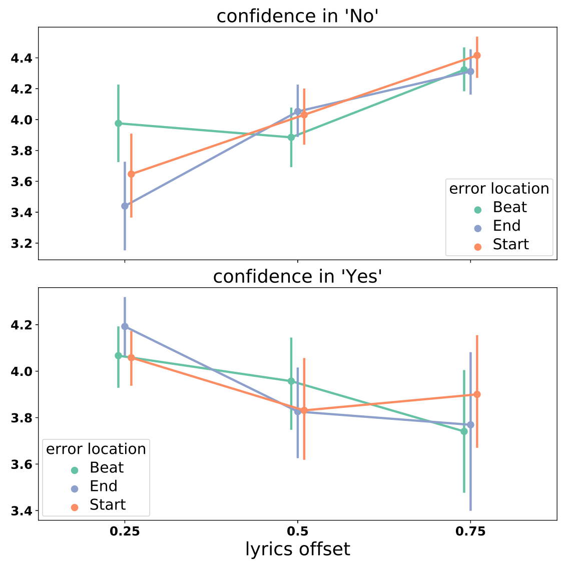


Figure 5. Plot of the reported answer (synchronous is "yes", asynchronous is "no") and confidence score (5 level Likert scale) in the perception of synchrony, by location of alignment error within the sentence.

that the local factors considered, i.e. the word position in the lyrical line are not the relevant ones. It is possible that words grammatical or semantic functions are more subject to human attention in a karaoke context. Indeed, the only significant phenomenon that we observed was on words located on *beats*, for which the asynchrony perception was more acute. Future work should investigate the relationship between rhythmic position and lyrical function of words and test new hypothesis of perceptual differences. Finally, although we did our best to build a realistic yet controlled experimental setup, we acknowledge that as a psychological experiment mostly conducted online, we can not completely rule out the possibility that the measurement noise was too high to allow us to detect signals on local factors.

There are arguably other factors that could influence this perception, notably at the human level. Indeed, familiarity with the song (e.g. previous knowledge of the lyrics and/or the music), but also participants' facility with the languages, level of musical expertise and even karaoke practice could be important variables to consider. In the conducted experiment, we collected such information from participants. Although we did observe some interesting phenomenon, for the sake of clarity, we chose not to present additional results on these variables here and leave it to a follow-up study.

5.2 Implication for evaluation metrics

Here we would like to present a practical use of our results, as a perceptually motivated evaluation metric for lyrics to audio alignment tasks. Overall, we propose a generalization of the PCO metric in the following form:

$$\psi^k = \frac{1}{N_k} \sum_{word\ i} f(\hat{t}_i - t_i) \times 100 \qquad (2)$$

	PCO	Asym-PCO	Perc-PCO
Gupta [13]	**94.47 (1.52)**	**93.66 (1.59)**	**89.94 (1.71)**
Vaglio [15]	91.85 (1.95)	90.82 (2.04)	86.79 (2.13)
Stoller [14]	87.02 (2.97)	85.23 (3.07)	79.93 (2.90)

Table 1. Averaged metrics over the Jamendo dataset songs. Standard errors are given in parenthesis.

where the function f can be seen as penalty weighting of the annotation offset and other notations are common with Equation 1. We then evaluated 3 state-of-the-art automatic lyrics-to-audio alignment models [14, 15, 30], on the 20 songs of the Jamendo dataset [14]. We have compared using the regular PCO ($f = 1_{[-0.3,0.3]}$), a slightly modified version still using a square window but taking into account the asymmetrical perception ($f = 1_{[-0.3,0.2]}$) and a Perceptual-PCO function that is the one fit from the data collected in our first experiment and depicted in Figure 2. This function can be seen as a smooth relaxation of the square window, taking into account the perceptive asymmetry of the error slopes.

Results are compiled in Table 1. Interestingly, there appears to be little difference between using the standard PCO window and a slightly shifted one. However, scores for the perceptual-PCO are much lower. This is despite the window support being larger (i.e. errors of more than 0.3s are not completely nullified). In our opinion, this new metric is better suited to capture the relative importance of alignment errors and weights them according to human perception. It can also help for comparing between alignment methods that achieve near perfect scores with the standard PCO. It is worth noticing that although we demonstrated it on the PCO, a similar weighting could be applied to other alignment metrics. A step further would be to parameterize the window function f on global song factors such as tempo and WPS. This would arguably require additional experiments with a larger, more diverse set of songs.

6. CONCLUSION

In this work, we challenged the objective evaluation of lyrics-to-audio alignment using hypothesis from psychological theory. We postulated three effects: asymmetry, influence of songs features and influence of words local positions. We were able to demonstrate the first two effects using a large scale online experiment, disguising the synchrony annotation task as a Karaoke experience. This framework proved less efficient for the third effect, despite our efforts to collect up to several thousands annotation points. We nonetheless proposed a readily usable weighting function to allow finer comparison between state-of-the-art alignment methods. Future work will investigate more diverse sets of factors, both on musical attributes and user features.

7. REFERENCES

[1] A. Adadi and M. Berrada, "Peeking inside the black-box: a survey on explainable artificial intelligence (xai)," *IEEE access*, vol. 6, pp. 52 138–52 160, 2018.

[2] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proc. of the 40th annual meeting of the Association for Computational Linguistics (ACL)*, 2002, pp. 311–318.

[3] L. Yin-Jyun, C. Ming-Tso, and C. Tai-Shih, "Singing voice correction using canonical time warping," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2007, pp. 396–400.

[4] C. Pei-Chun, L. Keng-Sheng, and C. Homer, "Emotional accompaniment generation system based on harmonic progression," *IEEE Trans. on Multimedia*, vol. 15, no. 7, pp. 1469–1479, 2013.

[5] A. Huang and R. Wu, "Deep learning for music," *arXiv preprint arXiv:1606.04930*, vol. abs/1606.04930, 2016.

[6] A. Holzapfel and E. Benetos, "Automatic Music Transcription and Ethnomusicology: a User Study," in *Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2019, pp. 678–684.

[7] C. Raffel, B. Mcfee, E. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. Ellis, "mir_eval: A transparent implementation of common mir metrics," in *Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 10 2014.

[8] X. Hu and N. Kando, "User-centered Measures vs. System Effectiveness in Finding Similar Songs." in *Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, Oct. 2012, pp. 331–336.

[9] A. Daniel, V. Emiya, and B. David, "Perceptually-based evaluation of the errors usually made when automatically transcribing music," in *Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2008.

[10] A. Ycart, L. Liu, E. Benetos, and M. Pearce, "Investigating the perceptual validity of evaluation metrics for automatic piano music transcription," *Trans. of the Int. Soc. for Music Information Retrieval (TISMIR)*, vol. 3, pp. 68–81, 2020.

[11] A. Ycart, L. Liu, E. Benetos, and M. T. Pearce, "Musical features for automatic music transcription evaluation," *arXiv preprint arXiv:2004.07171*, 2020.

[12] E. Vincent, "Improved perceptual metrics for the evaluation of audio source separation," in *Int. Conf. on Latent Variable Analysis and Signal Separation (LVA/ICA)*. Berlin, Heidelberg: Springer-Verlag, 2012, p. 430–437.

[13] C. Gupta, E. Yılmaz, and H. Li, "Automatic lyrics transcription in polyphonic music: Does background music help?" in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[14] D. Stoller, S. Durand, and S. Ewert, "End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 181–185.

[15] A. Vaglio, R. Hennequin, M. Moussallam, G. Richard, and F. D'alché-Buc, "Multilingual lyrics-to-audio alignment," in *Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2020.

[16] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "Creating dali, a large dataset of synchronized audio, lyrics, and notes," *Trans. of the Int. Soc. for Music Information Retrieval (TISMIR)*, vol. 3, no. 1, 2020.

[17] M. Mauch, H. Fujihara, and M. Goto, "Integrating additional chord information into hmm-based lyrics-to-audio alignment," *IEEE Trans. on Audio, Speech, and Language Processing (TASLP)*, vol. 20, no. 1, pp. 200–210, 2012.

[18] R. L. J. van Eijk, A. Kohlrausch, J. F. Juola, and S. van de Par, "Audiovisual synchrony and temporal order judgments: effects of experimental method and stimulus type," *Attention, Perception, & Psychophysics*, vol. 70, no. 6, p. 955–968, 2008.

[19] A. Vatakis, B. Fuat, D. L. Massimiliano, and C. Ángel, *Timing and Time Perception: Procedures, Measures, & Applications*. Brill, 2018.

[20] B. Repp and A. Penel, "Auditory dominance in temporal processing: New evidence from synchronization with simultaneous visual and auditory sequences," *Journal of experimental psychology. Human perception and performance*, vol. 28, pp. 1085–99, 11 2002.

[21] M. R. Jones and M. Boltz, "Dynamic attending and responses to time," *Psychological Review*, pp. 459–491, 1989.

[22] K. Dunlap, "Reaction to rhythmic stimuli with attempt to synchronize," *Psychological Review*, vol. 17, pp. 399–416, 1910.

[23] B. Repp, "Sensorimotor synchronization: A review of the tapping literature," *Psychonomic bulletin & review*, vol. 12, pp. 969–92, 01 2006.

[24] E. T. Klemmer, "Simple reaction time as a function of time uncertainty," *Journal of experimental psychology*, vol. 54, no. 3, pp. 195–200, 1957.

[25] J. D. McAuley and N. S. Miller, "Picking up the pace: Effects of global temporal context on sensitivity to the tempo of auditory sequences," *Perception & Psychophysics*, vol. 69, pp. 709–718, 2007.

[26] F. Lerdahl and R. S. Jackendoff, *A Generative Theory of Tonal Music*. The MIT Press, 06 1996.

[27] A. M. C. Sluijter, V. J. van Heuven, and J. J. A. Pacilly, "Spectral balance as a cue in the perception of linguistic stress," *The Journal of the Acoustical Society of America (JASA)*, vol. 101, no. 1, pp. 503–513, 1997.

[28] T. R. Knösche, C. Neuhaus, J. Haueisen, K. Alter, B. Maess, O. W. Witte, and A. D. Friederici, "Perception of phrase structure in music," *Human Brain Mapping*, vol. 24, no. 4, pp. 259–273, 2005.

[29] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, "An experimental comparison of audio tempo induction algorithms," *IEEE Trans. on Audio, Speech, and Language Processing (TASLP)*, vol. 14, pp. 1832–1844, 2006.

[30] C. Gupta, H. Li, and Y. Wang, "Automatic pronunciation evaluation of singing," in *Int. Speech Communication Association (INTERSPEECH)*, 09 2018, pp. 1507–1511.

THE WORDS REMAIN THE SAME: COVER DETECTION WITH LYRICS TRANSCRIPTION

Andrea Vaglio[12] **Romain Hennequin**[1] **Manuel Moussallam**[1]
Gaël Richard[2]
[1] Deezer R&D
[2] LTCI, Télécom Paris, Institut Polytechnique de Paris
research@deezer.com

ABSTRACT

Cover detection has gained sustained interest in the scientific community and has recently made significant progress both in terms of scalability and accuracy. However, most approaches are based on the estimation of harmonic and melodic features and neglect lyrics information although it is an important invariant across covers. In this work, we propose a novel approach leveraging lyrics without requiring access to full texts though the use of lyrics recognition on audio. Our approach relies on the fusion of a singing voice recognition framework and a more classic tonal-based cover detection method. To the best of our knowledge, this is the first time that lyrics estimation from audio has been explicitly used for cover detection. Furthermore, we exploit efficient string matching and an approximated nearest neighbors search algorithm which lead to a scalable system which is able to operate on very large databases. Extensive experiments on the largest publicly available cover detection dataset demonstrate the validity of using lyrics information for this task.

1. INTRODUCTION

Cover detection, also known as version identification, aims at detecting whether two recordings are of the same underlying musical work. A cover can be played by the same artist as the original song, or by another artist, and can be quite similar or vastly different. Generally, it is assumed, as in [1], that tonal progression features (chord, melody, and harmony) are mostly preserved between covers of the same work. Inversely, musical attributes such as key, timbre, tempo, and structure significantly vary across covers [1]. Variations of these features between covers were extensively studied in [2]. Cover detection systems are then built to be insensitive to these variations and exploit tonal progression features. The task has been frequently studied as a *query and answer* [1] one, i.e. given an input query, the system outputs a ranked list of possible covers from a music collection. True covers are to be ranked as highly as possible while other songs should be ranked low. This list is usually obtained by computing pairwise similarities between the query and each song of a pre-defined dataset [1]. If earlier cover detection systems were shown to be highly efficient on small datasets (1000 songs or less) [3], performances quickly dropped on larger ones [2, 4]. Recent works have made significant advances in scalability and accuracy, for larger datasets, taking inspiration from metric learning [5] and knowledge distillation [6].

Almost none of the existing approaches explicitly consider the textual information provided by the lyrics. To the best of our knowledge, it is only used in [4], in which lyrics are assumed to be available for a significant part of the dataset. In this paper, the authors use metadata and lyrics alongside audio to perform cover detection. The textual similarity of lyrics and song titles is computed using a plain Bag-of-word *Term Frequency–Inverse Document Frequency* (TFIDF). The authors show that results obtained with lyrics are on par with those given by audio-based features on a large-scale dataset. Moreover, the best results are obtained when combining all of the features. However, each feature is only used in a separate part of a multi-layer database pruning method; the information carried by each modality is thus not optimally combined. One limitation of this work is that it assumes that the lyrics of most songs are available. Considering the task of query by singing, which may be regarded as a related task to the cover detection one, authors in [7] employed lyrics and melody recognition to recognize a singing query and match it against a collection of songs. They employed a basic bigram *Hidden Markov model* (HMM) model that is trained on speech and adapted to singing voice. However, this approach also presupposes that the lyrics of songs are available. Lyrics from the considered dataset are, in fact, utilized to inform singing voice recognition.

While this assumption arguably does not hold for large musical collections, one could turn to *Singing Voice Recognition* (SVR) frameworks to retrieve a noisy estimate of the lyrics. We thus propose a novel cover detection approach leveraging lyrics information extracted from audio. It is based on the fusion of a SVR framework and a more classic tonal-based cover detection system. Based on our review of the literature, this is the first time that an estimation of lyrics transcripts from audio has been explicitly leveraged to perform cover detection. Our assumption, based on the results of [4], is that lyrics are often preserved between covers in popular western music. For the first modality of our fused system, we thus propose using transcription methods to obtain estimates of these lyrics for all songs. The cover song here is framed as a noisy text-matching task. We expect a lyrics-recognition based system to be particularly relevant for pairs of covers displaying hugely different tonal features while using the same lyrics. An example of such cases is the cover of *Summertime* by *Janis Joplin* where the harmony and melody are considerably different from the original score, but the lyrics remain quite similar. Nevertheless, it is clear that a pure lyrics-based system is inadequate for instrumental music (e.g. without a singing voice). Therefore, we use a tonal-based system such as the second modality of our fused system. An instrumental detector is applied on the output of the lyrics recognition framework to inform the fusion strategy. We provide extensive empirical evidence that both modalities are indeed complementary. Extra attention is placed on the scalability of our proposed approach using *Approximated Nearest Neighbors* (ANN) methods.

2. RELATED WORKS

Classically, cover song detection systems use tonal features, which are thought to be the least altered between a song and its covers. Chroma [8] and derived features such as *Harmonic Pitch Class Profil* (HPCP) [3] and CremaPCP [5] are among most effective examples. Before computing the similarity between two songs, multiple preprocessing steps can be applied to obtain features that are invariant to the key [9], the tempo [10], or the structure of the song [5]. After extracting the features to be compared for both songs, a cross similarity matrix [11], or a cross recurrent plot [9], is then generally computed. A similarity score is then computed using dynamic programming like *Dynamic Time Warping* (DTW) [12] and recurrence quantification analysis [3]. For a given query, this score is calculated for all tracks in a pre-defined dataset and thus yields the desired sorted list. These methods achieve satisfactory results for small datasets of up to a thousand songs [3], but are computationally costly for larger datasets.

To address this issue, some authors have attempted to reduce the size of the input representation to obtain a low-dimensional fixed size representation for each track. The similarity comparison thus boils down to a basic distance metric such as Euclidean distance or cosine similarity [5] that are much faster than dynamic programming algorithms of quadratic complexity. Early approaches of this type include using fingerprinting in the form of Chroma landmarks [10] and $2D$ Fourier transform of Chroma vectors [13], both obtaining low performances. More recent approaches using metric learning, triplet loss, and distillation methods show greater improvement [5, 6] in terms of computation speed and retrieval performances. Database pruning was also used to decrease the overall complexity in [4, 14]. A first fast global candidate selection using text and metadata was performed, followed by a more

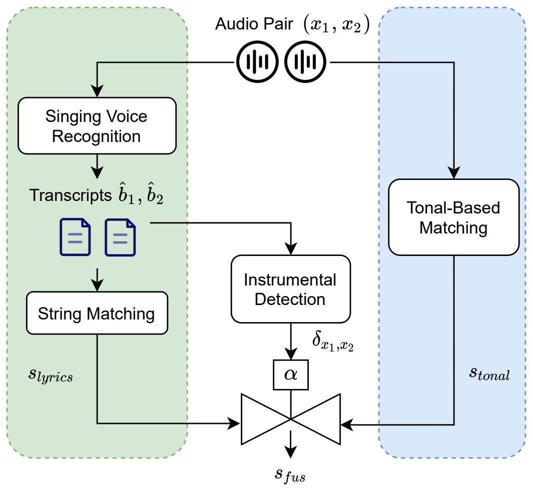


Figure 1. Audio from a pair of tracks is processed in parallel by two branches computing lyrics and tonal-based similarities respectively. The fusion mechanism is informed by an *instrumental* detection on the transcripts

complex similarity function to re-rank the subset. Most of these approaches, which are based on low-dimensional embeddings and simple distance functions, are then simply exploited into existing scalable nearest-neighbors methods. For example, the authors in [10, 15] use index-based matching on extracted audio fingerprinting. Scalable nearest-neighbors methods are more broadly discussed in Section 3.6.

3. PROPOSED APPROACH

A general overview of our approach is described in Figure 1. It is composed of a lyrics-recognition based cover detection system and a classic tonal-based system. The first branch is constituted of a lyrics recognition framework and a string matching function. It takes two songs x_1 and x_2 as input and outputs the respective estimated lyrics $\hat{b_1}$ and $\hat{b_2}$. A similarity estimation s_{lyrics} is then obtained using these transcriptions. The second branch of our approach, the classic tonal-based system, also takes these two songs as input and outputs a similarity estimation s_{tonal}. They are then fused using a fusion function to obtain a new similarity estimation s_{fus}. Extra input is added to the fusion function α to weigh the participation of both modalities during the fusion. The value of this input depends on the instrumental detector taking as input both transcripts and outputting the probability that at least one of the tracks is purely instrumental. This avoids using the lyrics-based recognition system during the fusion in the absence of lyrics. To obtain the desired sorted list, for a given query, a similarity is then computed for the considered system between the query and each track of the dataset. Finally, a fast approximate index search technique is used on our system to make it scalable. In our work, we rely on the ANN approach where the similarity is only computed between the query and the nearest neighbors returned by the method.

3.1 Lyrics recognition

We choose a state-of-the-art framework [16] that obtained the best results in the *Music Information Retrieval Evaluation eXchange* (MIREX) 2020 lyrics transcription challenge [1]. It uses an acoustic model composed of several layers of *Time Delay Neural Network* (TDNN) that are trained using the English tracks of the DALI dataset [17]. Background music is directly modeled as an output of the acoustic model as such that it does not use any preprocessing step of *Singing Voice Separation* (SVS). Moreover, phoneme units are annotated with genre labelling information. An extended lexicon is also employed to handle long-vowel duration. Finally, a 3-gram word language model with interpolated Kneser-Ney smoothing is trained on the English portion of DALI lyrics. The complete framework extracts *Mel-Frequency Cepstral Coefficients* (MFCC) of dimension 40 from the input audio and outputs transcribed English words. As this model cannot output non-English words, extra care on the results of non-English tracks will be considered later in this paper. The acoustic model and lexicon are collected from the code implementation of the authors [2]. We compute the language model with the kenLM toolkit [18]. The vocabulary of the language model is restricted to the 6000 most frequent words, thus reducing overfitting. Obtained transcription results are on par with those in MIREX with a *Word Error rate* (WER) of 62% on the Jamendo dataset [19].

3.2 String matching

To allow for a swift computation of the similarity between pairs of estimated transcripts, each string is transformed to a vector using a TFIDF based on a 3-gram at character level with IDF values computed from the DALI dataset. The complexity of this type of algorithm is $O(m + n)$ with m and n, which are the respective length of each transcript. The similarity is then simply given using a cosine similarity, which is independent of the length of each transcript. Using a word level 3-gram was not shown to improve performances on a cover song tuning set described in Section 4. We also considered the Levenshtein distance for the string matching, but it did not yield significant gains in performances while inducing a quadratic complexity.

3.3 Detecting instrumentals

Looking at various transcripts given by our SVR framework, we notice that, for most instrumental tracks, the transcript obtained is composed of either a very few number of words, or highly repeated ones such as onomatopoeia. Therefore, we consider a track as instrumental if the respective transcription is composed of less than l different words with l tuned on the cover song tuning set. The module outputs $\delta_{x_1,x_2} = 1$ if both tracks are not detected as instrumentals, and 0 otherwise. For some rare cases where the SVR framework is truly performing poorly, it is also only outputting a few words. The instrumental detector then helps with additionally filtering some marginal cases where the lyrics transcription fails completely. We chose to keep this very simple as it performed sufficiently well for our purposes and allowed for improvements in future works.

[1] https://www.music-ir.org/mirex/wiki/2020:MIREX2020_Results

[2] https://github.com/chitralekha18/AutoLyrixAlign

3.4 Tonal-based cover detection

The tonal-based cover detection method selected is described in [6]. This system, called Re-MOVE [6], is an updated version of MOVE [5] and obtains the second most accurate benchmark on the Da-Tacos dataset [2]. Compared to the best one reported [20], it has the advantage of being publicly available [3]. The system is trained using the training part of Da-Tacos, as described in Section 4.1, and early stopping is performed using its validation component. For a given track, the system takes CremaPCP extracted from the audio as input and outputs a corresponding compact embedding. The CremaPCP feature is an intermediate representation of a chord estimation model. It is considered an efficient way to capture the tonal information of music and is shown to outperform more classic HPCP features for cover detection [2]. The similarity between the query and each track of the dataset is then the cosine similarity of their respective embeddings.

The Re-MOVE system uses a latent space reconfiguration technique on top of MOVE in order to reduce the embedding dimension (and then reduce memory requirements and retrieval time) while maintaining high detection performances. This technique reconfigures a pre-trained learned distance metric into a more compact embedding space with the same learned semantic relation.

3.5 Fusion

It has been shown in multiple domains that the fusion of different modalities can yield better performances than those obtained with each single modality [21]. For cover detection, fusing modalities, features or similarities matrix have already shown to improve results [22,23], notably using rank aggregation methods [24]. The fusion function chosen here is a weighted sum. It is more precisely described by:

$$s_{fus} = \begin{cases} \alpha s_{lyrics} + (1-\alpha) s_{tonal} & \text{if } \delta_{x_1,x_2} = 1 \\ s_{tonal} & \text{otherwise} \end{cases} \tag{1}$$

α is a simple scalar defined as an hyperparameter to tune. As the distributions of both similarities are very different, calibration before fusion was also tested. However, no improvement was shown on the cover song tuning set. Other fusion functions, such as linear regression or max function, did not lead to improvements in our simulations.

3.6 Scalability

Pairwise comparisons between a given query and all tracks in a dataset are linearly dependent on the size of the dataset

[3] https://github.com/furkanyesiler/re-move

without optimization, which cannot be considered scalable. In fact, a linear complexity for the queries involves a quadratic complexity for retrieving all musical works in the dataset, which can quickly become prohibitive for large collections. To achieve better scalability properties, most cover detection studies use ANN methods such as *Locality Sensitive Hashing* (LSH) [25, 26]. The idea behind ANN is that for a given query x and a database D the method outputs an approximation of the k nearest neighbors of the query in the database with the complexity being sublinear in the size of the database. For a given query, in contrast with classic *K-Nearest Neighbors* (KNN), ANN methods are only browsing a subset of the complete search graph. All these methods are based on an index table allowing fast queries by outputting a "good" guess of the k nearest neighbors of a given query, making it possible to recover the most highly classified covers in the ranked list obtained with all candidate points. The recall is used to quantify the quality of an ANN method by averaging percentages obtained, for various queries, of true k-nearest-neighbors from k points returned by the method. In our case, we use the Hierarchical Navigable Small World Graph (HNSW) state-of-the-art ANN method; an extensive study of it is given in [27]. This algorithm gives logarithmic complexity for a query in terms of the size of the dataset. This method is directly applied on Re-MOVE and TFIDF embeddings, outputting for a given query k nearest neighbors for each of them. Both sets of points are then concatenated and merged, obtaining a maximum of $2k$ points to consider for the fusion. Pairwise similarities between the query and these points are then generated using the Re-MOVE system and our lyrics-recognition pipeline.

4. EXPERIMENTAL EVALUATION

4.1 Dataset

Da-Tacos [2, 6] is the largest publicly available dataset for cover detection; the training set is composed of 83904 songs in 14999 cliques and the validation set of 14000 songs in 3500 cliques. A clique is defined as a cover group gathering multiple recordings of the same underlying "piece". The Da-Tacos benchmark test subset is a 15000 tracks dataset composed of 1000 cliques with 13 songs each and 2000 noise songs (i.e. that are in a single-song clique) that are not queried. To avoid overfitting, no clique overlaps with any set of Da-Tacos. Instrumentals represent around 20% of the dataset which motivates our choice of using an instrumental detection process. Currently, only a set of precomputed audio features are publicly available for the benchmarking subset test dataset. The dataset is mainly composed of English tracks and popular western music with a few non-English cliques. All hyperparameter tuning made during this paper is carried out on a subpart of the Da-Tacos validation that we choose to refer to as a *Da-Tacos tuning* set. We verified that no clique of this subset overlaped with any clique present in the dataset used to train the tonal-based cover detection system, i.e. the Da-Tacos training set. Also, a clique is discarded if it possesses one track present in the dataset used to train the SVR framework, i.e. DALI dataset. Detection of overlapping tracks and cliques is made using metadata, *i.e.* titles and artists names. *Da-Tacos tuning* is notably used to choose the string matching algorithm and the fusion function. We recover audio of 12862 tracks from the test dataset. 1849 are in single-song cliques and thus are not queried and only used as noise songs. We make sure no clique of this dataset overlaps with cliques in the Da-Tacos train and validation and that cliques possessing tracks existing in the DALI dataset are discarded. It will be simply referred to as *Da-Tacos test* for the rest of the document.

4.2 Fusion parameters

A track is classified as instrumental if the number of different words of its transcript is less than $l = 8$. This number is adjusted using *Da-Tacos tuning* as the value that maximizes the recall for the highest $F1$ score. Emphasis is put on the recall in order to avoid taking into account the lyrics-recognition based similarity for an instrumental track that has been misclassified as non-instrumental. An α value of 0.6 for fusing both system is tuned on *Da-Tacos tuning*.

4.3 Parameters of ANN

We use the HNSW implementation of the NMSLIB similarity search library [28]. For each query, we return the $k = 100$ nearest neighbors. This choice is derived from [4], which shows performance does not evolve significantly after the top-100 pruning. We use an approximated cosine similarity function to retrieve 100 candidates for each branch which results in, at most, 200 items for the fused model after concatenating and merging both sets.

4.4 Evaluation

The empirical evaluation of the cover detection task performances is given using the *Mean Average Precision* (MAP) [4]. For a query, *Average Precision* (AP) is quantifying the number of actual covers that are highly ranked. The AP score increases when actual covers are detected in the top ranks. The MAP is then simply obtained by averaging on the AP of all queries. As the MAP is not properly defined for systems that do not score every track (such as ANN), we report MAP@100 for these cases considering only the top-100 ranked item of each query. In any case, the MAP does not significantly evolve after the top-100 pruning as explained in the previous section.

5. RESULTS AND DISCUSSION

5.1 Lyrics-recognition based system results

5.1.1 Instrumental detection

Among the 12862 tracks in the test set, 3269 are detected as instrumentals. We compared this with the "Instrumental" tag available in the Da-tacos for all tracks. We obtain a

[4] Computed using the Metrics toolkit from `https://github.com/benhamner/Metrics`

Query	System	MAP (%)
Da-Tacos-voice	Lyrics	**66.4 (0.4)**
	Tonal	54.0 (0.4)
Da-Tacos-instr	Lyrics	0.45 (0.06)
	Tonal	**47.8 (0.7)**

Table 1. Results of lyrics-recognition based and tonal-based cover detection system on *Da-Tacos-voice*. *Da-Tacos-instr* is the subset of the *Da-Tacos test* restricted to instrumental tracks. Standard errors are given in parenthesis

Dataset	System	MAP (%)
Da-Tacos test	Fused	**62.7 (0.3)**
	Fused-wo-inst	50.2 (0.3)
	Tonal	50.6 (0.3)
Da-Tacos-voice	Fused	**80.4 (0.3)**

Table 2. Results of fused, with and without instrumental detection, tonal and lyrics-recognition based cover detection system on various datasets

precision of 82.86% for the instrumental detection, a recall of 96.68% and a F1 score of 89.24%. A closer look at misclassified tracks showed that there is some annotation noise in the Da-Tacos annotations which could artificially lower the previous metrics. As simple as it is, the instrumental detection performance seems suitable for our application. After filtering detected instrumentals, we obtain a subset of 9593 tracks that we label *Da-Tacos-voice*. 1582 tracks are in single-song cliques. 8011 tracks are then queried.

5.1.2 Lyrics-based cover detection

We first evaluate our lyrics-recognition based system on the *Da-Tacos-voice*. Our results, displayed in Table 1, show that it is generally performing better than the tonal-based one in terms of MAP. They validate the assumption that lyrics can be considered as a strong invariant between covers. It also proves that the most recent state-of-the-art singing voice recognition framework produces transcriptions of sufficiently good quality to perform the cover song as a noisy text matching task. Looking empirically at results coming from both systems, most improvements of the lyrics-recognition system over the tonal-based system come, as expected, from covers with highly different tonal-content and lyrics being roughly the same.

We also query the tracks detected as instrumental and not from single-song cliques. Results are also displayed in Table 1. As expected, performance of the lyrics-recognition based system is almost close to zero. For the tonal-based system, results seem to degrade when compared to non-instrumentals tracks. This suggests that the system has either learned characteristics of the melody carried by the singing voice or implicitly estimated some of the lyrics information to perform cover detection.

5.1.3 The case of non-English tracks

As stated in Section 3.1, our lyrics recognition framework cannot output non-English words, therefore non-English tracks may produce unexpected results. In order to assess the impact of this issue, we predicted a language label for every track of the *Da-Tacos-voice* using a language classifier [29] taking track metadata as input. Results show that the dataset is largely composed of English with more than 92.4% of the tracks being detected as English. Looking at tracks outside single-song cliques detected as non-

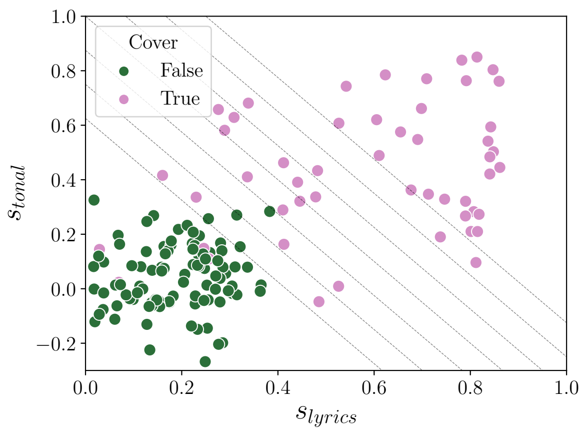


Figure 2. Similarities of sampled pairs of tracks from the *Da-Tacos-voice*. Here, each point is a pair of tracks. Each color indicates a same-clique belonging status. Some level curves of s_{fus} are also displayed

English, half of them are false positives. We query non-English tracks of the resulting 44 cliques on the *Da-Tacos-voice*, representing 299 tracks. It is interesting to report that almost all cliques are homogeneous in terms of language. We obtain a MAP of 28% (2). Results show that even if performances deteriorate for these cases, our system is often able to correctly classify these tracks. It can be explained as the chosen singing voice framework is transcribing something similar from one cover to another even for non-English lyrics. Considering the small quantity of non-English tracks and results on these tracks, we consider that this issue has a limited impact on performance in our evaluation setup.

5.2 Fused system results

Results for the fused system on the full *Da-Tacos test* and its *Da-Tacos-voice* subset are given in Table 2. The fused system significantly outperforms the results of the tonal-based one alone showing the validity of our assumption of both systems being highly complementary. The use of the instrumental detection module to inform the fusion strategy is empirically validated, with a major drop of performances occurring when it is not considered. The gain in performance comes essentially for increased accuracy on the *Da-Tacos-voice* subset, where information from both

System	SVR	MAP (%)
Lyrics	CTC	40.3 (0.7)
	Our	79.0 (0.6)
	Lyrics-informed	**89.7 (0.4)**
Fused	CTC	71.1 (0.6)
	Our	88.5 (0.4)
	Lyrics-informed	**93.6 (0.4)**

Table 3. Performances of lyrics-recognition and fused based cover detection system on *Da-Tacos-lyrics* with various SVR framework. Lyrics-informed framework are informed by lyrics at test time

branches is available and the MAP reaches around 80%. To highlight this complementarity, similarities for sampled pairs of tracks from the *Da-Tacos-voice* are displayed in Figure 2. While the majority of same-clique pair lyrics and tonal similarities are significantly higher than non-matching pairs, there are multiple cases where one modality seems more indicative than the other. Level curves of s_{fus} are also displayed illustrating most pairs being linearly separable in the combined modality plane.

5.3 ANN results

We first evaluate the impact of pruning results to the first 100 candidates by computing the MAP@100 of the fused system on the *Da-Tacos test* dataset. A small decrease is observed with a MAP@100 of 62.4% (0.3). After applying an ANN to our fused system, results remain the same with a MAP@100 of 62.4% (0.3) . This result can be explained as the recall of the HNSW for both a tonal-based and lyrics-recognition system being more than 99.5%. Thus, the scalability of our system is assured while maintaining the cover detection performances.

5.4 Impact of the SVR framework

A detailed analysis of failing samples of the lyrics-recognition based system shows that the main cause for failure is the low quality of the transcriptions. To further investigate this impact, we introduce two baselines by changing the SVR framework part of our system. In the first, an alternative *Connectionist Temporal Classification* (CTC) based SVR framework is used. The acoustic model of this framework is described in [30]. It consists of several *Bidirectional Long Short-Term Memory* (BiLSTM) layers, is trained on a multilingual subpart of the DALI dataset with a CTC algorithm and relies on a pre-processing step of singing voice separation. The language model used is the same as the one described in Section 3.1. Decoding is performed, after tuning the language model weight and insertion penalty value using a validation dataset, with a CTC beam search decoding tooklit[5] . The transcription of the results obtained on Jamendo dataset [19] are significantly lower than our current singing voice recognition framework with a WER of 84.4%. We thus expect this CTC-baseline to obtain results far below our system for cover detection tasks.

[5] `https://github.com/parlance/ctcdecode`

In the second baseline, we simulate an "ideal" SVR framework outputting an exact transcription. It can be considered as an oracle system, yielding an upper bound for performances of lyrics-recognition based systems. To compare these three systems, we retrieve the lyrics text information for part of the *Da-Tacos test*. The subset obtained is labeled *Da-Tacos-lyrics* and is composed of 3467 tracks for which we found matching lyrics. Considering that this subset only contains non-instrumental tracks, we discard the instrumental detector for this section. Again, tracks from single-song cliques are not queried and are used as noise songs.

The results obtained on *Da-Tacos-lyrics* are given in Table 3. These results confirm the intuition that the lyrics-recognition system's strength for covering detection task directly depends on the quality of the lyrics transcription. Ranking performances on *Da-Tacos-lyrics* for these systems are conserved after fusing them with the tonal-based branch. In comparison to the oracle system, our fused system shows excellent results even if there is still some room for improvement. With the transcription performances of our SVR framework being as low as 62% WER, it certainly indicates that a perfect transcription is not needed for the cover detection task. Interestingly, even an oracle system informed by the true lyrics benefits from being fused with a tonal-based one. This, once again, demonstrates both branches are acutely complementary to address the cover song detection problem. Future works will extend our system to take into account cases where lyrics are available for a part of the dataset.

6. CONCLUSION

Using only audio, we have proposed a framework that explicitly leverages two types of similarities, tonal and lyrics based, and reach high accuracy levels while remaining simple and scalable. With that said, work on more diverse data still remains to be done, notably on non-English tracks where performances seem to be limited.

Future work will include replacing the current monolingual lyrics recognition with a multilingual framework. A multilingual similarity, capable of detecting the similarity of two texts based on their semantic content, independently of their language, will also be defined and evaluated. More generally, the Da-Tacos dataset is quite biased towards popular western music. Additional experimentation on a wider range of genres, notably none western music, and cover types (e.g. karaoke, renditions, etc.) remains to be conducted. Finally, we will explore more elaborate fusion schemes, specifically, a mid-level fusion which can be further optimized and possibly lead to improved performance.

7. REFERENCES

[1] J. Serra, E. Gómez, and P. Herrera, "Audio cover song identification and similarity: background, approaches, evaluation, and beyond," in *Advances in Music Information Retrieval*. Springer, 2010, pp. 307–332.

[2] F. Yesiler, C. Tralie, A. A. Correya, D. F. Silva, P. Tovstogan, E. Gómez Gutiérrez, and X. Serra, "Da-tacos: A dataset for cover song identification and understanding," in *Int. Soc. for Music Information Retrieval (ISMIR)*, 2019.

[3] J. Serra, X. Serra, and R. G. Andrzejak, "Cross recurrence quantification for cover song identification," *New Journal of Physics*, vol. 11, no. 9, 2009.

[4] A. A. Correya, R. Hennequin, and M. Arcos, "Large-scale cover song detection in digital music libraries using metadata, lyrics and audio features," *arXiv:1808.10351*, 2018.

[5] F. Yesiler, J. Serra, and E. Gómez, "Accurate and scalable version identification using musically-motivated embeddings," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 21–25.

[6] F. Yesiler, J. Serra, and E. Gomez, "Less is more: Faster and better music version identification with embedding distillation," in *Int. Soc. for Music Information Retrieval (ISMIR)*, 2020.

[7] C.-C. Wang and J.-S. R. Jang, "Improving query-by-singing/humming by combining melody and lyric information," *IEEE/ACM Trans. on Audio, Speech, and Language Processing (TASLP)*, vol. 23, no. 4, pp. 798–806, 2015.

[8] D. P. Ellis and G. E. Poliner, "Identifying cover songs' with chroma features and dynamic programming beat tracking," in *IEEE Int. Conf. on Acoustics, Speech and Signal (ICASSP)*, vol. 4, 2007.

[9] J. Serra, E. Gómez, and P. Herrera, "Transposing chroma representations to a common key," in *IEEE CS Conf. on The Use of Symbols to Represent Music and Multimedia Objects*, 2008, pp. 45–48.

[10] T. Bertin-Mahieux and D. P. Ellis, "Large-scale cover song recognition using hashed chroma landmarks," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2011, pp. 117–120.

[11] J. Lee, S. Chang, S. K. Choe, and K. Lee, "Cover song identification using song-to-song cross-similarity matrix with convolutional neural network," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 396–400.

[12] J. Serra, E. Gómez, P. Herrera, and X. Serra, "Chroma binary similarity and local alignment applied to cover song identification," *IEEE Trans. on Audio, Speech, and Language Processing (TASLP)*, vol. 16, no. 6, pp. 1138–1151, 2008.

[13] D. P. Ellis and B.-M. Thierry, "Large-scale cover song recognition using the 2d fourier transform magnitude," in *Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2012.

[14] J. B. Smith, M. Hamasaki, and M. Goto, "Classifying derivative works with search, text, audio and video features," in *IEEE Int. Conf. on Multimedia and Expo (ICME)*, 2017, pp. 1422–1427.

[15] T. J. Tsai, T. Prätzlich, and M. Müller, "Known artist live song id: A hashprint approach," in *Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2016, pp. 427–433.

[16] C. Gupta, E. Yılmaz, and H. Li, "Automatic lyrics transcription in polyphonic music: Does background music help?" in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[17] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "Creating dali, a large dataset of synchronized audio, lyrics, and notes," *Trans. of the Int. Soc. for Music Information Retrieval (TISMIR)*, vol. 3, no. 1, 2020.

[18] K. Heafield, "KenLM: Faster and smaller language model queries," in *Workshop on Statistical Machine Translation (SMT)*, 2011.

[19] D. Stoller, S. Durand, and S. Ewert, "End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 181–185.

[20] X. Du, Z. Yu, B. Zhu, X. Chen, and Z. Ma, "Bytecover: Cover song identification via multi-loss training," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[21] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, "On combining classifiers," *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 3, pp. 226–239, 1998.

[22] N. Chen, W. Li, and H. Xiao, "Fusing similarity functions for cover song identification," *Multimedia Tools and Applications*, vol. 77, no. 2, pp. 2629–2652, 2018.

[23] C. J. Tralie, "Early mfcc and hpcp fusion for robust cover song identification," in *Int. Soc. for Music Information Retrieval (ISMIR)*, 2017.

[24] J. Osmalsky, J.-J. Embrechts, P. Foster, and S. Dixon, "Combining features for cover song identification," in *Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2015.

[25] M. Khadkevich and M. Omologo, "Large-scale cover song identification using chord profiles," in *Int. Soc. for Music Information Retrieval (ISMIR)*, vol. 13, 2013, pp. 233–238.

[26] P. Grosche and M. Müller, "Toward characteristic audio shingles for efficient cross-version music retrieval," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 473–476.

[27] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 4, pp. 824–836, 2018.

[28] L. Boytsov and B. Naidan, "Engineering efficient and effective non-metric space library," in *Similarity Search and Applications (SISAP)*, 2013.

[29] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv:1607.01759*, 2016.

[30] A. Vaglio, R. Hennequin, M. Moussallam, G. Richard, and F. D'alché-Buc, "Multilingual lyrics-to-audio alignment," in *Int. Soc. for Music Information Retrieval Conference (ISMIR)*, 2020.

ECOLE
DOCTORALE

Titre : Exploiter les paroles de chansons à partir de l’audio pour le MIR

Mots clés : Reconnaissance de la voix chantée, Détection de contenu explicite, Alignement des paroles et de l’audio, Identification du langage, Détection de reprise

Résumé : Les paroles de chansons fournissent un grand nombre d’informations sur la musique car elles contiennent une grande partie de la sémantique des chansons. Ces informations pourraient aider les utilisateurs à naviguer facilement dans une large collection de chansons et permettre de leur offrir des recommandations personnalisées. Cependant, ces informations ne sont souvent pas disponibles sous leur forme textuelle. Les systèmes de reconnaissance de la voix chantée pourraient être utilisés pour obtenir des transcriptions directement à partir de la source audio.
Ces approches sont usuellement adaptées de celles de la reconnaissance vocale. La transcription de la parole est un domaine vieux de plusieurs décennies qui a récemment connu des avancées significatives en raison des derniers développements des techniques d’apprentissage automatique. Cependant, appliqués au chant, ces algorithmes donnent des résultats peu satisfaisants et le processus de transcription des paroles reste difficile avec des complications particulières.
Dans cette thèse, nous étudions plusieurs problèmes de ’Music Information Retrieval’ scientifiquement et industriellement complexes en utilisant des informations sur les paroles générées directement à partir de l’audio. L’accent est mis sur la nécessité de rendre les approches aussi pertinentes que possible dans le monde réel. Cela implique par exemple de les tester sur des ensembles de données vastes et diversifiés et d’étudier leur extensibilité. À cette fin, nous utilisons un large ensemble de données publiques possédant des annotations vocales et adaptons avec succès plusieurs des algorithmes de reconnaissance de paroles les plus performants.
Nous présentons notamment, pour la première fois, un système qui détecte le contenu explicite directement à partir de l’audio. Les premières recherches sur la création d’un système d’alignement paroles-audio multilingue sont également décrites. L’étude de la tâche alignement paroles-audio est complétée de deux expériences quantifiant la perception de la synchronisation de l’audio et des paroles. Une nouvelle approche phonotactique pour l’identification de la langue est également présentée. Enfin, nous proposons le premier algorithme de détection de versions employant explicitement les informations sur les paroles extraites de l’audio.

Title : Leveraging lyrics from audio for MIR

Keywords : Singing voice recognition, Explicit content detection, Lyrics-to-audio alignment, Language identification, Cover song detection

Abstract : Lyrics provide a lot of information about music since they encapsulate a lot of the semantics of songs. Such information could help users navigate easily through a large collection of songs and to recommend new music to them. However, this information is often unavailable in its textual form. To get around this problem, singing voice recognition systems could be used to obtain transcripts directly from the audio.
These approaches are generally adapted from the speech recognition ones. Speech transcription is a decades-old domain that has lately seen significant advancements due to developments in machine learning techniques. When applied to the singing voice, however, these algorithms provide poor results. For a number of reasons, the process of lyrics transcription remains difficult.
In this thesis, we investigate several scientifically and industrially difficult ’Music Information Retrieval’ problems by utilizing lyrics information generated straight from audio. The emphasis is on making approaches as relevant in real-world settings as possible. This entails testing them on vast and diverse datasets and investigating their scalability. To do so, a huge publicly available annotated lyrics dataset is used, and several state-of-the-art lyrics recognition algorithms are successfully adapted.
We notably present, for the first time, a system that detects explicit content directly from audio. The first research on the creation of a multilingual lyrics-to-audio system are as well described. The lyrics-to-audio alignment task is further studied in two experiments quantifying the perception of audio and lyrics synchronization. A novel phonotactic method for language identification is also presented. Finally, we provide the first cover song detection algorithm that makes explicit use of lyrics information extracted from audio.

Institut Polytechnique de Paris
91120 Palaiseau, France