Approches et méthodologies pour la réponse automatique à des questions adaptées à un cadre interactif en domaine ouvert

Olivier Galibert

NOTES et DOCUMENTS LIMSI N^{O} : 2009 - 05
Juillet 2008

APPROCHES ET MÉTHODOLOGIES POUR LA RÉPONSE AUTOMATIQUE À DES QUESTIONS ADAPTÉES À UN CADRE INTERACTIF EN DOMAINE OUVERT

Olivier GALIBERT

Thèse soutenue le 4 juin 2009 devant le jury composé de :

Rapporteurs	Patrice BELLOT
	Guy LAPALME
Directrice	Martine ADDA–DECKER
Président du Jury	Pierre ZWEIGENBAUM
Examinateurs	Jean-Luc GAUVAIN
	Édouard GEOFFROIS
	Brigitte GRAU

Notes et Documents LIMSI N^{O} : 2009 - 05 **Juin 2009**

Auteurs (Authors) : Olivier Galibert

Titre : Approches et méthodologies pour la réponse automatique à des questions adaptées à un cadre interactif en domaine ouvert.

Title : Approaches and methodologies for automatic Question-Answering in an open-domain, interactive setup.

Nombre de pages (Number of pages) : 184

Résumé : *L'objectif de cette thèse a été de proposer de nouvelles approches robustes pour le problème de la réponse à des questions dans un cadre ouvert interactif.*

Notre première contribution a été la conception et mise en oeuvre d'un moteur générique d'analyse de la langue. C'est un moteur sans a-priori sur les types d'analyses qui peuvent être effectués, dans la limite de ce qu'il peut représenter et qui met en avant la structuration de l'analyse.

Notre seconde contribution a consisté en la conception et la mise en oeuvre d'un Système Question-Réponse dont les principales forces sont flexibilité de l'entrée, la robustesse et le contrôle des performances. Cela se fait via une intégration de bout en bout du résultat de l'analyse, qui permet de ne manipuler que les structures résultantes de l'analyse sans devoir redescendre aux mots. Il propose aussi, et c'est une des grandes originalités de ce travail, une abstraction de la requête, source de sa flexibilité, et facilitant sa compréhension et sa maintenance.

Nous avons participé à des campagnes d'évaluation internationales, où nos systèmes ont obtenu d'excellents résultats. En particulier, ils ont montré une bonne résistance aux erreurs induites par un système de transcription automatique de la parole.

Il est cependant important de noter que notre but a été atteint. Le système global de Question-Réponse a les capacités nécessaires pour s'intégrer dans un système interactif. Il est utilisé dans le cadre du projet Ritel et a permis des premières expériences dont le but était d'étudier le comportement des humains face à un tel système et l'interaction homme-machine en domaine ouvert en général.

Mots clés : Question-Réponse, Analyse de la langue, Interaction, Domaine ouvert

Abstract : *The objective of this work is to introduce new robust approaches to handle the problem of Question Answering in an open-domain, interactive setup.*

Our first contribution is the design and implementation of a generic rules-based engine for language analysis. That engine is open to any kind of analysis, within the limits of its internal representation, and leverages an heavy structuring of the analysis.

Our second contribution is the design and implementation of a Question-Answering system which main strengths are the flexibility of the input, the robustness and the explicit performance control. These characteristics have been reached through an end-to-end integration of the language analysis results, allowing to manipulate structures build by that analysis only, without having to go back to the individual words. Another advance, and it is one of its main originalities, is an abstraction of the request, enabling its flexibility and making diagnostic and maintenance easier.

We participated to a number of international evaluation campaigns where our system achieved excellent results. In particular they have shown a good robustness to automatic speech recognition induced errors.

It is important to note that our aim has been reached. The Question-Answering system has the necessary capabilities to be integrated in an interactive system. It is used in the Ritel project and allowed some preliminary experiences aiming at studying the human behavior in front of such a system, and human-machine interaction in general.

Keywords : Question-Answering, Language analysis, Interactivity, Open domain

Table des matières

Introduction

Interaction orale homme-machine

L'interaction en langue entre un humain et une machine est un très vieux sujet. Il faut se souvenir que le grand classique du domaine, le programme Eliza de Joseph Weizenbaum [Weizenbaum 1966], date de 1966. Il faut noter toutefois que Eliza n'était pas vraiment un système de dialogue, ne mémorisant rien de l'interaction, ni un système de recherche d'informations. Il s'agissait d'un pur jeu d'interaction. Le premier système de dialogue homme-machine pour de la recherche d'informations est LUNAR [Woods 1973], qui répondait à des questions concernant des échantillons du sol lunaire. L'interaction *orale* homme-machine toutefois est beaucoup plus récente car elle dépend des progrès de la reconnaissance automatique de la parole. Celle-ci a connu un essor déterminant au début des années 90 grâce notamment à des programmes d'évaluation massifs [Pallett, et al. 1994]. Le dialogue oral a suivi le mouvement et une sous partie du domaine en particulier a connu un très fort développement : les systèmes permettant à un humain d'effectuer une tâche dans un domaine précis, comme trouver des renseignements touristiques ou réserver une place de train. Des projets américains comme Communicator [Walker, et al. 2002] (horaires d'avions) ou européens comme Arise [Lamel, et al. 2000] ont contribué à atteindre un niveau de performances tout à fait acceptables. La figure 1 donne un extrait d'un tel dialogue, et on peut constater qu'il est raisonnablement naturel et efficace.

S : Quels renseignements désirez-vous ?
U : euh je pars de Paris pour aller à Caen euh je pars lundi
S : Il y a plus de 10 trains de Paris à Caen le lundi 28/04. Le premier train part à 00h25 et le dernier à 23h14. À quel moment de la journée voulez-vous partir ?
U : vers midi
S : Vous avez un train direct de Paris à Caen à 12h25 arrivant à 14h09 le lundi 28/04. Quelle autre information désirez-vous ?
U : je voudrais connaître le prix du train précédent

FIG. 1 – Extrait de dialogue du système Arise, tiré de [Rosset 2000]

La recherche continue bien entendu sur ce sujet, en particulier pour essayer de diminuer les coûts toujours élevés de conception de tels systèmes. Cependant de nombreux chercheurs veulent essayer d'aller plus loin. Une option est de changer la tâche. Par exemple des études très intéressantes ont lieu dans le domaine de l'éducation [Cole, et al. 1998]. Mais il existe aussi l'option de rester dans le domaine de la recherche d'informations, mais d'ouvrir le domaine. Le but devient alors d'interagir et de répondre de façon pertinente sur un peu n'importe quel sujet non décidé à l'avance. Un tel système doit donc trouver des réponses précises à des questions aussi précises que possible. Ce domaine de recherche existe déjà : il s'agit de l'étude des *Système de Réponse à des Questions* ou, pour faire court *Système Question-Réponse* par analogie à leur nom anglais, *Question-Answering System* ou *QA system*.

Recherche d'Informations et Question-Réponse

Au début des années 50, la recherche d'informations était déjà un domaine de recherche actif. L'origine du terme *Information Retrieval* peut être retracé à Calvin Mooers en 1950, un pionnier du domaine, qui travaillait sur la sélection de cartes perforées d'après des mots-clés [Mooers 1948] par des méthodes hybrides informatiques et mécaniques. Les premiers travaux concernaient ainsi essentiellement l'indexation, et spécifiquement la sélection de documents dans des bases trop grandes pour être traitées à la main (données du recensement américain, livres de bibliothèque, bases d'articles). L'aspect applicatif et expérimental a toujours été central dans la discipline. On peut donner pour exemple le projet Cranfield [Cleverdon 1967] qui cherchait à évaluer les systèmes de l'époque. Un corpus de 18 000 documents et 1 200 requêtes étaient fournis et la performance était mesurée sous forme de précision et rappel. Plus récemment la série de campagnes d'évaluation TREC (Text REtrieval Conference) organisée par le NIST tous les ans depuis 1992 [Press 2005] a permis de raffiner à travers les ans aussi bien les systèmes que les méthodes, approches et l'évaluation elle-même. D'autres campagnes ont suivi le pas comme CLEF [Magnini, et al. 2003], un pendant européen qui met en avant les aspects multi- et trans-lingues.

Un tel système de sélection de documents est très utile, comme le démontre tous les jours un moteur de recherche sur le Web comme Google ou dans des transcriptions d'émissions d'informations comme Audiosurf (figure 2, www.audiosurf.org). Seulement, comme on peut le voir dans cet exemple, trouver la réponse à une question précise telle que "*Qui a remporté le caucus en Iowa chez les démocrates en 2008 ?*" n'est pas immédiat malgré l'aide de l'ordinateur. L'utilisateur a besoin de parcourir les différents extraits et même souvent les documents eux-mêmes pour trouver la réponse à sa question. Ainsi une extension naturelle des systèmes de recherche d'informations est de tenter de retourner des réponses précises aux questions de l'utilisateur. Après des expérimentations avec des encyclopédies comme base de connaissances (MURAX [Kupiec 1993]) sont apparues dès 1999 des évaluations dites *Question-Réponse* (*Question Answering*). Cette tâche, introduite dans TREC-8 [Voorhees & Tice 1999] demandait à un système confronté à la même question de répondre "*Barak Obama*". Cette tâche a été reprise, étendue et ses formes modernes seront décrites dans la partie III.

Cette notion de réponse courte à des questions factuelles est particulièrement intéressante dans un cadre interactif oral. Elle permet en effet au système de dialogue de produire des réponses efficaces aux questions de l'utilisateur et, dans le cas de documents audio, de rejouer les bouts de documents associés.

Cadre de la thèse

Le projet Ritel [Ritel 2007 ; Rosset, et al. 2006] a vu le jour au LIMSI en 2004. Son objectif est de construire une plateforme de dialogue permettant d'assister un utilisateur dans diverses tâches dont, en premier lieu, la recherche d'informations en domaine ouvert. Le probleme de la recherche d'infor-

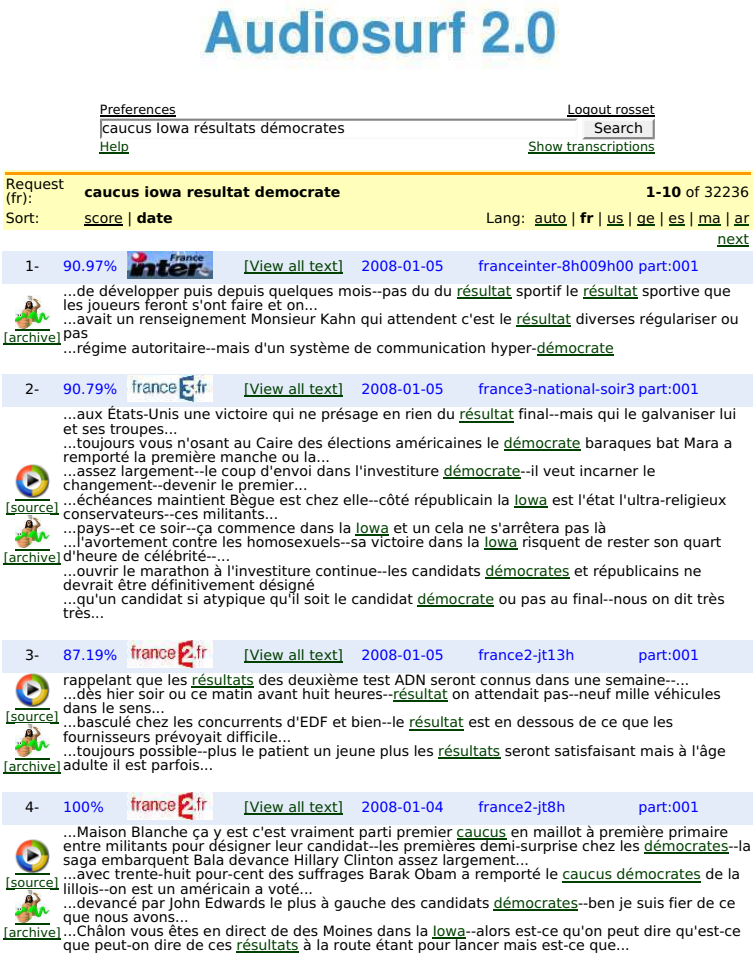


FIG. 2 – Système de recherche d'informations

mations interactive en domaine contraint est étudié depuis longtemps. Passer en domaine ouvert pose de nombreuses difficultés, et une des questions que nous avons eu à nous poser est *quels types d'information est-on capable de rechercher en dehors d'une tâche précise*. Cette réflexion nous a mené dans le domaine des *Systèmes Question-Réponse*. Intégrer un tel système dans un environnement de dialogue oral impose des contraintes inhabituelles pour ces systèmes. La première est un besoin de flexibilité sur l'entrée du système, c'est à dire les questions. En effet l'entrée orale implique plusieurs difficultés : non seulement la parole a une syntaxe très différente de l'écrit, mais la reconnaissance automatique de la parole ajoute des erreurs. Plus important encore, l'aspect interactif implique une

gestion d'historique du dialogue et donc de pouvoir ajouter des éléments potentiellement pertinents à la question. La deuxième contrainte est la gestion des temps de réponse. En effet dans le cadre d'une interaction, orale ou non, il n'est pas concevable de laisser l'utilisateur attendre longtemps pour sa réponse. Il faut donc pouvoir contrôler la quantité de travail maximale que le système est autorisé à effectuer. Ces contraintes nouvelles nécessitent des approches spécifiques adaptées.

En dehors d'une structure générale similaire il n'y a pas vraiment d'approche standard pour la conception de systèmes de Question-Réponse. Ils peuvent aller du tout statistique sans connaissance explicite de la langue au très linguistique avec analyse profonde basée sur de grandes bases de connaissances et incluant du raisonnement logique sur les concepts extraits. Notre expérience dans le domaine du dialogue nous a poussé vers une organisation un peu intermédiaire : une *analyse de la langue*, que l'on pourrait qualifier de *compréhension*, est appliquée aux documents et aux questions et leur impose une *structure*. L'ensemble des algorithmes de recherche travaille alors uniquement sur le résultat de cette structuration.

Notre travail se divise donc en deux parties, une analyse de la langue et un système de recherche de réponses. Construire une analyse de la langue, qui unifie les besoins du dialogue et de Question-Réponse, reste un problème ouvert en pleine évolution. De plus il ne relève pas vraiment de notre compétence mais plutôt de celle de linguistes. Notre but est donc devenu de fournir à des linguistes le meilleur outil possible pour leur permettre de construire au mieux une analyse performante aussi bien pour le dialogue que pour la réponse aux questions.

Nous avons donc dû étudier le problème des *moteurs d'analyse*. Un tel moteur doit permettre d'expérimenter tant sur les méthodes d'analyse que sur les représentations et annotations obtenues, tout en offrant de bonnes performances et une ergonomie de qualité. Ces besoins nous ont conduit à nous intéresser en particulier aux moteurs permettant d'écrire des *systèmes à base de règles*. En effet les systèmes statistiques reposent sur de grands corpus annotés en fonction des besoins de l'analyse. Devoir réannoter ces corpus pour tester des variations rend l'expérimentation beaucoup plus difficile. Avec un système basé sur des règles rien n'est figé. Il est toujours possible de modifier localement le schéma d'annotation choisi en modifiant un petit nombre de règles concernées. Il est de même souvent plus facile d'agir sur des points précis, des tournures de phrases par exemple, où le système fait des erreurs, en travaillant là encore sur les règles spécifiquement concernées. Dans un cadre statistique une action équivalente nécessite de construire un corpus supplémentaire où la tournure pertinente est souvent représentée et l'annoter correctement.

Problématique

L'objectif de notre travail est de proposer de nouvelles approches robustes pour le problème de la réponse à des questions dans un cadre ouvert interactif. Nous décomposons ce problème général en deux sous parties. La première traite de la définition d'un moteur d'analyse de la langue permettant à un expert d'expérimenter librement et efficacement autour du problème de la compréhension pour la

recherche d'informations. La seconde s'attaque au problème spécifique de la réponse aux questions en s'appuyant sur un type d'analyse donné tout en respectant deux contraintes fortes liées au cadre interactif : les questions peuvent être déstructurées car construites à partir d'éléments provenant de plusieurs échanges entre l'utilisateur et la machine, et les temps de réponse doivent être contrôlables a priori.

Principales contributions

Notre première contribution est la conception et la mise en œuvre d'un **moteur générique d'analyse de la langue** permettant à un linguiste de construire un système pour extraire toutes sortes d'informations qu'il peut juger utiles de documents ou de questions. C'est un moteur **sans a priori** sur les types d'analyses qui peuvent être effectuées, dans la limite de ce qu'il peut représenter. Il tente d'obtenir un bon **équilibre entre l'expressivité et l'ergonomie**. Pour cela il **met en avant la structuration de l'analyse**, tant au niveau de l'organisation générale avec un support fort de l'analyse incrémentale, mais aussi à l'intérieur des règles elles-mêmes, et offre de **très bonnes performances** en terme de vitesse, permettant un travail expérimental plus efficace.

Notre seconde contribution consiste en la conception et la mise en œuvre d'un **Système Question-Réponse** utilisant un système d'analyse performant produit avec notre moteur afin de chercher des réponses précises à des questions précises et qui essaie de mettre en valeur la **flexibilité de l'entrée**, la **robustesse** et le **contrôle des performances**. Cela se fait via une **intégration de bout en bout du résultat de l'analyse**, qui permet de ne manipuler que les structures résultantes de l'analyse sans devoir redescendre aux mots. Il propose aussi, et c'est une des grandes originalités de ce travail, une **abstraction de la requête**, source de sa flexibilité, et facilitant sa compréhension et sa maintenance.

Organisation du document

La **première partie** de ce document traite du moteur d'analyse. Nous présentons dans l'**introduction** la problématique générale de l'analyse de la langue en domaine ouvert. L'analyse nécessaire pour de la recherche d'informations dans un cadre ouvert et interactif est un champ de recherche en pleine évolution. Cela entraîne un besoin d'expérimentation à tous les niveaux et en particulier dans la définition même des annotations utiles. En conséquence un support fort des approches à base de règles est indispensable. Le **chapitre 1** décrit plusieurs moteurs existants proposant un support pour l'écriture de règles et analyse leurs points forts et limitations par rapport à nos objectifs. Le **chapitre 2** est alors consacré au moteur que nous proposons. Après une présentation de son architecture générale, nous détaillons les points saillants de sa conception tels que vus par un expert construisant son système d'analyse. Cela inclut en particulier sa structuration pour l'analyse incrémentale, sa représentation interne des informations et les points importants de son langage d'écriture de règles. Tout cela est suivi,

chapitre 3, par la présentation des algorithmes importants utilisés par ce moteur qui contribuent fortement à sa rapidité. Enfin cette première partie se termine par une évaluation du moteur, **chapitre 4**, à travers une présentation de cas d'utilisations et des mesures quantitatives de performance sur l'analyseur le plus complexe dont nous disposons.

La **seconde partie** se tourne alors vers le système Question-Réponse. Après une **introduction** posant le problème, le **chapitre 5** présente la structure générale commune à l'essentiel des systèmes QR existants et décrit plus en détail trois systèmes représentatifs des différentes approches décrites dans la littérature : un sans connaissances explicites sur la langue et se fondant en conséquence sur des approches purement statistiques, un très linguistique s'appuyant sur des taxonomies détaillées et du raisonnement logique, et enfin un que nous considérons intermédiaire. Nous appuyant sur une analyse de la langue construite utilisant le moteur présenté dans la première partie (**chapitre 6**), nous introduisons au **chapitre 7** une première approche pour Question-Réponse construite sur des listes de requêtes écrites à la main. La généralisation de cette approche constitue la base d'une approche globale plus avancée que nous présentons ensuite en détail **chapitre 8**. Cela inclut la représentation abstraite de la recherche, la gestion de la variabilité de la langue via des transformations, la sélection de documents et de passages et enfin l'extraction et l'évaluation des réponses. Si nous nous sommes particulièrement intéressé aux questions dites factuelles nous présentons **chapitre 9** des méthodes préliminaires pour traiter d'autres types de questions telles que les demandes de définitions ou d'explications, ces méthodes réutilisant tout ou partie des approches proposées.

La **troisième partie** est dédiée à l'évaluation du système Question-Réponse proposé. L'**introduction** présente le problème général de l'évaluation d'un système répondant à des questions en langue naturelle. Elle est suivie du **chapitre 10** qui détaille les caractéristiques principales des campagnes d'évaluation majeures de ces dix dernières années. C'est dans le cadre de telles campagnes que nous avons évalué notre travail. Le **chapitre 11** présente les résultats obtenus aux campagnes auxquelles nous avons participé. Ces résultats donnent une information globale sur la qualité du système de bout en bout, mais une telle évaluation n'est pas tout, et nous nous sommes donc attaché à donner d'autres points de vue. Le **chapitre 12** tout d'abord étudie l'impact de la taille des données de développement sur la qualité finale du système. Le **chapitre 13** détaille les résultats à chaque étape du système global et essaie de mettre en avant leurs interactions. Enfin il ne faut pas oublier que le contrôle des temps de réponse est un de nos besoins de base, et deux paramètres numériques sont prévus pour cela. Le **chapitre 14** s'attache à étudier le système du point de vue vitesse et qualité du résultat en fonction des valeurs données à ces paramètres.

Enfin un tel document ne saurait être complet sans une discussion finale de l'ensemble et une présentation de quelques perspectives envisagées, qui font l'objet de la **dernière partie**.

Première partie

Un moteur d’analyse de la langue

Introduction

Dans la conception d'un système de dialogue oral homme-machine, un des problèmes se posant est celui de la *compréhension*. Il faut pouvoir analyser la requête de l'utilisateur, qui a été préalablement automatiquement transcrite par le système de reconnaissance, et produire une représentation en lien avec le sens de la demande. L'analyse doit être robuste face aux erreurs de reconnaissance et face aux particularités de l'oral (hésitation, répétition, auto-correction, reprise, etc.). Un tel système doit être en mesure de détecter, d'extraire et de typer l'*information pertinente* dans les énoncés. Il s'agit donc d'être capable de représenter le sens d'un énoncé, d'une phrase. Cette représentation peut utiliser les éléments qui semblent importants pour l'application visée.

Dans le cadre des systèmes de dialogue pour de la recherche d'informations, l'essentiel des travaux [Bonneau-Maynard, et al. 2005 ; Lamel et al. 2000 ; Walker et al. 2002 ; Boye, et al. 2006] a porté sur des domaines limités, bien ciblés. Le type d'information que l'on peut capturer, représenter et repérer est lié à la tâche et à la présence d'éléments dans une base de données. Souvent une représentation est proposée qui s'appuie sur trois types d'informations : le domaine tout d'abord, la base de données dans laquelles les informations seront recherchées ensuite, et enfin un ensemble d'actes communicatifs correspondant à la tâche. Avec une telle définition précise des éléments qui sont utiles à reconnaître, la démarche classique consiste à élaborer des conventions d'annotation, annoter des corpus, et ensuite à construire des systèmes capables d'extraire automatiquement ces annotations.

Nous pouvons donner en exemple le cas de MEDIA [Bonneau-Maynard et al. 2005 ; Devillers, et al. 2004], un projet français consacré à l'évaluation de systèmes de compréhension dans un cadre de dialogue oral en domaine fermé. Les participants à ce projet ont dû définir une représentation couvrant le type d'informations qu'ils voulaient pouvoir extraire et donc évaluer. Le domaine couvre en l'occurence les informations touristiques et plus précisément la réservation de chambres d'hôtel. La base de données associée contient les noms d'hôtels, de villes, les services, les tarifs, etc. Enfin les actes communicatifs couvrent requête, acceptation, rejet, ouverture et fermeture. La représentation construite en partant de ces informations définit les segments sémantiques comme des 5-uplets contenant :
– un identifiant numérique
– la réalisation linguistique du segment (séquence de mots)
– le mode (affirmatif '+', négatif '-', interrogatif ' ?' ou optionnel '$\sim$')

identifiant	séquence de mots	mode	nom de l'attribut	valeur
0	euh	+	null	
1	oui	+	reponse	oui
2	l'	+	lienRef-coRef	singulier
3	hôtel	+	ObjetBD	hotel
4	dont	+	null	
5	le prix	+	object	paiement-montant-chambre
6	ne depasse pas	+	comparatif-paiement	inferieur
7	cent dix	+	paiement-montant-entier-chambre	110
8	euros	+	paiement-monnaie	euro

FIG. 3 – Représentation sémantique dans le cadre de MEDIA sous forme de 5-uplets de la phrase *euh oui l'hôtel dont le prix ne dépasse pas cent dix euros.* tiré de [Bonneau-Maynard et al. 2008]

– le nom de l'attribut représentant le sens de la séquence de mots
– la valeur de l'attribut

Un exemple d'annotation de phrase suivant ce protocole d'annotation est donné figure 3.

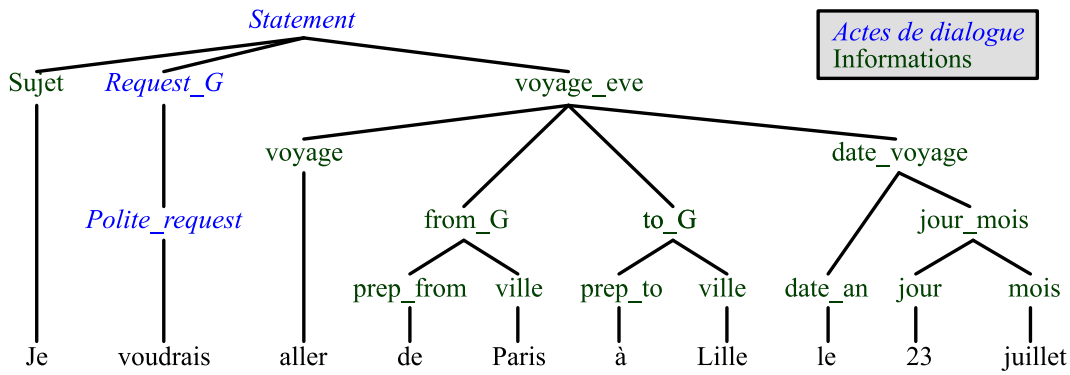


FIG. 4 – Représentation arborescente de l'information contenue dans la phrase *je voudrais aller de Paris à Lille le 23 juillet*

Les 5-uplets de MEDIA sont loin d'être la seule représentation possible pour ce genre d'informations. Par exemple, la figure 4 montre comment une structure arborescente peut être utilisée pour analyser *je voudrais aller de Paris à Lille le 23 juillet*. Le même exemple peut plus simplement être traité par une analyse par mots-clés pour obtenir une représentation sous forme de schéma :

{
ville-from : Paris
ville-to : Lille
date : 23 juillet
}

Mais dès que nous entrons dans un cadre plus ouvert, le problème devient beaucoup plus difficile. En effet, en l'absence d'un domaine précis, quel sera le vocabulaire à traiter ? Quels seront les concepts utiles, produits par le locuteur ou présents dans les documents, à chercher par l'analyseur et le système de recherche d'informations ? Nous dépassons là le cadre habituel de la compréhension pour l'interaction et devons retourner aux sources, l'analyse de la langue en tant qu'objet.

Nous cherchons donc en premier lieu à extraire les *informations utiles* des textes ou de la parole transcrite que nous voulons traiter. C'est une notion volontairement floue, car elle met souvent en jeu des concepts difficiles à définir formellement (sens, intention, besoin, importance...). Bien souvent il s'agit d'un compromis entre ce que l'on sait extraire et ce que l'on sait exploiter dans la suite de l'application et évolue régulièrement en fonction des résultats obtenus dans le système global.

Cette information prend en général la forme de structures (regroupements, liens), le plus souvent incluant un nommage (on parle de typage). Cette structuration est nommée une *annotation*. Contrairement aux mots ou à un signal de parole, une annotation n'est pas directement observable mais est une construction de l'esprit en vue de mettre en valeur les points qui nous intéressent. Il n'existe donc pas de vérité absolue dans le domaine de l'annotation, et chaque schéma d'annotation est défini en fonction de ce que l'on comprend de la langue à un moment donné et des besoins, applicatifs ou non, de l'utilisateur du résultat. Analyser de la langue revient donc à poser, automatiquement ou non, des annotations en fonction d'un schéma d'annotation défini par des humains. Un certain nombre de ces schémas d'annotation venant de la linguistique sont standardisés dans leurs grandes lignes et ont des utilités reconnues, même si les détails spécifiques (listes exactes des types ou structures employées par exemple) varient.

Plusieurs types d'analyse peuvent être utiles. L'analyse en Parties du Discours (Part-Of-Speech, POS) [Adda, et al. 1999] indique pour chaque mot sa catégorie grammaticale (verbe, substantif, déterminant...). La détection d'Entités Nommées [Nadeau & Sekine 2007] repère les mots et expressions désignant des entités concrètes (personnes, organisations, lieux...) du monde réel et les type. Le chunking [Abney 1991] découpe les phrases en groupes syntaxiques élémentaires (groupes nominaux, groupes verbaux...). L'analyse syntaxique [Bourigault 2007] détecte et type les relations entre ces groupes d'un point de vue grammatical. La détection des rôles sémantiques (Semantic Role Labeling, SRL) [Carreras & Màrquez 2005] réinterprète ces groupes et relations en termes plus sémantiques, en particulier ceux tournant autour du verbe. Chacune de ces analyses apporte une partie des informations utiles pour l'application. La figure 5 montre un exemple regroupant plusieurs de ces types d'annotations.

Les approches d'analyse de la langue sont traditionnellement divisées de par leur fonctionnement

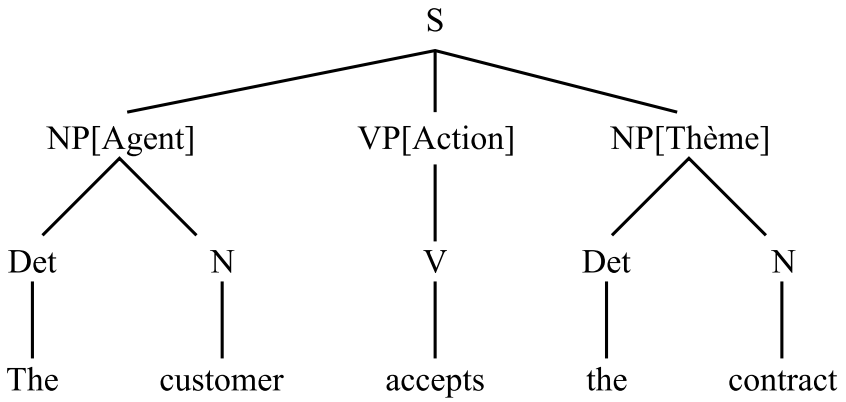


FIG. 5 – Exemple d'annotation sur la phrase *The customer accepts the contract* incluant Parties du Discours (Det=Déterminant, N=Nom, V=Verbe), Arbre syntaxique (NP=Groupe nominal, VP=Groupe verbal, S=Phrase) et Rôles sémantiques (Agent, Action et Thème). Tiré de [De Mori et al. 2008].

sous-jacent. Plus précisément, la séparation est faite entre les approches statistiques d'un côté et les approches dites symboliques, essentiellement par règles, de l'autre. Chacunes ont leurs forces et leurs faiblesses.

Les approches statistiques suivent en général la même structure. Un modèle statistique est défini permettant de calculer, pour chaque annotation possible, un score qui a souvent la forme d'une probabilité. Un algorithme dit de recherche essaie ensuite de trouver, dans l'ensemble de toutes les annotations possibles, celle qui obtient le meilleur score. Les systèmes s'appuyant sur des approches statistiques ont tendance à ne produire, d'un point de vue linguistique, qu'un seul niveau à la fois. Chacun s'appuie d'ailleurs souvent sur les résultats d'un ou plusieurs des niveaux précédents. Cela a deux conséquences : la première est que les erreurs ont tendance à s'accumuler et une mauvaise décision tôt, par exemple une désambigüisation ratée pour *la* entre substantif et déterminant, peut avoir des conséquences importantes sur tous les niveaux suivants. La seconde est que les décisions peuvent avoir lieu trop tôt. Certaines parties du discours ne peuvent être choisies correctement sans des informations de niveaux supérieurs, syntaxiques ou même sémantiques. La situation peut être partiellement corrigée en annotant plusieurs niveaux simultanément avec des modèles interdépendants ou en transmettant plusieurs hypothèses plutôt qu'une seule entre chaque étape. Cependant ces méthodes posent des difficultés au niveau combinatoire et peuvent donner lieu à des temps de calcul excessifs.

Les approches statistiques peuvent elles-mêmes être subdivisées en trois sous-catégories. Les traditionnelles, dites supervisées, s'appuient sur un *corpus d'entraînement* contenant une quantité importante de textes annotés par des humains. Le modèle contient alors un certain nombre, souvent grand, de variables numériques libres dont les valeurs sont choisies pour maximiser un score, souvent une probabilité, calculé sur ce corpus. Le modèle ainsi optimisé sert alors à produire les nouvelles an-

notations. Le coût de la création de ce corpus est évident et est souvent un obstacle à l'usage de ces méthodes. Plus subtil est la difficulté inhérente à la création du schéma d'annotation. En effet, même sur les niveaux les plus bas, il n'y a pas toujours consensus sur les choix d'annotation (les étiquettes de Parties du Discours doivent-elles être Verbe, Nom ou alors Verbe à la 3ème personne du singulier du présent de l'indicatif et Nom féminin pluriel), et plus les niveaux sont élevés plus les possibilités sont nombreuses et deviennent un problème de recherche à part entière. Par exemple une grande partie du travail dans les projets MEDIA (compréhension de la parole) [Bonneau-Maynard et al. 2005] ou EASY (analyse syntaxique) [Paroubek, et al. 2006] a été la définition des annotations. De plus le coût d'une réannotation manuelle du corpus d'entraînement rend difficile l'expérimentation avec les annotations, en particulier dans le cadre de l'application complète. Il est désagréable et coûteux de s'apercevoir qu'avoir le genre et nombre dans les parties du discours aurait été utile après que quelques centaines de milliers de mots aient déjà été annotés. Et la portabilité inter-lingue ou inter-domaine est inexistante. Il faut à chaque changement de langue, domaine ou modalité (écrit vers oral par exemple) créer un nouveau corpus d'entraînement adapté. Cependant quand le besoin est bien défini, ou mieux encore des corpus appropriés sont disponibles (en anglais le Treebank [Marcus, et al. 1993] est très populaire pour les parties du discours et la syntaxe), les approches statistiques supervisées sont des plus utiles et efficaces.

Une seconde sous-catégorie des méthodes statistiques, en plein essor, est l'ensemble des méthodes semi-supervisées qui combinent une taxonomie linguistique telle que VerbNet pour les rôles sémantiques en anglais avec une grande quantité de textes non annotés pour tenter d'obtenir via des statistiques tirées des textes bruts puis dirigées par, ou corrélées avec, la taxonomie des modèles d'annotation intéressants. Il est ainsi possible d'obtenir avec VerbNet un système d'annotation en rôles sémantiques avec des performances intéressantes sans corpus annoté explicitement avec ces rôles [Swier & Stevenson 2004]. Ces approches nouvelles sont très prometteuses mais les taxonomies nécessaires représentent une quantité de travail phénoménale : VerbNet [Schuler 2005] représente plus de 5 ans de travail par des linguistes et évolue encore, et s'appuie sur d'autres taxonomies (WordNet [Press 1998], XTag [Prolo 2002], FrameNet [Baker, et al. 1998]) qui représentent des dizaines d'années de travail. Rien à voir avec un corpus d'apprentissage pour système supervisé dont le temps de création, une fois le schéma d'annotation défini, se compte en mois. Ce sont là encore des méthodes très intéressantes si les taxonomies existent et sont disponibles. Des efforts existent pour constituer de telles ressources pour de nombreuses langues incluant le français, mais la langue la mieux couverte à ce jour reste l'anglais.

La dernière sous-catégorie couvre les méthodes non-supervisées. À partir de rien d'autre qu'un ensemble de textes non annotés ces approches fournissent des pseudo-parties du discours [Schütze 1995] ou encore un *bracketing*, réécriture de la phrase sous une forme d'arbre qui est une des formes possibles de l'analyse syntaxique [Seginer 2007]. Elles s'appuient pour cela essentiellement sur des statistiques de co-occurrence orientée de telle ou telle façon (co-occurrences simples de mots, co-occurrences de contextes entre des mots différents, ...) suivant le type d'information recherchée. Ce sont des approches montantes des plus intéressantes, ne demandant que des textes faciles à obtenir en grande quantité et du temps CPU. Malheureusement une limitation intrinsèque de ces méthodes qui les rendent difficilement utilisables dans un cadre applicatif est celui du nommage. En effet un

système d'annotation en parties du discours non-supervisé ne va pas décider entre verbe et nom mais entre les classes numéro 1 et 6 qu'il aura créées lui-même. Difficile ensuite pour l'humain d'interpréter cela en des termes adaptés à sa compréhension. D'autant plus que les classes créées par le système auront des points communs avec celles qu'utiliserait un humain, mais elle ne seront pas strictement identiques. À moins de pouvoir faire tourner la totalité de l'application avec de telles classes abstraites générées automatiquement (et il n'existe à notre connaissance pas encore de telles méthodes pour les entités nommées, le chunking ou les rôles sémantiques) l'intégration avec des méthodes plus traditionnelles qui demandent et produisent des classes nommées conventionnelles est extrêmement difficile.

Les approches dites symboliques couvrent en pratique des méthodes très variées : règles, grammaires, logique... Les caractériser par le terme symbolique est un peu un abus de langage. Les systèmes statistiques, après tout, travaillent sur les mêmes symboles. Ces approches symboliques ont cependant des points communs : un expert (ou plusieurs) programme le système d'annotation dans un langage qui lui paraît adapté à ses besoins, et qui, comme les grammaires non-contextuelles, peut ne pas ressembler du tout à un langage de programmation. Les opérations de ce "programme" sont discrètes et transforment la représentation interne du langage. En comparaison, les méthodes statistiques, via la recherche de l'annotation donnant le meilleur score, déplacent toute décision le plus tard possible. Les approches symboliques nécessitent un gros travail initial d'écriture et de débogage des règles (ou autres). Ce travail est rendu d'autant plus compliqué qu'il est souvent difficile de repousser une décision, les règles suivantes ayant besoin des résultats des précédentes, et tout aussi difficile de revenir dessus. Difficile ne veut pas dire impossible, c'est essentiellement un problème d'expressivité du moteur d'application des règles et de facilité ergonomique pour l'expert d'accéder à ces possibilités. De plus la maintenance d'un ensemble de règles peut être difficile. Ces ensembles sont bien souvent équivalents à un programme traditionnel important au niveau de la complexité, mais ils bénéficient rarement des avantages internes (structuration, approche objet, abstraction...) et externes (outils de débogage, de trace, de profiling, de design) de la programmation moderne. Cependant ces approches n'ont pas que des inconvénients, loin de là. Déjà, et c'est fondamental, elles permettent d'expérimenter avec l'annotation. Aucun choix n'est définitif, et ajouter des niveaux de détails ou élargir la couverture peut ne représenter qu'un changement mineur par rapport à la difficulté, au coût et au temps de réannotation d'un corpus d'entraînement. De plus différentes langues et les différents domaines ont souvent bien des choses en commun et cela rend les approches symboliques bien plus portables qu'il peut n'y paraître au premier abord. Il suffit souvent de traduire les mots importants présents dans les règles et d'en réorganiser quelques unes pour transformer un système d'analyse pour le français en un capable de traiter décemment l'espagnol par exemple. Enfin, dans certains cas, la division en niveaux d'analyse linguistique différents peut être plus une convention qu'une contrainte forte. Si l'expert dispose d'un moyen d'en représenter plusieurs de façon unifiée il peut considérer pertinent de les traiter en même temps en travaillant par îlots de confiance, commençant par les annotations les plus sûres et s'y appuyant pour la suite, quels que soient les niveaux de ces annotations.

Pour le français peu de ressources sont disponibles publiquement. Des corpus ou des annotateurs statistiques existent pour les parties du discours. Nous disposons de corpus annotés en entités nommées en français mais avec une couverture de types relativement limitée (ESTER [Gravier, et al. 2004]).

Des corpus existent pour l'analyse syntaxique (EASY [Paroubek et al. 2006], PASSAGE [de la Clergerie, et al. 2008]) mais de taille insuffisante à l'heure actuelle pour construire des analyseurs statistiques. De même des taxonomies linguistiques sont développées (EuroWordNet, FrameNet, Lexique-Grammaire) mais leur couverture ou leur disponibilité semblent pour l'instant insuffisantes pour les exploiter directement. De plus il n'existe pas de consensus sur le choix des informations les plus utiles pour le dialogue en domaine ouvert ou même pour Question-Réponse, et encore moins sur comment les représenter. Le besoin d'expérimenter sur l'annotation est donc très présent. En conséquence nous avons décidé de nous baser essentiellement sur une approche symbolique et plus spécifiquement à base de règles. De plus, nous avons voulu garder la possibilité d'intégrer des analyses statistiques quand elles sont disponibles et performantes, en particulier pour les parties du discours. Ces besoins de flexibilité et d'intégration ont motivé nos travaux visant à la création d'un nouveau moteur générique d'analyse de la langue.

Chapitre 1

État de l'art

Les moteurs disponibles publiquement permettant à des experts d'écrire leurs propres règles d'analyse sont peu nombreux. Bien souvent ce moteur finit par être PERL. Il en existe cependant quelques uns qui méritent d'être regardés de près pour inspirer notre réflexion. Cass, de Steven Abney, est un peu le grand ancêtre. Créé au milieu des années 90, il propose un pur système de règles et a été moteur dans le développement du concept d'analyse incrémentale. GATE et UIMA, plus récents, sont des *frameworks*, des systèmes faits pour intégrer plusieurs modules d'analyse de la langue autour d'une représentation commune de l'état de l'analyse. NLTK est une librairie interfacée avec le langage python intégrant un large éventail d'algorithmes et de méthodes d'analyse de la langue. Enfin Corpus Query Processor (CQP) n'est pas un moteur d'analyse mais un système pour faire des recherches dans des corpus. Son langage de requêtes est cependant très intéressant et similaire avec la partie *matching* d'un langage de règles.

Sous quel angle examiner ces systèmes ? Le plus souvent un langage de règles est étudié d'un point de vue génératif, en déterminant l'ensemble des textes qu'il est capable, ou non, de reconnaître. La hiérarchie de Chomsky [Chomsky 1956], par exemple, est un travail fondamental du domaine, classifiant les grammaires formelles en quatre catégories suivant leur niveau d'expressivité. Cependant nous sommes dans un cadre d'analyse robuste où les contraintes sont un peu différentes. En effet le but est d'annoter autant d'informations que possible sans aucune assurance que la totalité d'un énoncé pourra être reconnu par les règles. Il faut donc prendre les décisions d'annotation tôt et de manière non ambiguë, car il n'est pas possible de compter sur une validation globale pour sélectionner les possibilités les plus pertinentes. Il est à noter qu'il est, par contre, possible de modifier des annotations dans des étapes ultérieures, pour les affiner ou même les corriger. La démarche est plus transformative que générative.

Les points importants sont donc différents. Le premier point concerne la capacité de représentation. Quels types d'informations peuvent être annotées de façon naturelle dans les représentations proposées. C'est un peu le pendant analytique de l'expressivité du point de vue génératif, en examinant

non pas quels types de textes peuvent etre générés, et donc reconnus, mais quels types d'informations peuvent être annotées, et donc détectées et ensuite exploitées. Le second point important est l'ergonomie globale du langage de règles. C'est une notion un peu floue mais qui regroupe les facteurs agissant sur la facilité d'écriture, de lecture et de maintenance en général des règles écrites. C'est donc ces points que nous mettons en priorité dans notre état de l'art.

1.1 Le système Cass

Le système Cass, par Steven Abney [Abney 1996], plus connu sous le nom de son packaging Scol, est un des précurseurs en la matière.

```
:chunk
        NP -> D ? N+ ;
        VP -> V-tns | Aux V-ing ;
:pp
        PP -> P NP ;
:clause
        S -> PP* NP PP* VP PP* ;
```

FIG. 1.1 – Exemple de règles Cass, tiré de [Abney 1997]

C'est un pur moteur de règles qui ne propose pas d'intégration avec d'autres approches. Il s'attend en entrée à des mots, si possible annotés en parties du discours. Un exemple d'analyse utilisant ce moteur est donné en figure 1.1. Les règles sont des expressions régulières traditionnelles qui s'appuient sur les tags ou les mots et substituent la zone reconnue par son label. Elles sont organisées en passes nommées, ici *chunk, pp* et *clause,* qui sont appliquées successivement. Il est aussi possible d'insérer des *actions* similaires dans l'esprit à celles de Lex [Lesk 1978] qui permettent de donner des valeurs à des attributs associés aux labels. Il n'est cependant pas possible d'utiliser ces valeurs dans les règles des passes suivantes. Le moteur sous-jacent fonctionne sur le principe d'une succession de *transducteurs déterministes finis*. Les expressions régulières sont transformées en automates finis déterministes, avec un traitement spécifique pour les ϵ-transitions dues aux actions.

Cass est une bonne mise en œuvre du principe de l'*analyse incrémentale* : chaque passe de l'analyse s'appuie sur les passes précédentes et permet à l'expert de procéder par expansion d'*îlots de confiance*. Ce type d'approche permet plus facilement d'assurer une certaine robustesse car, même si les structures les plus longues et complexes ne sont pas reconnues, les sous-structures les composant, plus simples, l'auront probablement été dans des passes précédentes. L'application a donc ainsi toujours de l'information utile, même si elle est partielle. Ce système de règles a plusieurs limitations. Les expressions régulières traditionnelles, très intéressantes d'un point de vue de l'implémentation en permettant leur transformation en automate fini déterministe, sont limitées au niveau de ce qu'elles permettent d'exprimer. En particulier il leur manque toute notion de *contexte,* ou en d'autres termes

on ne peut regarder autour d'un bloc à substituer pour prendre une décision. Il est de plus impossible avec ce moteur de revenir sur des annotations faites dans des passes précédentes ou même de regarder "en dessous" d'une substitution déjà effectuée.

Cass a été utilisé dans plusieurs domaines. On peut citer par exemple une analyse syntaxique légère (relations sujet-verbe, verbe-objet, nom-nom et adjectif-nom) pour l'extraction de concordances [Smadja 1993], l'analyse de questions pour des systèmes de Question-Réponse [Grau, et al. 2005b] ou de génération en langue de réponse [Schilder, et al. 2005] ou encore l'extraction automatique d'ontologies [Cimiano, et al. 2006].

1.2 Les frameworks GATE et UIMA

Plus qu'un système de règles comme Cass, GATE [Cunningham, et al. 2002] est avant tout un *système d'intégration* de divers modules d'analyse.

Texte				
^{0}Cyndi^{5} savo^{10}red t^{15}he so^{20}up.^{23}				
Annotations				
Id	Type	Début	Fin	Attributs
1	token	0	5	pos=NP
2	token	6	13	pos=VBD
3	token	14	17	pos=DT
4	token	18	22	pos=NN
5	token	22	23	
6	name	0	5	name_type=person
7	sentence	0	23	constituents=[1],[2],[3].[4],[5]

FIG. 1.2 – Exemple d'annotation sous GATE, tiré de [Gaizauskas, et al. 1996]

GATE s'appuie sur une bibliothèque Java commune qui implémente une structure de données représentant un document et ses annotations ainsi que le support nécessaire pour la gestion et la coordination de modules Java d'analyse de la langue. Un exemple d'annotation est donné figure 1.2. Chaque annotation est un quadruplet formé d'un type, d'une position de début et de fin en caractères, et d'un ensemble d'attributs. Ces attributs peuvent eux-mêmes référer à d'autres annotations. Cette structure assez peu contrainte permet d'encoder pratiquement n'importe quelle annotation, hiérarchique ou non. De nombreux modules sont disponibles traitant les problèmes d'entrée et de sortie dans des formats divers (texte, SGML, HTML, XML, RTF, email...), de tokenisation ainsi que diverses méthodes d'analyse. Une interface graphique est disponible pour aider à l'organisation des modules ainsi que pour visualiser les résultats.

```
Macro : MILLION_BILLION
({Token.string == "m"}|
{Token.string == "million"}|
{Token.string == "b"}|
{Token.string == "billion"}
)

Macro : AMOUNT_NUMBER
({Token.kind == number}
(({Token.string == ","}|
      {Token.string == "."})
{Token.kind == number})*
(({SpaceToken.kind == space})?
   (MILLION_BILLION)?)
)

Rule : Money1
      (
            (AMOUNT_NUMBER)
            ({SpaceToken.kind == space})?
            ({Lookup.majorType == currency_unit})
      )
   :money –>
      :money.Number = {kind = "money", rule = "Money1"}
```

FIG. 1.3 – Exemple de règles JAPE/GATE, tiré de [Gaizauskas et al. 1996]

Un des modules fournis permet d'utiliser le langage JAPE (Java Annotation Patterns Engine), créé pour l'occasion, pour écrire des règles d'annotation. Ces règles travaillent sur les annotations existantes et les modifient ou en produisent de nouvelles. Les annotations initiales sont produites par les modules de tokenisation, qui produisent des mots sous le nom d'annotation *Token* et de gazetteer qui cherchent des expressions d'après des listes et produisent des annotations *Lookup*. Un exemple de ces règles est donné figure 1.3. Les groupes entre accolades représentent des matchings élémentaires sur les annotations. Ils peuvent se faire sur n'importe quel champ des annotations et prennent la forme d'une conjonction de tests d'égalité ou d'inégalité, ou même d'une fonction Java. Ces matchings sont structurés avec les opérateurs d'expressions régulières classiques (parenthèses, répétitions, alternatives). Des sous-expressions peuvent être mises sous la forme de macros et utilisées ensuite. Les groupes reconnus peuvent ensuite donner lieu a une nouvelle annotation, *money* dans l'exemple, ou là encore exécuter une fonction Java qui peut faire les manipulations qu'elle veut sur les annotations. Les annotations n'étant pas naturellement séquentielles, elles sont pour les besoins des règles classées de par leur position initiale. Si plusieurs annotations commencent au même endroit et qu'un matching se fait sur elles il suffit qu'au moins une respecte la condition pour que le matching soit accepté. Quand

plusieurs la respectent le choix parmi elles est aléatoire. Le cas est dans la pratique rare d'après les auteurs.

Ce format de règles, avec la possibilité de tester n'importe quel attribut ainsi que de revenir à tout moment à des fonctions Java, offre une grande flexibilité, peut-être au détriment de la lisibilité. De même la structure d'annotation permet de représenter pratiquement n'importe quelle structure. Cependant cette structure a le défaut de ses qualités. Sans structuration explicite en arbres, relations, frames ou autre organisation sous-jacente à des classes d'analyse linguistiques JAPE doit s'appuyer sur un ordre semi-arbitraire des annotations pour l'application des matchings des règles. Cela complique les structurations en passes incrémentales d'un ensemble d'analyses car il est difficile de donner une priorité aux résultats des dernières passes, ce qu'une structure en arbres permet naturellement. De même, ne rien définir au niveau des attributs (noms, significations) revient à déplacer le problème vers les modules qui doivent de toute façon se mettre d'accord entre eux. Mais ceci fait que JAPE ne peut facilement supprimer les parties répétitives de sa syntaxe comme les *{ Token.string == "..." }* car le choix du type d'annotation *Token* et du nom d'attribut *string* n'est pas défini par GATE mais seulement une convention implicite des tokeniseurs actuellements implémentés.

GATE a des utilisations très variées. Nous pouvons citer par exemple KIM [Popov, et al. 2003], une plateforme d'annotation sémantique, comme utilisation de son aspect plateforme d'intégration et de gestion de documents. Ses capacités d'analyse ne sont pas en reste, soit en les utilisant telles quelles, par exemple la détection d'entités nommées dans le cadre d'un système de Question-Réponse [Mollá, et al. 2006], soit en développant des annotations adaptées en utilisant JAPE comme dans [Agatonovic, et al. 2008] pour l'annotation de brevets ou encore [Plamondon, et al. 2004] pour l'anonymisation de documents.

Le framework UIMA [Ferrucci & Lally 2004], en plein essor, est très similaire à GATE. Son principe est le même : intégrer un ensemble de composants travaillant sur la langue. Sa principale différence, en dehors de details d'implémentation comme le support de plus de langages de programmation pour les composants, est le support de plus de types de données. En effet UIMA permet d'annoter n'importe quel type de donnée organisable en séquence comme le texte, évidemment, mais également le son ou la vidéo. Ceci est tout simplement fait en généralisant le concept de position en caractères dans le texte en position dans le flux, quel que soit son type. Cela permet par exemple d'avoir un moteur de reconnaissance vocale comme composant UIMA, permettant d'aller plus loin dans l'intégration de systèmes. Cependant les difficultés de création d'un système d'analyse par règles dues à l'absence de structure explicite sont toujours présentes, ce qui doit expliquer pourquoi aucun ne semble disponible à l'heure actuelle. Il existe bien un composant, le *Regular Expression Annotator*, qui est capable de produire des annotations à partir d'expressions régulières appliquées sur le texte original. Cependant cela ne permet pas d'analyse incrémentale et sort donc du cadre de cet état de l'art.

```
grammar = r"""
        # chunk determiner/possessive, adjectives and nouns
        NP : {<DT|PP$> ?<JJ>*<NN>}
        # chunk sequences of proper nouns
                {<NNP>+}
"""
cp = nltk.RegexpParser(grammar)
sentence = [("Rapunzel", "NNP"), ("let", "VBD"), ("down", "RP"),
        ("her", "PP$"), ("long", "JJ"), ("golden", "JJ"), ("hair", "NN")]
```

```
» print cp.parse(sentence)
(S
        (NP Rapunzel/NNP)
        let/VBD
        down/RP
        (NP her/PP$ long/JJ golden/JJ hair/NN))
```

FIG. 1.4 – Exemple de chunking par expressions régulières dans NLTK sur la phrase annotée en partie du discours *Rapunzel/NNP let/VBD down/RP her/PP long/JJ golden/JJ hair/NN* (tiré de [Bird et al. 2009]).

1.3 La librairie NLTK

Le *Natural Language Toolkit* (NLTK) [Bird, et al. 2009] est une bibliothèque s'interfaçant avec le langage de programmation *python* et offrant une large gamme d'approches pour l'analyse de la langue. Il propose ainsi différentes méthodes de tokénisation, annotation en parties du discours, chunking, détection d'entités nommées, analyse syntaxique, etc, allant même jusqu'au calcul de prédicats pour aider au raisonnement au niveau sémantique. De plus ces méthodes proposées couvrent aussi bien les approches statistiques que symboliques. Nous nous intéressons plus spécifiquement à deux de ces méthodes : le chunking par expressions régulières et l'analyse, souvent syntaxique, par grammaires non-contextuelles.

Cette librairie, s'appuyant sur un langage de programmation existant, exploite les structures de données qu'il propose. Un texte est une simple chaine de caractères. Le résultat d'une tokénisation est une liste de chaines contenant les mots. L'analyse d'une telle liste en parties du discours (POS) a pour résultat une liste de paires *(mot, POS)*. Cette liste de paire peut ensuite être utlisée pour une analyse syntaxique qui donne comme résultat un arbre, et ainsi de suite. Il n'y a pas une représentation commune à toutes les étapes de l'analyse mais un ensemble de représentations adaptées à chacune.

Deux méthodes spécifiques nous ont paru pertinentes pour notre état de l'art. La première est un

```
grammar2 = nltk.parse_cfg("""
        S -> NP VP
        NP -> Det Nom | PropN
        Nom -> Adj Nom | N
        VP -> V Adj | V NP | V S | V NP PP
        PP -> P NP
        PropN -> 'Buster' | 'Chatterer' | 'Joe'
        Det -> 'the' | 'a'
        N -> 'bear' | 'squirrel' | 'tree' | 'fish' | 'log'
        Adj -> 'angry' | 'frightened' | 'little' | 'tall'
        V -> 'chased' | 'saw' | 'said' | 'thought' | 'was' | 'put'
        P -> 'on'
        """)
```

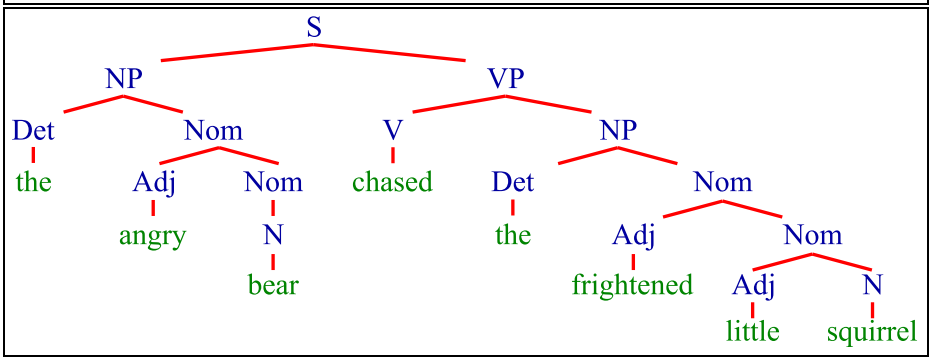


FIG. 1.5 – Exemple de grammaire non-contextuelle pour l'analyse syntaxique et d'arbre résultant dans le cadre de NLTK (tiré de [Bird et al. 2009]).

chunker par expressions régulières dont nous pouvons voir un exemple d'utilisation figure 1.4. Le moteur propose de définir des zones à regrouper via des expressions régulières sur les annotations en parties du discours. Ces expressions sont à deux niveaux : un premier niveau, que l'on pourrait nommer caractères, reconnait les annotations elles-mêmes. Ce sont les sous-expressions présentes dans les <...>. Ces sous-expressions sont ensuite combinées dans un deuxième niveau d'expressions régulières qui cherchent elles à reconnaitre des successions spécifiques de parties du discours. Une succession reconnue permet de délimiter un chunk, construisant une structure en arbre. Les ambiguités sont résolues de manière prédéterminée, les chunks les plus proches du début de la phrase sont prioritaires et en second lieu les règles les plus proches du début de la liste. Ces expressions peuvent aussi s'appliquer aux arbres, permettant une analyse incrémentale. Il est à noter qu'aucune syntaxe n'est proposée pour accéder aux mots ni, dans un cas incrémental, au contenu des chunks déjà annotés. Les possibilités de cette méthode semblent similaires à ce que propose Cass.

La deuxième méthode est l'analyse, habituellement syntaxique, par grammaires non-contextuelles. Un exemple d'une telle grammaire et d'un résultat qu'elle permet d'obtenir est donné figure 1.5.

Plusieurs méthodes de parsing sont proposées, en particulier le *chart parsing*, qui est la référence en la matière. Cependant les grammaires non-contextuelles simples ont des limites. En particulier plusieurs arbres syntaxiques sont bien souvent possibles, et les non-terminaux (NP, VP, ...) peuvent manquer d'informations, comme par exemple le type de verbe (transitif ou non, ...) ou de nom (humain, animé, objet...). Pour aider à ces problèmes NLTK propose de rajouter un poids à chaque règle, permettant d'associer un score à chacun des arbres syntaxiques possibles. Il permet aussi de rajouter des *traits*, paires attribut/valeur, aux non-terminaux, permettant de remonter de l'information des feuilles vers la racine et d'utiliser cette information comme contrainte quand cela est utile.

NLTK connait des utilisations variées. Offrant un accès facile à de nombreux algorithmes et méthodes fondamentaux en analyse de la langue, il est souvent utilisé dans l'enseignement. Nous le rencontrons aussi dans des systèmes variés où des parties spécifiques sont utilisées. Par exemple [Barrón-Cedeño, et al. 2009] utilisent son module de chunking par expression régulières pour extraire des termes en espagnol. Ou encore [Blunsom 2004] qui utilise la bibliothèque pour la tokénisation, l'analyse en partie du discours et le chunking dans le cadre de son préprocessing pour la détection automatique de rôles sémantiques.

1.4 CQP - Corpus Query Processor

CQP [Christ 1994b], qui fait partie de l'*IMS Corpus Workbench* [Christ 1994a] n'est pas à proprement parler un moteur d'analyse de la langue. C'est un langage créé pour permettre des recherches dans des corpus pré-annotés pour en tirer en particulier des informations de collocations. Ce problème de définition de langage est toutefois similaire à celui de la définition de la partie *matching* d'un moteur de règles, d'où l'intérêt de l'observer.

Mot	POS	Lemme
A	DT	a
form	NN	form
of	IN	in
asbestos	NN	asbestos
once	RB	once
used	VBN	use
to	TO	to

FIG. 1.6 – Extrait de Penn Treebank vu en tant que corpus CQP. La représentation est similaire à une table de bases de données, avec les colonnes typant les informations et les lignes contenant le corpus.

Les corpus pré-annotés traités par CQP sont représentés sous une forme relativement simple, qui peut être assimilée à une table de base de données, comme visible sur la figure 1.6. Un certain nombre d'attributs tels que mot, partie du discours, lemme sont définis, formant les colonnes. Les lignes forment le corpus lui-même. La notion de séquence est ici naturelle, facilitant grandement la définition

du langage.

```
"Clinton" ;
[word = "Clinton"] ;
"alumini ?um" ;
[word = "rain" & pos = "NN"] ;
"Bill" "Clinton" ;
[pos = "NP"] "of" ? [pos = "NP"] ;
[lemma = "give"] [pos != "SENT"]{0, 5} "up" ;
```

FIG. 1.7 – Exemples du langage CQP, essentiellement tirés de [Christ, et al. 1999]

Le langage se décompose en 3 niveaux de filtrage successifs :
– Les expressions régulières de caractères sur le contenu des cases de la table
– Les expressions booléennes sur ces résultats sélectionnant des lignes
– Les expressions régulières sur ces lignes sélectionnant les passages finaux
Nous en donnons des exemples figure 1.7. Les deux premiers sont équivalents et cherchent le mot Clinton. En l'absence de choix explicite du nom d'attributs *word* est pris par défaut. Ce défaut est changeable globalement si, par exemple, les recherches sur les parties du discours sont les plus fréquentes. Le troisième montre que les recherches individuelles de valeurs sont bien des expressions régulières de caractères, cet exemple cherchant les instances de aluminium (orthographe anglaise) et aluminum (orthographe américaine). Le suivant montre un exemple de construction d'expressions booléennes au-dessus de ces expressions régulières de base en cherchant toutes les instances de *rain* en tant que nom commun (et non verbe). Nous avons ensuite des exemples de recherche de séquence, qui sont là aussi des expressions régulières, en commençant par une simple paire de mots, *Bill Clinton*, suivie d'une recherche de paires de noms propres optionnellement séparées par *of*, et enfin une recherche des instances d'utilisation du verbe à particule *to give up*, en autorisant jusqu'à 5 mots qui ne soient pas une ponctuation de fin de phrase (SENT) entre le verbe et la particule.

La syntaxe de ce langage de requête est très intéressante par son équilibre entre expressivité et lisibilité. La décomposition entre recherche intra-mot et recherche inter-mots permet une bien plus grande lisibilité que ce que des simples expressions de caractères avec marqueurs de limites de mots, comme propose PERL, permettent. La présence des expressions booléennes permet d'utiliser des conditions négatives, mais la portée reste limitée au niveau du mot. Or il existe des cas où l'on veut pouvoir exprimer de telles contraintes négatives sur des expressions composées de plusieurs mots.

CQP est bien évidemment populaire en linguistique de corpus. On peut citer par exemple [Heiden & Lafon 2002] qui extraient de nombreuses informations statistiques sur les coquilles dans l'*Encyclopédie* de Diderot et d'Alembert [Diderot & d'Alembert 1751–1772]. Mais on peut aussi trouver des exemples dans le domaine de Question-Réponse avec en particulier [Eckle-Kohler 1998] où il est utilisé pour extraire automatiquement des lexiques ciblés.

1.5 Discussion

Nous avons présenté quatre systèmes proposant un langage de règles. Chacun a ses qualités et ses limitations et nous donne des pistes pour notre propre moteur.

Cass, le grand ancêtre, est un pur moteur d'analyse par règles. Son point fort a été d'introduire la notion d'*analyse incrémentale*. L'analyse se fait en une série de passes consécutives, chacune rajoutant des annotations complétant celles construites précédemment. Une telle approche permet de travailler par îlots de confiance, annotant d'abord les informations les moins ambiguës et s'appuyant sur ces annotations pour résoudre les cas moins tranchés. L'implémentation proposée de cette approche a cependant deux limitations. Il n'est tout d'abord pas possible de regarder "sous" une annotation mise par une passe. Ces annotations sont des substitutions, et les mots ou sous-annotations remplacés ne sont plus accessibles par les règles. Il est pourtant utile de pouvoir les consulter au besoin, et notre langage de règles ne devra donc pas avoir une telle limitation. La seconde limitation est l'impossibilité de revenir sur une annotation. Des informations de plus haut niveau peuvent êtres utiles pour préciser des informations plus locales. Par exemple il est possible de conclure grâce à la structure globale de la phrase qu'un élément qui avait initialement été détecté comme une date simple est en pratique une date de naissance. Pour permettre de tels raffinements le langage de règles doit donc permettre de modifier des annotations et non uniquement d'en ajouter.

GATE et UIMA sont des réponses au problème de l'intégration de multiples modules d'analyse fonctionnant sur des niveaux ou des approches différentes. Cette intégration se fait à travers une *représentation commune* de l'état de l'analyse. Cette approche nous parait très pertinente mais se pose le problème de la définition de cette représentation. Celle proposée est extrêmement générique et peut représenter tous les types d'analyse auxquels nous avons fait allusion. La contrepartie est que les informations contenues dans cette représentation sont très faiblement structurées. Ce manque de structure pose des difficultés pour la définition d'un système de règles capable de les exploiter. La représentation que nous allons définir devra ainsi être suffisamment structurée pour représenter naturellement les types d'analyse qui nous intéressent.

NLTK est une librarie utilisable à partir du langage python, proposant des implémentations de nombreux algorithmes et méthodes utiles pour l'analyse de la langue. Les structures de données intrinsèques de python sont utilisées pour définir un ensemble de représentations spécifiques aux différentes annotations. C'est une approche intéressante, permettant d'avoir à chaque fois une représentation collant au mieux à la structure intrinsèque des annotations. Elle a cependant l'inconvénient de rendre plus difficile l'intégration de différents niveau d'analyse. Il parait par exemple relativement complexe d'intégrer en une seule représentation les résultats d'une extraction d'entités nommées et d'une analyse syntaxique. Cette librairie propose deux moteurs d'annotation par règles qui nous paraissent pertinents pour nos besoins. Le premier est un chunker par expressions régulières suffisamment similaire dans ses capacités à Cass pour que les mêmes remarques s'appliquent. Le second permet d'analyser un texte à partir de grammaires non-contextuelles. Pour augmenter l'expressivité du moteur il permet l'ajout de poids sur les règles et de traits sur les non-terminaux. Ce type de grammaire est

classique en analyse syntaxique mais rien n'empêche de d'en utiliser pour d'autres besoins. A priori elles paraissent cependant difficiles à utiliser dans un cadre d'analyse robuste. En effet, l'absence de contextes explicites pose plusieurs problèmes. Avec des grammaires contextuelles, une partie des ambiguïtés entre règles peut être résolue par un simple test sur les annotations ou mots voisins. C'est le cas par exemple d'une grande partie des ambiguïtés qui apparaissent au moment d'une annotation en entités nommées à partir de listes. Par exemple dans *le conseil général de la Loire* et *la Loire prend sa source en Ardèche*, le nom propre *Loire* n'est pas ambigu dans son contexte entre département et fleuve, mais l'est hors contexte. Dans un cadre non-contextuel ces ambiguïtés doivent être levées via l'application, ou l'échec d'application, des règles suivantes, et ce jusqu'à avoir réussi à construire un arbre d'analyse complet. Ce déplacement de la résolution des ambiguïtés rend toute analyse partielle, où la résolution n'a pas encore été effectuée, de qualité inconnue et complique d'autant l'écriture des grammaires. Enfin les traits, attributs des non-terminaux prenant la forme de paires type/valeur, sont utilisés pour remonter des informations plus détaillées que ce qu'un simple label tel que *V* peut représenter en soi. Ces informations peuvent porter par exemple sur la transitivité du verbe. Mais cette approche a ses limites. Décider quelles informations seront utiles doit être effectué dès les plus bas niveaux des annotations. Ces informations doivent en plus être regroupées correctement pendant la construction de l'arbre d'analyse. Il parait plus intuitif, et meilleur d'un point de vue structuration, de pouvoir observer le contenu des sous-arbres au moment précis où l'information est utile plutôt que devoir la constituer à l'avance.

Enfin CQP n'est pas un moteur d'analyse mais un système de recherche dans des corpus. La syntaxe de ses commandes d'extraction est cependant intéressante : elle montre l'expressivité et la relative simplicité de lecture des *expressions régulières de mots*. Les mots, dans le cadre des langues auxquelles nous nous intéressons et en particulier le français, sont une unité de recherche, de déplacement, de répétition souvent plus pertinente que les caractères. De plus la liberté de formatage des expressions que l'on obtient favorise la lecture. Cependant la syntaxe proposée n'est performante que pour les mots du texte initial. Atteindre les annotations, parties du discours ou lemmes dans les exemples que nous avons montrés, nécessite une écriture bien plus lourde mettant en jeu un test explicite d'égalité. Nous devrons donc proposer une méthode alternative pour atteindre ces annotations tout en conservant une syntaxe simple.

Nous présentons dans la suite de cette partie le moteur que nous avons conçu. Le prochain chapitre est consacré à cette présentation du point de vue d'un utilisateur : organisation générale, représentation proposée, les différentes transformations, le langage de règles et les entrées-sorties. Le chapitre suivant regarde ce même système de l'intérieur en abordant les aspects algorithmiques mis en jeu. Ce sont en effet les décisions prises à ce niveau là qui vont décider de la viabilité finale du moteur. Enfin nous terminons par une évaluation de ce système. Évaluer un tel moteur est un problème difficile en soi car la qualité des analyses produites dépend en premier lieu de la qualité des règles que l'utilisateur écrit. Il en découle que la qualité du moteur peut être illustrée par ce qu'il a permis de construire. Nous présentons donc plusieurs cas d'utilisation, incluant des systèmes complets construits avec le moteur. Des mesures quantitatives sont cependant possibles et nous présentons pour le plus important de ces systèmes quelques mesures de performance.

Chapitre 2

Un moteur à base de transformations

Notre moteur d'analyse a pour objectif de permettre de construire des analyseurs robustes incrémentaux. En plus des besoins habituels de vitesse et de flexibilité nous voulions éviter d'être lié à un modèle linguistique ou d'analyse spécifique, ou même de choisir un camp dans l'éternel débat règles contre statistiques. Il est d'ailleurs intéressant de noter que les meilleurs analyseurs en parties du discours sont statistiques alors que pour certaines langues incluant le français les meilleurs analyseurs syntaxiques sont à base de règles [Branco, et al. 2003 ; Paroubek et al. 2006].

Nous avons décidé de construire ce moteur sur la base d'une *représentation commune de l'état de l'analyse*, capable de contenir toute l'information que l'analyse peut extraire. L'analyse peut alors être structurée en un ensemble de passes incrémentales, chacune ajoutant ses propres résultats mais conservant la possibilité de modifier les résultats des passes précédentes. De plus définir une version texte simple de cette représentation permet d'utiliser des outils externes qui iront faire leurs propres modifications, permettant de tester des approches alternatives sans avoir besoin d'intégrer immédiatement toutes les possibilités dans l'outil global.

Nous arrivons donc à une définition de l'analyse comme une série de transformations élémentaires sur une représentation commune. Ces transformations peuvent se regrouper en 3 catégories principales :
– Tranformations à base de règles
– Tranformations à base statistique
– Tranformations algorithmiques
Les sections suivantes présentent la représentation commune puis les différentes transformations que nous utilisons.

2.1 Représentation commune de l'état de l'analyse

La représentation interne de l'état du texte analysé est primordiale dans tout système d'analyse de la langue, cela d'autant plus quand ce système est structuré en moteur de transformations. Elle doit satisfaire un certain nombre de besoins en partie contradictoires :
– elle doit pouvoir contenir tous les types d'informations qui nous intéressent,
– elle doit être lisible par un humain une fois transformée dans un format texte,
– elle doit être complète : il ne doit pas y avoir de références à des entités externes non présentes dans son contenu, telles que la phrase initiale ou les résultats de transformations précédentes.

Au-delà de ces besoins de base se pose le problème du niveau de structuration de la langue que l'on veut avoir explicitement dans la représentation. Un extrême tel que la représentation de GATE n'impose aucune structure sur les annotations. La contrepartie est qu'aucune structure n'est utilisable implicitement par les transformations, compliquant comme nous l'avons vu l'écriture d'un moteur de règles performant.

En pratique, les annotations de la langue tendent à tourner autour de 3 axes. Par ordre de complexité de représentation, nous avons :
– *tags* où des symboles sont associés aux mots, tels des parties du discours
– *blocs arborescents* où des blocs typés regroupent des ensembles connexes de mots ou de types, et ce récursivement. Les entités nommées ou encore les analyses en composants suivent une telle structure.
– *relations entre entités* où des liens typés, binaires ou n-aires, relient mots ou blocs ensemble. L'analyse syntaxique ou les rôles sémantiques suivent ce genre de formalisation.

Idéalement, ces trois types d'annotation devraient être représentables. Cependant la complexité augmentant il devient difficile de définir des syntaxes de règles lisibles. Nous avons donc décidé de nous limiter aux deux premiers types pour obtenir une *forêt d'alternatives*. Un exemple est donné figure 2.1 dans le cas d'une analyse simple en parties du discours et constituants.

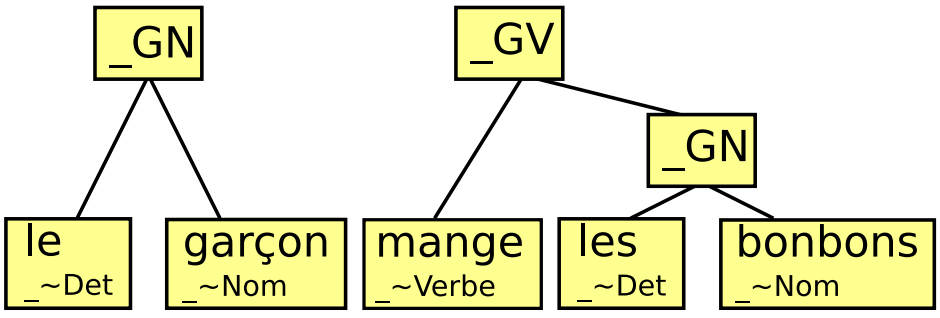


FIG. 2.1 – Exemple d'utilisation de la représentation commune pour encoder une analyse syntaxique simple en composants. Les nœuds du bas tirent parti des alternatives pour combiner mot et partie du discours. Les arbres servent à représenter les composants.

Dans notre cas, la forêt est un vecteur ordonné d'arbres. Chaque arbre et sous-arbre correspond pour cette analyse à un constituant, le nœud racine donnant son type. La notion d'*alternative* est utilisée dans les feuilles où chaque mot de la phrase initiale a en alternative possible sa partie de discours associée. Du point de vue de la représentation ces deux valeurs sont équivalentes pour représenter le nœud. Nous avons en particulier décidé de pas les typer explicitement. En effet les écritures du type *pos=NP*, comme nous avons vu dans CQP, sont rapidement assez lourdes. Nous avons préféré utiliser une distinction de types implicite via l'utilisation de vocabulaires disjoints. En particulier les parties du discours sont conventionellement préfixées par _∼ et les autres tags, syntaxiques dans notre exemple, par _. Ces conventions sont suffisantes pour éviter les collisions avec les mots normaux.

Cette structure nous permet de représenter efficacement toutes les analyses basées sur des arbres ou des labels. Il est à noter qu'elle peut être interprétée de deux manières : soit une forêt de nœuds, soit un vecteur de nœuds contenant chacun optionnellement un vecteur de nœuds en dérivation, et ainsi de suite récursivement. Cette vision plus linéaire en vecteur de vecteurs est très utile pour le moteur de transformation par règles.

Enfin construire une représentation textuelle raisonnablement lisible depuis une telle représentation est simple. Il suffit d'utiliser une syntaxe inspirée d'XML pour l'arborescence et une barre verticale (*pipe*) pour séparer les alternatives. L'exemple devient ainsi :

```
<_GN> le|_∼Det garçon|_∼Nom </_GN> <_GV> mange|_∼Verbe <_GN>
les|_∼Det bonbons|_∼Nom </_GN> </_GV>
```

La représentation ainsi définie, nous pouvons nous intéresser aux transformations qui vont agir dessus.

2.2 Transformations à base de règles

D'un point de vue général, un système de transformation par règles peut être décomposé en 3 parties :
- *Pattern Matching*, avec une règle sélectionnant où elle s'applique
- *Transformation de la représentation*, où la règle transforme localement l'arbre à l'emplacement choisi
- *Stratégies de résolution de conflits et d'application des règles*, où une règle est choisie quand plusieurs peuvent s'appliquer au même endroit, et en général où dans la représentation les règles doivent être appliquées

2.2.1 Pattern matching par expressions régulières

Le *Pattern Matching* est la sous-partie d'une règle qui définit où la transformation s'applique. Comme nous l'avons vu dans l'état de l'art, dans le cadre du Traitement Automatique de la Langue les *Ex-*

pressions Régulières, ou encore *Expressions Rationnelles* sont l'outil le plus utilisé pour cette tâche. Le concept original des expressions régulières a beaucoup évolué depuis sa création et certaines de ces évolutions se sont révélées utiles pour le traitement de la langue. Nous avons de plus ajouté nos propres extensions.

Nous avons décidé, à l'instar de Scol et CQP, de travailler uniquement sur des mots et non des caractères comme le veut la tradition. En pratique, cela signifie que l'unité élémentaire de comparaison et de répétition est le mot et que les espaces ne sont pas significatives au-delà de leur rôle de séparateur. La facilité de lecture en est grandement améliorée, permettant à la personne écrivant les règles de choisir sa propre présentation. Mais cela va plus loin. Un des problèmes posés par les expressions régulières est la difficulté à en réutiliser des parties. Par exemple nous voudrions pouvoir construire une liste des conjonctions de coordination du français (liste de mots). Ou encore détecter les conjugaisons d'un verbe donné (mini-expression régulière mettant en jeu une alternative entre expressions multi-mots). Dans un cadre tel que PERL ce genre de besoin est satisfait en utilisant des variables contenant les sous-expressions sous forme de chaîne de caractères. Cependant ces sous-expressions ne sont pas analysées syntaxiquement et les ajouter au milieu d'une expression est peu lisible. Avoir des mots comme unité de base permet de définir des classes de mots correspondant à des *classes nommées*, simples listes de mots, et des *macros*, sous-expressions définies au préalable. Nous définissions tout mot préfixé par % comme une classe et par & comme une macro. Ceci permet par exemple de détecter un certain nombre de fonctions politiques américaines en anglais avec un petit nombre de règles simples :

&towns : Washington | New York | Boston | ... ;
&states : Alabama | Mississippi | New York | ... ;
&districts : *&towns* | *&states* ;
%titles : mayor congressman senator governor ;
_Office : *%titles* of *&districts* ;

En utilisant une telle décomposition en classes et macros les règles restent simples alors qu'une expansion complète ne serait que très difficilement maintenable.

Une amélioration classique, que nous avons vue dans CQP, concerne le contrôle fin de l'opérateur de répétition. L'opérateur original de Kleene ne propose pas de limite sur le nombre de répétitions et, traditionnellement, essaie d'obtenir la séquence la plus longue possible. Pour l'analyse de la langue un opérateur permettant d'obtenir la séquence la plus courte est souvent utile. La version longue est habituellement appellée *greedy*, et la version courte *shy* ou *lazy*. De plus la possibilité de préciser des limites hautes ou basses sur le nombre de répétitions est là encore très utile.

Une autre amélioration, souvent négligée dans le domaine des grammaires formelles et pourtant extrêmement utile pour l'écriture de règles, est la capacité à tenir compte du contexte. Une méthode venant de PERL, qui l'avait lui-même reprise semblerait-il de PCRE, qui nous a parue intuitive et efficace est la notion de *lookahead*. Un lookahead est une sous-expression régulière que le moteur essaie

d'appliquer à la position courante. Cette application peut se faire vers la droite (lookahead avant) ou la gauche (lookahead arrière ou lookbehind). De plus on peut exiger que l'application réussisse (lookahead positif) ou bien, au contraire, qu'elle échoue (lookahead négatif). Par exemple, dans la suite de l'exemple précédent, il est possible de détecter les noms d'états qui ne sont pas ambigus avec un nom de ville. (? ! ...) est l'opérateur de lookahead négatif avant :

_unambiguous_state : (? ! *&towns*) *&states* ;

Travailler sur des mots a cependant un inconvénient : il n'est pas possible a priori de classifier les mots sur leur structure interne. Il est en effet utile de pouvoir détecter les nombres, les mots entièrement en majuscules (acronymes), ceux avec une majuscule en tête (noms propres), etc. Trois approches principales sont envisageables : accepter des expressions régulières de caractères pour classer les mots (approche CQP), créer une transformation spécifique pour classer les mots (approche GATE) ou simplement prédéfinir un certain nombre de classes utiles, qui peuvent être étendues suivant les besoins. Nous avons choisi la dernière possibilité. Bien que limitant plus l'utilisateur, elle a trois avantages : il est possible de choisir une syntaxe qui s'intègre bien dans les règles. Il est relativement facile d'obtenir de bonnes performances. Et surtout il est possible de créer des *catégories paramétrables*. En effet, en plus de catégories simples comme acronyme ou nom propre, nous avons pu ajouter une catégorie *intervalle de nombres*. Un utilisateur peut ainsi écrire *%number(1900,2050)* pour reconnaître les valeurs pouvant désigner des années avec une forte probabilité. Devoir les définir dans une autre transformation serait bien moins pratique, tout comme devoir écrire des expressions régulières de caractères reconnaissant un intervalle. De plus nous avons séparé les nombres en *cardinaux* (1, 2, 3) et *ordinaux* (1er, 2e, 3e). Les ordinaux sont conventionnellement écrits comme leur valeur suivie d'un tiret (1-, 2-, 3-), charge à une autre transformation de les marquer ainsi.

Il ne faut pas négliger la structure intrinsèque de la représentation. Les expressions régulières ne peuvent travailler que sur des structures linéraires, or nous avons là un arbre. Nous avons d'ailleurs remarqué dans le cas de Scol l'impossibilité d'accéder au texte original dès qu'il est masqué par une substitution. Cependant accéder à des niveaux inférieurs de l'arbre est très utile. Ce n'est pas par exemple parce que le groupe verbal a été identifié et annoté qu'il n'est pas utile de pouvoir vérifier de quel verbe il s'agit. Nous avons ajouté des opérateurs permettant de descendre dans les arbres. Leur principe est simple : ils essaient d'appliquer une sous-expression régulière quelque part dans les descendants du nœud en cours d'examen. Définir le "quelque part" est la raison sous-jacente au besoin de plusieurs opérateurs. Nous avons défini trois variantes : la plus simple descend d'un niveau et essaie d'appliquer l'expression régulière sur tous les nœuds situés à ce niveau. Pour permettre un ancrage aux extrémités de la dérivation, les opérateurs traditionnels de début et fin de ligne (^ et $) sont redéfinis dans un tel contexte pour signifier début et fin de dérivation. Une seconde version de l'opérateur essaie l'expression régulière sur tous les nœuds dérivés du nœud de départ, quelle que soit leur profondeur. Enfin la troisième variante est intermédiaire : elle tente d'appliquer la sous-expression à tous les nœuds descendants qui n'ont pas eux-mêmes de descendants. Elle permet en pratique de ne regarder que le texte original en ignorant autant que possible les annotations qui ont été effectuées. Un exemple de règle utilisant de telles constructions est donné en figure 2.2.

```
%vehicule : voiture voitures bus car cars autobus autocars avion avions ;
_transport : (_subs « %vehicule » | _NN <* %vehicule *>) ;
```

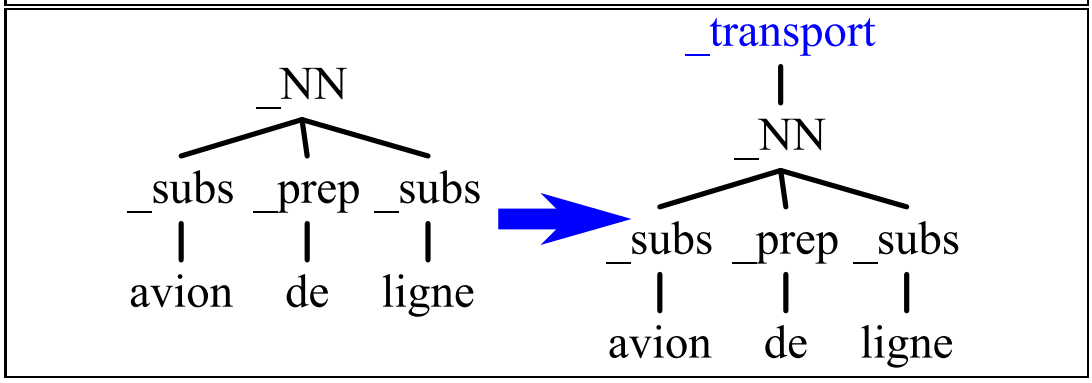


FIG. 2.2 – Exemple de règle utilisant la descente dans les arbres pour détecter les moyens de transport.

Enfin le but d'une règle est de définir une ou des zones où la transformation doit s'appliquer. Pour cela les traditionnelles *parenthèses de substitution*, aussi appellées *groupes de substitution*, sont utilisées. Elles permettent de définir des blocs contigus de nœuds situés au même niveau dans la représentation qui seront ensuite les points d'ancrage de la transformation à effectuer. Le besoin d'avoir une et une seule zone sélectionnée par groupe de substitution interdit de les avoir dans la sous-expression d'une répétition, d'une alternative ou d'un lookahead, mais c'est la seule contrainte. Elles peuvent en particulier se trouver à l'intérieur d'un opérateur de descente dans les arbres, permettant de désigner des nœuds qui ne sont pas situés au plus haut niveau dans la représentation.

2.2.2 Transformation de la représentation

Une fois qu'une ou plusieurs zones sont sélectionnées par une règle, une transformation va pouvoir s'appliquer. Nous avons défini deux catégories de règles, les règles *passives* et les règles *actives*.

Les règles passives se contentent d'associer des noms aux zones. Par exemple la règle suivante associe le nom *_pers* aux mots *Albert Einstein* quand ils sont trouvés dans la représentation :

```
_pers : (Albert Einstein) ;
```

Que faire de ces noms est décidé à plus haut niveau. Nous avons deux possibilités, la *substitution* et le *tagging*. La substitution crée un nouveau nœud qui va remplacer la zone. Ce nœud va avoir comme alternative unique le nom de la zone et comme dérivation le contenu de la zone. Elle va donc en l'occurrence construire une sous-structure : *<_pers> Albert Einstein </_pers>*. Le terme substitution est utilisé car là où le nom Albert Einstein était visible c'est ensuite le tag _pers qui l'est.

Le tagging ajoute en alternative le nom de la zone à tous les nœuds la constituant. La sous-structure construite sera là *Albert|_pers Einstein|_pers*.

Les règles actives décident directement du résultat de l'application. Quatre opérateurs sont possibles : destruction de nœuds et remontée de leur descendants, destruction complète de nœuds et de leur descendants, replacement des alternatives présentes dans des nœuds et replacement complet d'un ensemble de nœuds, descendants compris. Par exemple cette règle simple va détruire toutes les zones qui ont été annotées _filler :

%delete_tree(%1) : (_filler) ;

Dans cette règle, %1 désigne la première zone, sachant qu'il n'y en a en l'occurrence qu'une seule. Une règle plus complexe peut construire directement une structure complète :

%replace_tree(%1, _pers «_prenom «%2» _nom «%3» ») : ((%prenom) (%nom)) ;

Les classes %prenom et %nom sont considérées contenir des listes de noms propres, et l'opérateur « » correspond à la création d'un sous-arbre. Une telle règle pourrait créer la sous-structure *<_pers> <_prenom> Albert </_prenom> <_nom> Einstein </_nom> </_pers>*.

Ces règles actives ne sont pas très souvent utilisées mais sont en pratique très importantes. Ce sont elles qui permettent de revenir sur des décisions prises dans des passes précédentes. Bien au-delà de la correction d'erreur, elles permettent de compléter des décisions locales une fois que des informations de plus haut niveau ont pu être extraites.

2.2.3 Stratégies de résolution de conflits et d'application des règles

Parfois plusieurs règles de la même passe peuvent s'appliquer au même endroit dans la représentation. Nous sommes alors dans le cas d'un *conflit*, et il doit être résolu en décidant quelle règle s'applique. Ce problème fait partie de la *stratégie d'application des règles*, qui vise à décider où dans la représentation le moteur doit tenter d'appliquer les règles.

Nous commençons par définir un algorithme de décision global en cas de conflit entre deux règles :
– si les deux règles ont un niveau de priorité différent (donné dans la définition de la règle avec une valeur par défaut), la plus prioritaire gagne
– si les deux règles s'appliquent à des zones de taille différente, la règle englobant la zone la plus grande gagne
– si les deux règles sont des règles différentes, la première règle définie dans le fichier de règles gagne

– sinon le matching dont la zone est la plus à gauche gagne

À partir de cet algorithme de décision, nous avons testé plusieurs stratégies d'applications des règles et deux se sont révélées utiles en pratique. Nous nommons la première *résolution globale*. Toutes les règles sont essayées sur tous les nœuds racine de la représentation, donnant un ensemble de zones de matchings et règles associées. Tous les conflits sont alors résolus via l'algorithme indiqué. Les règles qui restent ont alors les transformations associées appliquées. C'est la stratégie qui est en pratique la *moins surprenante*. Un expert linguiste comprend bien pourquoi une règle a été choisie plutôt qu'une autre.

La seconde stratégie a été développée pour le problème spécifique de la reconstitution des nombres à partir de leur expression en mots. Les nœuds de haut niveau sont pris un par un de gauche à droite. Toutes les règles sont essayées sur le nœud pris, et la recherche s'arrête dès qu'un matching a lieu. Si plusieurs règles s'appliquent il y a nécessairement conflit, et le vainqueur est choisi avec le même algorithme. La transformation associée est alors appliquée et la recherche recommence au début. Le cycle s'arrête quand plus aucune règle ne s'applique. Nous appelons cette stratégie *récursion gauche*, la version symétrique, *récursion droite* peut être utile aussi suivant les langues.

2.3 Transformation statistique : le TreeTagger

Comme nous l'avons vu dans l'introduction, les approches statistiques dans l'annotation de la langue ne sont pas à négliger. Le problème de la disponibilité de données ou de modèles dans les langues qui nous intéressent a cependant été un frein à l'intégration de telles approches dans notre moteur d'analyse. Nous avons cependant trouvé utile d'intégrer un système d'annotation en parties du discours nommé *TreeTagger* [Schmid 1994 ; Schmid 1995], pour lequel des modèles sont disponibles pour plusieurs langues européennes.

Le principe de fonctionnement du TreeTagger combine deux modèles, un donnant pour chaque mot l'ensemble des annotations possibles avec les probabilités associées, et un autre calculant une probabilité pour toute succession d'annotations. Les deux modèles sont combinés via l'algorithme de Viterbi pour obtenir l'annotation la plus probable. Cette structure est illustrée figure 2.3. L'annotation en parties du discours est d'ailleurs une des premières utilisations de l'algorithme de Viterbi dans le domaine de l'analyse de la langue [Derose 1989 ; Church 1988].

L'originalité du TreeTagger est dans la construction de son modèle de successions. Dans le cadre qui nous intéresse, le modèle de successions, dit trigramme, doit être capable, à partir des deux parties du discours précédant le mot à classifier, de donner une probabilité pour chacune des parties du discours possibles. Ce type de modèle est construit à partir de mesures statistiques dans un corpus annoté. Cependant il existe beaucoup de trigrammes possibles, de plusieurs dizaines de milliers à plusieurs millions suivant le nombre de parties du discours considérées, et surtout leur distribution, comme beaucoup de choses dans la langue, n'a rien d'uniforme. De nombreux trigrammes peuvent rester très peu, voire pas, observés dans le corpus d'entraînement. Il est donc nécessaire de lisser

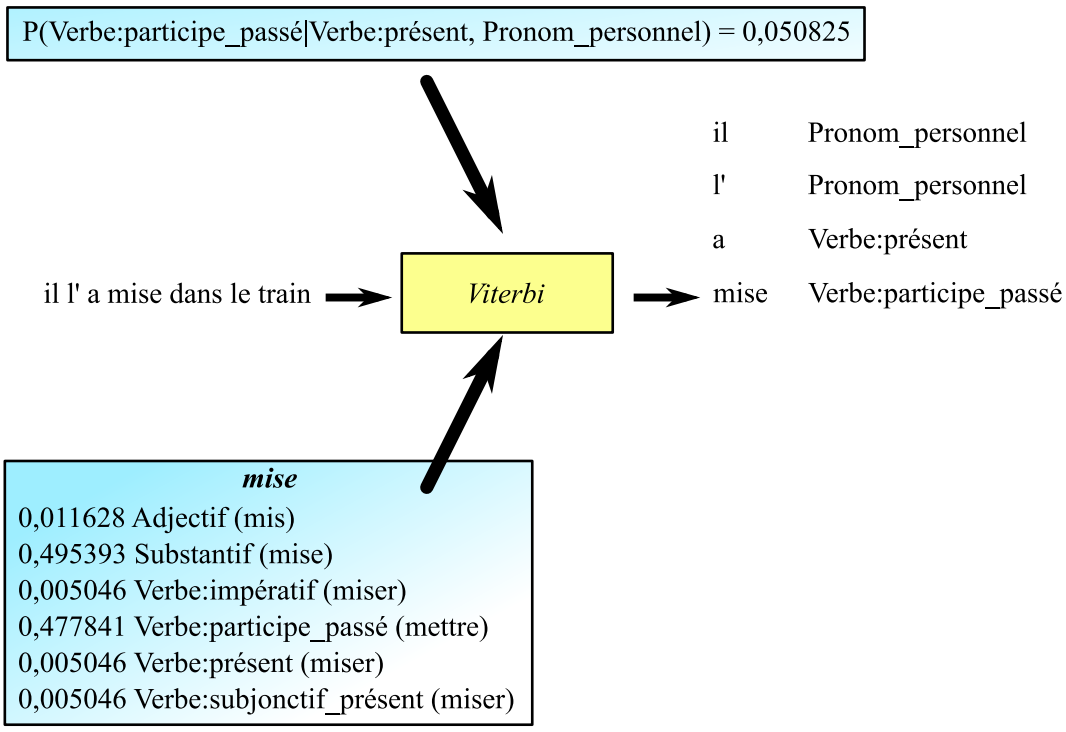


FIG. 2.3 – Fonctionnement général du TreeTagger. Un modèle de mots (en bas) est combiné avec un modèle de successions (en haut) via l'algorithme de Viterbi pour obtenir des annotations en parties du discours.

les probabilités pour obtenir une meilleure généralisation du modèle. [Chen & Goodman 1998] reste la référence sur le sujet. Les auteurs du TreeTagger ont cependant décidé d'utiliser une approche alternative en structurant l'espace des contextes possibles (i.e. les deux parties du discours précédents) via un arbre de décision construit automatiquement sur des considérations d'information mutuelle. Le lecteur intéressé est invité à se référer à [Schmid 1994].

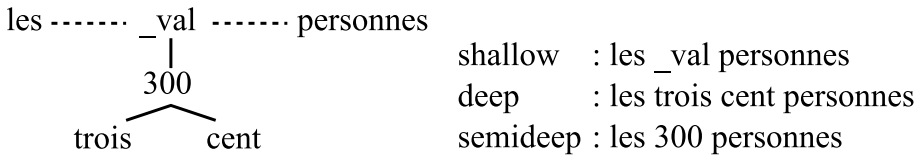


FIG. 2.4 – Les trois modes d'interprétation de la structure proposés pour l'annotation en parties du discours. *shallow* ne prend les labels qu'aux racines, *deep* prend les feuilles, et *semideep* cherche les premiers mots en descendant, n'étant pas un mot ce qui commence par _.

Intégrer un tel système dans notre moteur pose la question de l'interprétation de la représentation. Il faut décider des mots sur lesquels les parties du discours vont être déterminées. Nous avons décidé de

proposer à l'utilisateur trois possibilités : *shallow*, qui ne prend en compte que les mots de haut niveau (les racines des mini-arbres), *deep* qui ne prend que ceux de bas niveau (les feuilles) et *semideep*, un mode hybride qui cherche les mots les plus haut placés dans la structure. Un mot est défini comme toute alternative ne commençant pas par _. La figure 2.4 donne un exemple de ces trois modes. Les parties du discours sont alors ajoutées comme alternatives sur les mots. L'expert linguiste peut ainsi choisir la meilleure méthode d'application en fonction de l'avancée de l'analyse au moment où il utilise ce modèle statistique et du type de données sur lesquelles il a été entraîné.

2.4 Transformations algorithmiques

Le dernier type de transformations qu'il nous reste à couvrir sont les transformations *algorithmiques*. C'est un peu un usage abusif du terme, étant donné que toutes les transformations que nous avons présentées jusque là ont bien évidemment une composante algorithmique, mais nous regroupons dans cette catégorie toutes celles qui sont simples et s'écrivent naturellement sous forme de programme. Elles prendraient bien souvent la forme d'un court script PERL dans une chaîne de traitement traditionnelle. Nous les avons ajoutées au moteur au fur et à mesure de nos besoins, parfois en les testant d'abord via un programme extérieur.

La première de ces transformations est l'utilisation de dictionnaires associant un ou des tags à des mots pour enrichir la représentation. Cette transformation ajoute à chacun des mots de la représentation les tags associés en alternatives. Nous l'utilisons entre autres pour intégrer le dictionnaire DELAS [Courtois 1990], qui contient pour plus de 500 000 formes du français les classes grammaticales, d'inflexion et sémantiques possibles associées.

Un autre tranformation est un *filtrage*. Il permet de supprimer systématiquement les tags ou même les mots contenus dans une liste de la représentation, remontant les descendants au besoin quand toutes les alternatives d'un nœud sont supprimées. Nous nous en servons en particulier pour supprimer les tags morphosyntaxiques en fin de chaîne pour rendre la sortie plus lisible pour un humain.

Une troisième transformation détecte les ordinaux numériques, comme *1er*, *2nd*, etc, et supprime l'extension en la remplaçant par un tiret, donnant *1-*, *2-*, permettant ainsi d'utiliser les catégories *%ordinal* et *%number*.

Enfin deux transformations sont complémentaires des règles. La première reconstruit des mots acronymes à partir de leurs lettres individuelles après que des règles les aient regroupées sous un tag *@acro*. Ainsi *<@acro> S. N. C. F. </@acro>* devient *<SNCF> S. N. C. F </SNCF>*. La forme d'origine est ainsi conservée mais il devient possible de manipuler l'acronyme reconstitué facilement.

La dernière transformation reconstruit les nombres cardinaux à partir de leurs mots à partir d'annotations d'un ensemble de règles appropriées qui regroupent les mots d'un même nombre et notent les valeurs numériques de chacun. Par exemple *deux mille neuf* est annoté par les règles en *<@Num-*

ber> <@2U> deux </@2U> <@1K> mille </@1K> <@9U> neuf </@9U> </@Number> qui est ensuite simplifié par la transformation en *<2009> deux mille neuf </2009>*.

Aucune de ces transformations ne fait d'opération complexe ou même très originale, mais il est pratique pour le créateur d'un système d'analyse de pouvoir les intégrer explicitement dans sa chaîne de traitement.

2.5 Gestion des entrées/sorties

Définir une représentation interne et être capable de la transformer sont deux points essentiels pour la construction de notre moteur. Cependant un point reste : pouvoir lire des documents en les mettant au format de la représentation et inversement écrire l'état de la représentation sous la forme d'un nouveau document. Nous regroupons cela sous le terme général d'entrées/sorties.

Deux positions opposées existent en général. Soit un petit nombre de formats est imposé par l'outil, soit, ce que préfèrent les frameworks, des composants spécifiques s'occupent du problème. Dans le cas d'UIMA [Ferrucci & Lally 2004] par exemple les composants de début de chaîne qui créent la forme initiale de la représentation sont nommés *lecteurs* et ceux de fin de chaîne qui recréent des documents ou rangent les résultats dans une base de données sont nommés *consommateurs*.

Étant donné nos besoins, il nous a semblé plus approprié de suivre l'approche simple d'un ensemble de formats fixes. Nous proposons ainsi comme formats d'entrée du texte simple qui sera découpé en mots et mis dans la représentation dans un simple vecteur. Nous entendons par mot une suite de caractères entre deux espaces. Nous considérons que le problème général de la *normalisation* [Adda, et al. 1997], qui couvre séparation des mots des ponctuations, correction de la casse, séparation de certains mots composés, des apostrophes, etc, est un problème à part entière qui sort du cadre du moteur d'analyse.

Mais une idée directrice sous-jacente à notre choix de représentation est la possibilité de l'externaliser dans un format texte aussi lisible que possible. Nous avons donc créé deux formats d'externalisation. Un, nommé *xml*, est moins lisible mais plus robuste : les groupes d'alternatives sont marqués par <a> ... et les dérivations par ... . La fin du document est marquée par <d>. L'exemple de la figure 2.1 page 38 s'écrit alors :

```
_GN <b> <a> le _∼Det </a> <a> garçon _∼Nom </a> </b> _GV <b> <a> mange
_∼Verbe </a> _GN <b> <a> les _∼Det </a> <a> bonbons _∼Nom </a> </b> </b>
```

Un format alternatif, moins robuste mais plus lisible, est le format *xtag*, où les dérivations prennent la forme de tags XML et les alternatives sont séparées par des | :

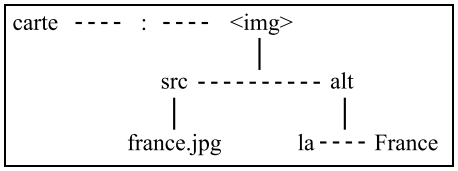
```
<_GN> le|_∼Det garçon|_∼Nom </_GN> <_GV> mange|_∼Verbe <_GN>
les|_∼Det bonbons|_∼Nom </_GN> </_GV>
```

En pratique nous utilisons le format xml pour communiquer avec des transformations externes et le format xtag pour les documents annotés finaux. Ces deux formats sont bien évidemment utilisables aussi en entrée du système, permettant de ré-analyser des documents déjà analysés.

Enfin il est intéressant de noter que la représentation est en pratique assez flexible et permet des réalisations un peu en dehors de l'analyse de texte. Par exemple nous avons conçu un filtre d'entrée nous permettant de traiter de manière robuste les documents HTML et XML. Ce filtre voit ces documents comme des flux de tags, ouvrants ou fermants, et de texte, dans une approche similaire à SAX [Press 2002]. Pour un extrait de document tel que :

```
carte :<img src="france.jpg" alt="la France">
```

la représentation sera :



les opérateurs de descente dans les arbres permettent alors de récuperer les attributs des tags et leurs valeurs. Cet aspect nous a souvent aidés à récupérer efficacement des informations utiles de documents xml, html ou même openoffice plus ou moins corrects.

2.6 Construction d'un analyseur complet

Utilisant les transformations que nous avons présentées, un expert peut alors construire un système d'analyse complet en combinant une série de transformations les unes à la suite des autres. La représentation de l'état de l'analyse étant prévue pour être externalisable, nos premières expériences ont pris la forme d'un simple *pipe* liant les différentes invocations des programmes implémentant les transformations. Une librairie commune gère les entrées/sorties vers et depuis la représentation interne. La figure 2.5 donne un exemple d'un tel script. Cette approche, bien que simple et très flexible, a plusieurs inconvénients :

– Pour un nombre de passes assez grand (autour de la trentaine), plus de 50% du temps d'analyse passait en entrées/sorties.
– Une fois construit, un système d'analyse était peu pratique à intégrer dans une application.

```
# !/bin/sh
h="/people/.../system"

r="$h/regles"
p="$h/par"

wmatch -Ftxtlines -Txml -R $r/sp-num.wm | \
wnumbers -Fxml -Txml | \
wtagger -Fxml -Txml -s $p/spanish-par-linux-3.1.bin $p/sp-mapping.txt | \
wmatch -Fxml -Txml -S $r/type-question.wm | \
wmatch -Fxml -Txml -S $r/sp-date-time.wm | \
wmatch -Fxml -Txml -S $r/org.wm | \
wmatch -Fxml -Txtag -S $r/loc.wm
```

FIG. 2.5 – Exemple de système d'analyse sous forme de script. *wmatch* correspond à la transformation par règles, *wnumbers* à la transformation algorithmique de reconstitution des nombres, *wtagger* à la transformation statistique applicant les modèles du TreeTagger. Les paramètres -F et -T gèrent les formats d'entrée/sortie, les autres paramètres sélectionnent les méthodes d'application.

– En cas de crash causé par un bug du moteur, il n'était pas toujours facile d'isoler laquelle des étapes posait problème.

Elle avait cependant l'avantage d'être naturellement parallèle, chaque transformation pouvant potentiellement s'exécuter sur un processeur différent des autres. Cependant le temps pris par chaque transformation peut être très déséquilibré et le système se retrouve en pratique limité par la vitesse de la plus lente des transformations.

Pour remédier à ces différents problèmes, nous avons décidé d'intégrer la notion de système complet dans le moteur. Un mini-langage s'appuyant sur *lua* [Ierusalimschy, et al. 1996] permet de décrire les différentes étapes de l'analyse ainsi que les chemins d'accès des différents fichiers externes utiles. Un exemple est donné figure 2.6. Cette différence peut paraître insignifiante, mais elle se révèle à l'usage très importante par ce qu'elle permet. Tout d'abord, combinée à une organisation du moteur sous forme de bibliothèque, elle permet l'intégration d'un système d'analyse dans une application l'utilisant, comme un système Question-Réponse ou un système de dialogue, sans avoir à se préoccuper de la structure interne de l'analyse. Un seul fichier décrivant l'analyse est visible de l'application l'utilisant. Elle permet aussi d'avoir une notion de *compilation*, où ce fichier décrivant l'analyse est passé à un programme qui fournit à partir de là un fichier binaire contenant la totalité des informations utiles ainsi que les résultats de tous les précalculs dont le moteur a besoin. Ce fichier permet ensuite de charger l'analyse en un temps et une utilisation mémoire minimale, ce qui est très commode pour toutes les applications qui sont simples utilisatrices de l'analyse comme par exemple les systèmes Question-Réponse présentés dans les parties suivantes. Enfin il reste possible de paralléliser le résultat non plus au niveau des étapes mais au niveau des données, en distribuant les phrases ou les documents aux différents processeurs disponibles, avec pour effet qu'aucun processeur ne va faire

```
# !/people/.../bin/wmatch
h = "/people/.../system"

paths.regles = h.."/regles"
paths.treetagger = h.."/par"

r = input()
r = match_left_recurse(r, "sp-num.wm")
r = numbers(r)
r = treetagger_semideep(r, "spanish-par-linux-3.1.bin", "sp-mapping.txt")
r = match_global_replace(r, "type-question.wm")
r = match_global_replace(r, "sp-date-time.wm")
r = match_global_replace(r, "org.wm")
r = match_global_replace(r, "loc.wm")
output(r)
```

FIG. 2.6 – Exemple de système d'analyse sous forme intégrée. Les fonctions *match* correspondent à la transformation par règles, *numbers* à la transformation algorithmique de reconstitution des nombres, *treetagger* à la transformation statistique applicant les modèles du TreeTagger. Le nom des fonctions indique la méthode d'application, les formats d'entrée/sortie ne sont pas sélectionés à ce niveau.

attendre les autres.

Nous sommes arrivés à la conclusion que proposer la possibilité d'avoir une version *packagée* de l'analyse était bien plus qu'un simple plus au niveau des performances ou du débogage. Elle permettait une *abstraction* de l'analyse. Cette abstraction facilite l'utilisation boite-noire d'une analyse et ainsi la collaboration entre plusieurs personnes travaillant sur des aspects différents d'un système. Elle permet aussi la création d'outils de diagnostic capables de répondre à des questions telles que *qu'est-ce qui dans l'analyse a décidé que cette instance du mot avocat était un fruit ?* en permettant de reproduire et de contrôler la totalité des changements de la représentation effectués par les transformations.

Chapitre 3

Aspects algorithmiques

Après avoir exposé la spécification de notre moteur d'analyse nous abordons ici les problèmes d'implémentation. En particulier certains aspects spécifiques ont un effet primordial pour sa qualité du résultat : l'encodage de la représentation et en particulier la représentation des mots et des catégories, et les difficultés liées au moteur de règles.

3.1 Encodage de la représentation

Un premier aspect fondamental concerne l'encodage de la représentation de l'état de l'analyse. La structure elle-même ne pose pas de problème particulier, il ne s'agit que d'un vecteur d'objets nœuds pouvant récursivement contenir un vecteur de nœuds dérivés. L'encodage des mots est plus critique. En effet une grande partie des opérations effectuées par les transformations consistent à comparer si un mot de la représentation est identique à un mot particulier, venant des règles ou des modèles, ou encore chercher si un mot fait partie d'une table et si oui accéder aux informations associées. Comparer sans cesse des chaînes de caractères n'est pas très efficace et il est plus judicieux d'associer à chaque mot un identifiant numérique.

3.1.1 Attribution d'identifiants numériques aux mots

Deux méthodes principales existent pour associer un identifiant à un objet : recenser l'ensemble des objets possibles et leur associer à chacun un numéro, ou construire algorithmiquement un nombre à partir de la valeur de l'objet en espérant que deux valeurs différentes ne donneront pas en pratique le même numéro. La seconde méthode est souvent qualifiée de *hachage*.

A priori il n'est pas possible de prévoir l'ensemble des mots possibles, d'autant plus quand la définition de mot est *ensemble de caractères entre deux espaces*. Cela pousserait donc vers la solution de hachage. Cependant elle n'est pas sans inconvénients. En effet, pour réduire la probabilité de collision à un niveau acceptable la valeur maximale produite par une fonction de hachage doit être au minimum le carré du nombre d'objets différents attendus. Cela implique des valeurs de 64 bits pour couvrir les mots de la langue, 32 bits étant insuffisant, plaçant la limite autour de 65 000 mots, et les tailles intermédiaires étant sous-optimales dans les architectures modernes. Une valeur de 64 bits est efficace pour les comparaisons d'égalité mais elle est potentiellement trop grande pour servir d'index dans un tableau, obligeant à recourir à des structures plus lentes telles les tables de hash pour trouver les informations associées à un mot.

Mais en pratique une variante du recensement est possible : recenser l'ensemble des mots présents dans les règles, modèles et dictionnaires utilisés dans les transformations et associer à chacun un numéro. En pratique nos analyses les plus importantes ont un vocabulaire d'environ 1,5 million de mots. Les inévitables mots de documents qui n'entrent pas dans ce vocabulaire sont regroupés sur un numéro spécial et un emplacement est prévu dans la structure nœud pour mettre le texte effectif. Cette méthode est en pratique très efficace car un mot ayant cet identifiant *hors vocabulaire* ne peut pas être dans les règles ou les tables, garantissant la validité de la comparaison des identifiants pour comparer l'identité des mots à partir du moment où un des mots comparés vient des règles ou modèles. De même il ne peut avoir d'informations associées pour des raisons identiques.

3.1.2 Gestion des catégories

Un autre aspect important du contenu des nœuds est la gestion des catégories. Ces classes de mots, décrites à la section 2.2, contiennent des types prédéfinis tels que *tout en majuscules*, *tout en minuscules*, *commençant par une majuscule*, mais aussi des intervalles de nombres, cardinaux ou ordinaux. Tester ces classes à chaque fois qu'il y en a besoin serait inefficace, et il est plus performant de le faire au moment de la lecture du document et quand le contenu de nœuds est modifié par une transformation. À chaque catégorie simple est associé un numéro et le nœud contient la liste de numéros de catégories qui lui sont associés. Seuls les intervalles numériques pourraient a priori poser problème. Mais en pratique il est possible de recenser l'ensemble des intervalles utilisés, faire l'inventaire de leurs bornes et s'en servir pour décomposer l'ensemble des entiers en segments disjoints tels que chaque intervalle soit exactement composé d'un ensemble fini de segments. Les segments sont alors considérés comme des catégories élémentaires et numérotés. Les références aux intervalles dans les règles sont remplacées par une union de références aux segments les composant. La classification d'un nombre ne demande alors qu'à rechercher dans quel segment il est contenu, ce qui peut être fait efficacement de manière dichotomique.

3.2 Difficultés liées au moteur de règles

La représentation décidée, les transformations statistiques et algorithmiques que nous avons présentées ne posent pas de difficultés d'implémentation particulières. Le moteur de règles possède des particularités intéressantes à étudier au niveau du matching. La performance globale du système dépend fortement de sa capacité à trouver rapidement quelles expressions régulières étendues sont applicables à un endroit spécifique dans la représentation. La méthode traditionnelle pour prendre en compte des expressions régulières est de les transformer en automates finis déterministes, assurant ainsi un temps de recherche linéaire en fonction de la taille du document. Cependant les extensions rendent cet exercice particulièrement difficile : la présence de bornes sur les nombres de répétitions peut déclencher des explosions combinatoires sur le nombre d'états dans l'automate. Les automates ne sont pas capables de tenir compte des répétitions *shy* ou *greedy*. Ils ne peuvent pas non plus indiquer quelles zones sont couvertes par les parenthèses de substitution. Et enfin les *lookaheads* négatifs ou arrières sont totalement en dehors de leurs capacités. Cette transformation n'est donc pas utilisable en l'état.

3.2.1 Matching d'expression régulières par interprétation de patterns

Nous avons pris l'option de considérer les expressions régulières comme étant un *langage de patterns* pour lequel nous écrivons un *interpréteur récursif*. Pour chaque type de nœud que l'on peut trouver dans l'arbre syntaxique d'une de nos expressions, par exemple matching de mot simple, matching de classe, concaténation, alternative, répétition, lookahead, il est possible d'associer deux opérateurs *match* et *next*. *match* doit, étant donné une position dans la représentation, indiquer si un premier matching est possible à cet endroit et la zone correspondante. *next* indique ensuite à chaque appel si un matching supplémentaire est possible. Ces deux opérateurs peuvent être écrits en fonction de contenu de la représentation pour les nœuds feuilles et en fonction de ces mêmes opérateurs sur le ou les descendants pour les autres, d'où la récursivité. En cas d'échec un *backtrace* se produit où les appels récursifs sont remontés jusqu'à ce qu'un matching alternatif soit trouvé ou que le matching global soit abandonné. Un intérêt supplémentaire de cette approche est que les opérateurs sont *retournables*, permettant de faire un matching de droite à gauche dans le document en partant de la fin de l'expression régulière, ce qui rend immédiate l'implémentation des lookaheads arrières mais aussi des recherches récursives partant de la droite.

Une telle approche est bien évidemment plus lente qu'un automate. Elle peut cependant devenir très performante si l'on arrive à limiter deux facteurs : d'une part la quantité de travail par nœud (*temps passé par instruction*), et d'autre part le nombre d'essais se révélant a posteriori inutiles (*nombre d'instructions exécutées*).

3.2.2 Limitation de la quantité de travail par nœud et gestion de la mémoire

Limiter la quantité de travail par nœud est un problème très lié à l'implémentation. La transformation des mots et catégories en nombres aide déjà, permettant d'effectuer les matchings simples de mots par comparaison d'entiers et les matchings de classe par lookup dans un vecteur de bits de la taille du vocabulaire. Une difficulté reste cependant : la gestion de la mémoire. En effet l'opérateur *next* nécessite d'avoir assez d'informations stockées pour pouvoir calculer le matching suivant. Or la solution évidente, stocker ces informations dans le nœud, n'est pas aussi simple qu'il y paraît. En présence de macros, plusieurs matchings peuvent être actifs pour le même nœud en même temps. Dupliquer les nœuds, ou en d'autres termes instancier les macros, donne lieu à une explosion combinatoire. Stocker plusieurs informations par nœud pose ses propres difficultés. Il est en effet possible de les organiser en pile, la recherche par backoff assurant qu'un matching sera terminé avant qu'un précédent soit continué. Cela garantit qu'un appel *next* agit sur l'état situé en haut de la pile. Cependant vider correctement les piles après un matching réussi complet d'une expression est complexe et coûteux. En effet il n'est pas possible de simplement supprimer le contenu des piles. Les lookaheads et les descentes dans les arbres sont en pratique des matchings complets de sous-expressions où seul le premier nous intéresse et qui donc nécessitent une passe de vidage après usage. Mais ces sous-expressions peuvent, via là encore les macros, avoir des nœuds communs avec l'expression principale qui est elle non-terminée. Il ne faut donc enlever des piles que les parties du haut correspondant à la sous-expression. Cela se révèle en pratique assez coûteux.

Cependant une approche alternative permet de régler ce problème. La clé est de noter qu'à l'exception des répétitions sans borne supérieure, chaque type de nœud a besoin d'une place de taille fixe pour son stockage, taille calculable au moment de la lecture des règles, à laquelle il faut ajouter la place nécessaire pour ses descendants. Même le cas des répétitions illimitées peut rentrer dans ce cadre en ayant comme information de taille fixe un pointeur vers une zone mémoire allouée dynamiquement au besoin. Chaque nœud possède ainsi une zone mémoire avec laquelle travailler. Il passe à ses descendants leur propre zone qui est une sous-partie de la sienne. Cette structuration en zones mémoire incluses les unes dans les autres récursivement sépare naturellement les multiples matchings pouvant s'appliquer sur le même nœud via les macros. Il n'y a alors plus besoin d'action spécifique après un matching complet réussi. Nous avons constaté que même avec nos règles les plus complexes cette zone mémoire ne fait pas plus que quelques kilo-octets, rendant cette solution très performante et l'utilisation mémoire du système après initialisation minimale.

3.2.3 Limitation du nombre de tests inutiles

Limiter le nombre d'essais inutiles se fait d'abord au niveau des règles elles-mêmes, et donc de ce qu'a écrit l'expert linguiste. En effet, comme dans tout langage de programmation, des modifications subtiles peuvent avoir de gros effets sur le nombre de tests à effectuer. Il est cependant possible de l'aider avec un outil de profiling lui indiquant quelles règles prennent le plus de temps, et même dans ces règles quelles macros sont les plus coûteuses.

Le moteur lui-même peut aussi agir sur deux points pour limiter le nombre de tentatives de matching inutiles. Le premier point est un filtrage des règles. Il est possible d'obtenir un sur-ensemble de l'ensemble des mots ou des catégories pouvant apparaître en première position dans toutes les suites de mots acceptées par une règle donnée. L'estimation de l'ensemble ne peut être exacte à cause de l'existence des lookaheads arrières. Ces sur-ensembles calculés permettent alors de construire une table permettant de savoir immédiatement pour un emplacement donné dans un document quelles règles peuvent potentiellement s'appliquer.

Le second point concerne les alternatives. Une part significative de leur utilisation concerne la reconnaissance de listes d'expressions multi-mots, comme des noms de villes, de départements, de pays, pouvant avoir des centaines de milliers d'entrées. Essayer les alternatives une par une est dans ce genre de cas très inefficace. Il est donc intéressant d'utiliser une approche de filtrage similaire à celle utilisée sur les règles. Filtrer uniquement sur le premier mot possible s'est cependant révélé insuffisant, et nous filtrons actuellement sur un arbre de préfixes possibles. Nous pensons généraliser cela en une version limitée de la transformation en automates. Appliquer un arbre de préfixes ou un automate a un coût similaire, mais la meilleure couverture de variantes d'un automate permet d'obtenir un meilleur résultat au niveau du filtrage. N'utiliser les automates que pour un tel filtrage local, qui n'a pas besoin d'être parfait, permet d'éviter les difficultés dues aux extensions proposées. L'interprétation des patterns est toujours effectuée et permet de prendre en compte les lookaheads, répétitions limitées et autres problèmes. Les automates permettent de filtrer efficacement les cas où les expressions multi-mots ont un fort taux de mots optionnels ou de variantes possibles dont la combinatoire est explosive pour un filtrage par préfixes.

Chapitre 4

Évaluation

Évaluer un *moteur* d'analyse tel que celui que nous proposons n'est pas une tâche aisée. En effet on ne peut, comme pour un *système* d'analyse, annoter un corpus de test avec les informations que l'on veut obtenir et mesurer la qualité de la sortie. Un moteur d'analyse, et en particulier un mettant en avant l'écriture de règles, est similaire à un langage de programmation : le résultat est en grande partie dépendant de l'ensemble de règles.

Se pose donc la question de *sur quels critères pouvons-nous évaluer un moteur ?*. L'analogie avec un langage de programmation est utile : un moteur d'analyse est performant si, dans le cadre où il a été conçu, il aide effectivement l'expert linguiste en charge de l'analyse (le programmeur) à obtenir les résultats recherchés. Une grande partie de l'évaluation prend donc la forme de *cas d'usage*. Nous en présentons trois, qui donneront une idée de ce que le moteur nous a permis de construire :

– L'analyse pour Ritel, un système interactif multimodal de recherche d'informations en domaine ouvert en français.
– L'adaptation de la sous-partie spécifique Question-Réponse de l'analyse de Ritel à l'espagnol et à l'anglais.
– L'exploration de corpus.

Cependant, même pour un langage de programmation, mesurer la vitesse d'exécution des programmes écrits est une information pertinente. Nous présentons donc un certain nombre de mesures que nous avons effectuées sur l'analyseur développé pour Ritel, qui est le plus complet que nous ayons développé.

4.1 Ritel : un système interactif de recherche d'informations en français

4.1.1 Une analyse multiniveaux unifiée

Ritel est un projet existant depuis 2004 au LIMSI [Ritel 2007 ; Galibert, et al. 2005]. Son but est de faire progresser la recherche dans le domaine de l'interaction homme-machine en s'attaquant à une tâche difficile : la recherche interactive d'informations en domaine ouvert. Les travaux présentés dans ce document, moteur d'analyse et système Question-Réponse, visent à apporter des solutions efficaces dans cette direction.

Les besoins en analyse de langue pour un projet tel que Ritel poursuivent trois buts :
– Extraire les informations nécessaires pour la recherche d'informations.
– Extraire les informations nécessaires pour la gestion du dialogue.
– Détecter la thématique de la demande pour pouvoir rebondir dessus si besoin est.
Ces besoins se recoupent partiellement, et nous avons choisi de détecter cinq types d'informations différents, qui font l'objet des prochaines sections. Il est important de noter que le développement des grammaires a été fait par des linguistes et non par nous, mais que l'ensemble des travaux (moteur, grammaires) ont été conduits en collaboration étroite, impliquant de nombreuses discussions et échanges de vues.

4.1.2 Les entités nommées, étendues et spécifiques

Les entités nommées (NE, pour Named Entity), désignent classiquement des noms de lieux, de personnes et d'organisations. Les entités numériques (souvent associées aux NEs) représentent des dates ou des unités de mesure, en particulier monétaires. Ces entités nommées sont perçues comme des éléments majeurs pour l'analyse de texte et documents, et la recherche d'information en particulier. Les conférences MUC [Kaufman 1998] ont mis en avant plusieurs tâches génériques dont l'analyse en entités nommées. Cette analyse recouvre d'une part la détection des entités nommées (les passages contenant l'entité nommée et ses frontières) et d'autre part le typage de cette entité. Ce typage est effectué d'après des ontologies définies préalablement. La définition de base la plus utilisée est héritée des conférences MUC. Elle comprend trois catégories : les expressions de noms propres (personne, lieu, organisation), les expressions temporelles (date, heure) et les expressions numériques (les valeurs monétaires et les pourcentages). La hiérarchie définie est présentée dans [Grishman 1995]. Ces définitions sont parfois étendues à l'intérieur d'une catégorie, comme les organisations, afin d'en affiner le contour référentiel, par exemple les définitions adoptées dans le cadre de la campagne d'évaluation ACE [ACE 2000]. Afin de couvrir de nouveaux besoins ou de nouvelles tâches, ces définitions et hiérarchies sont étendues. Par exemple Sekine propose jusqu'à 200 types [Sekine 2004].

La couverture de ces entités nous a paru insuffisante. En plus des entités normales, dont nous avons

plus d'une centaine de types, nous avons ajouté des entités nommées *non précises*, couvrant une classe d'objets plutôt qu'un instance particulière. Nous avons rajouté des types qui ne sont pas traditionnellement considérés comme des entités nommées comme les fonctions, les titres, les couleurs. Et enfin nous avons permis une *hiérarchisation* de l'annotation, des entités pouvant se contenir les unes dans les autres, ainsi que la possibilité d'attribuer plusieurs types de spécificité différente à la même entité, parfois en relation d'hyperonymie, nous parlons alors de superclasse. Le tableau 4.1 donne des exemples de cet ensemble d'entités.

Entités nommées	<_org> NIST </> <_eve> festival de Cannes de 2006 </> qui a dit <_cit> veni vidi vici </>
Entités non précises	<_Eve> festival de Cannes </> le <_Pers> president </> a déclaré ...
Entités étendues multiniveaux	Fonctions, titres (président, professeur, évêque...) couleurs, animaux...
Superclasses	évêque → fonction religieuse → fonction
Multi-type non hypéronyme	<_loc> <_pays> France </> </>
Hiérarchiques	<_eve> <_Eve> festival de <_ville> Cannes </> </> de <_date> <_annee> 2006 </> </> </>

TAB. 4.1 – Variation sur le thème des entités nommées normales et étendues

4.1.3 Les mots de question

Les mots de question tels que *Qui* ou *Quoi* ou encore *Combien* sont bien évidemment indispensables pour interpréter les questions de l'utilisateur. Il convient donc de les détecter et de les typer aussi précisément que possible. Il est en effet plus intéressant d'interpréter la locution complète *quel est le nom de* plutôt qu'un simple *quel* pour définir au mieux le type d'informations recherchées. Là encore une structuration hiérarchique est utilisée, le mot *quel* seul étant d'abord annoté puis ensuite la locution complète.

4.1.4 Les marqueurs thématiques

Les marqueurs thématiques sont ceux qui permettent de détecter le sujet de la discussion, comme le mot littérature dans *je m'intéresse à la littérature*. Ils sont intéressants pour permettre au système de répondre à l'utilisateur dans les cas où la question ne contient pas assez d'informations pour permettre une recherche pertinente. Pour notre exemple, la réponse pourrait être *la littérature est un sujet passionant, que voulez-vous savoir ?*, le système confirmant ainsi sa compréhension du thème à l'utilisateur.

4.1.5 Les marqueurs dialogiques

Pour permettre un échange naturel, nous avons besoin de détecter et d'interpréter les éléments de gestion de l'interaction. Ils prennent plusieurs formes, des simples commandes directes *au revoir*, *pouvez-vous répéter*, aux réponses aux questions *oui*, *non* en passant par les corrections *non, pas* X, *je voulais dire* Y. Ils doivent donc être intégrés dans l'analyse.

4.1.6 Les chunks linguistiques

Les mots non couverts par les définitions précédentes possèdent potentiellement une information utile, ce qui nécessite donc de les annoter. Ils sont regroupés en blocs les plus longs possibles de catégorie comparable (groupe nominal, groupe verbal, etc.) typés avec des catégories syntaxiques grossières (nom composé, verbe, adjectif, etc.). Ces goupes sont des éléments de bas niveau et ne sont donc pas récursifs. Comme dit précédemment, cette définition se rapproche fortement de celle donnée par [Abney 1995] pour la notion de *chunk*.

4.1.7 Quelques résultats préliminaires

Nous avons effectué une évaluation interne sur les entités nommées et étendues. Pour cela, différents corpus, décrits dans le tableau 4.2, ont été utilisés :
– questions orales : énoncés utilisateurs collectés avec la première plateforme Ritel, transcrits manuellement ;
– questions écrites : questions de l'évaluation CLEF'04 ;
– émissions d'information radio- et télé-diffusées : documentaires et informations en français transcrits manuellement ;
– journaux : Le Monde et ATS 1994-1995.
Ces corpus ont été annotés manuellement en fonction des définitions des entités que nous traitons. Ils ont servi de référence pour cette évaluation.

Catégorie	# doc.	# mots	# Entités
questions orales	840	10k	1 102
questions écrites	200	1,5k	331
émissions d'informations	7 887	88k	5 898
journaux	1 000	22k	2 485

TAB. 4.2 – Caractéristiques des corpus de test

Les entités évaluées étaient les suivantes : *lieux*, *organisations*, *personnes*, *dates*, *montants*, *valeurs*, *âges*, *couleurs*, *fonctions*, *phénomènes météorologiques*, *monnaies*, *mesures physiques*, *organisations*

gouvernementales, *évènements*, *prix* et *musées*. Les entités plus spécifiques comme *ville*, *pays*, *province* ou *fleuve* ont également été évaluées. Dans le cas d'un groupe hiérarchique seule l'entité la plus large a été évaluée.

Le tableau 4.3 présente les résultats obtenus sur les différents corpus. Ce système obtient une F-mesure allant de 0,82 sur le corpus d'émissions d'informations à 0,88 sur le corpus de questions orales. La plupart des systèmes de détection en entités nommées obtiennent une F-mesure autour de 0,9 [Poibeau 2005] sur des données journalistiques et dans le cadre d'une définition simple des entités nommées. [Favre, et al. 2005] rapportent une F-mesure entre 0,84 et 0,74 sur un corpus d'émissions d'information radio- et télé-diffusées de la campagne d'évaluation ESTER [Gravier et al. 2004]. Sur des transcriptions manuelles comme celles de notre corpus d'émissions d'information, leur meilleur système obtient une F-mesure de 0,84. Par ailleurs, [Surdeanu, et al. 2005] rapportent une F-mesure sur des corpus de type conversationnel (en anglais) de 0,75. Comparer nos résultats avec ceux-ci est relativement peu aisé car les définitions des entités diffèrent. Néanmoins, on peut probablement affirmer que sur des données journalistiques textuelles (comme notre corpus de journaux) notre système est légèrement moins bon que ceux décrits dans la littérature. En revanche, il est à hauteur de l'état de l'art pour les corpus oraux.

Mesure	Questions orales	Questions écrites	Émissions d'informations	Journaux
Precision	90,3%	85,5%	83,5%	87,1%
Rappel	86,2%	83,4%	81,3%	86,5%
F-mesure	0,882	0,845	0,824	0,868

TAB. 4.3 – Résultats pour la détection des entités nommées

Ces résultats suggèrent qu'il est possible d'écrire un système de détection d'entités nommées robuste et de bonne qualité en s'appuyant sur notre moteur.

4.2 Adaptation de l'analyseur à l'espagnol et à l'anglais

Afin de pouvoir tester entre autres les capacités multilingues de notre moteur et de notre système Question-Réponse nous avons participé à la campagne d'évaluation *Question-Answering on Speech Transcripts 2008* [Turmo, et al. 2008]. Cette campagne et les résultats de notre participation sont décrits partie III section 11.1. Cette participation a nécessité que nos collègues linguistes adaptent l'analyse présentée à l'anglais et à l'espagnol. L'adaptation a bien sûr été facilitée parce que ces deux langues sont proches du français. Toutefois nous pensons qu'elle aurait été moins simple si le moteur et le langage que nous avons mis à disposition des linguistes pour écrire les grammaires n'était pas si simple à manipuler.

Le passage du français à l'espagnol a été fait en deux temps : la recherche de parties du discours non-désambiguïsée dans le DELAS a été remplacée par une annotation désambiguïsée via la transfor-

```
%lex : University university Université université Universidad universidad ;
&prep :
        // français
        de la | des | du | "d'" |
        // espagnol
        de los | de |
        // anglais
        of the | of
;
// français
&univ-fr : %lex &prep ? ( ? : %caps* ? | %acronym | _ville) ;
// espagnol
&univ-es : %lex &prep ? ( ? : %caps* ? | %acronym | _ville) ;
// anglais
&univ-en : %lex &prep ? ( ? : %caps* ? | %acronym | _ville) |
                ( ? : %caps* ? | %acronym | _ville) %lex ;
```

FIG. 4.1 – Exemple d'adaptation interlingue d'une règle.

mation statistique utilisant les modèles du TreeTagger. De plus aux différents lexiques ont été ajoutés les équivalents en espagnol. Toutefois, seule une petite partie des lexiques a été adaptée. Pour l'anglais l'effort a été plus important, nécessitant en plus de modifier l'ordre d'application de certains contextes. La figure 4.1 illustre cette adaptation.

Cette méthode d'adaptation nous a permis d'obtenir des analyses comparables entre les trois langues, dont un exemple est donné figure 4.2. Cette adaptation même incomplète a donné des résultats tout à fait raisonnables lors de la campagne d'évaluation QAst 2008.

4.3 Exploration de corpus

Une autre utilisation possible de ce moteur est l'exploration de corpus. C'est une utilisation qui rappelle ce que propose CQP. Au Limsi, deux études ont débuté mi-2008 nécessitant l'exploitation de corpus préalablement annotés.

L'extraction de patrons et de co-occurrences est rendu possible simplement par la multiplicité des formats d'entrées et de sorties que manipule le moteur. En particulier les deux expériences que nous abordons ici bénéficient de la possibilité d'accepter en entrée le format de sortie. Ainsi des documents préalablement annotés peuvent être réannotés/traités avec de nouvelles passes.

Dans un premier cas, il s'agit d'étudier la fonction pragmatique des hésitations et de leurs équivalents

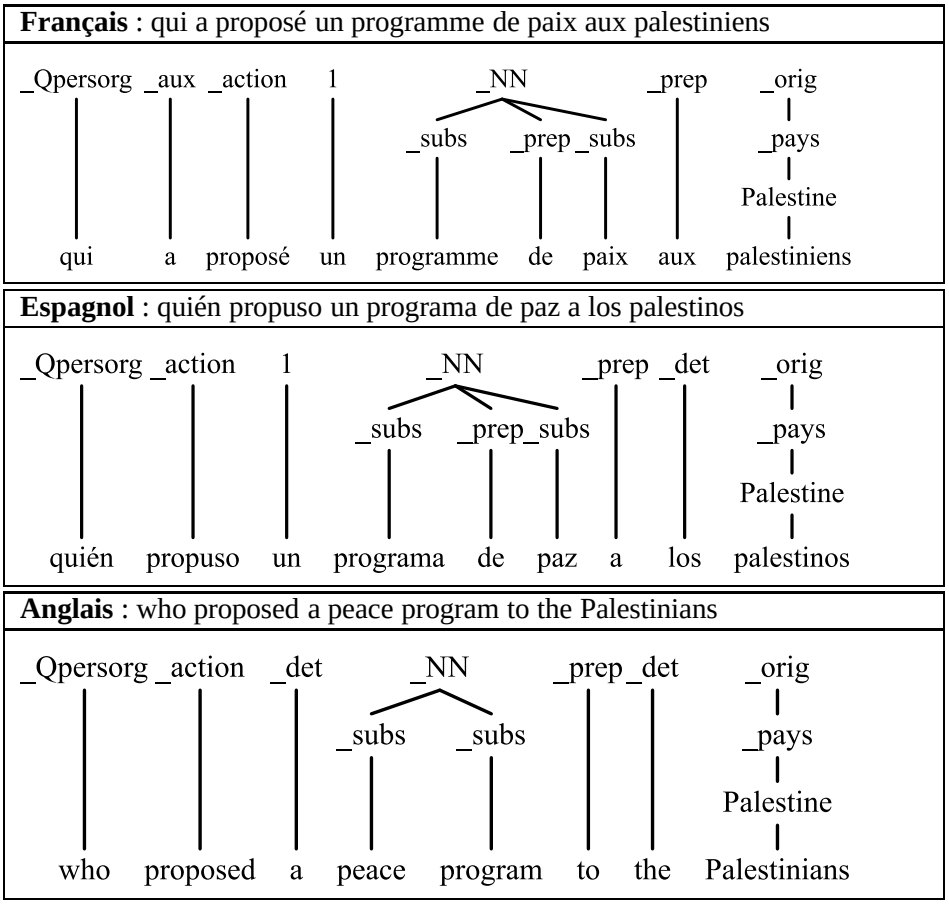


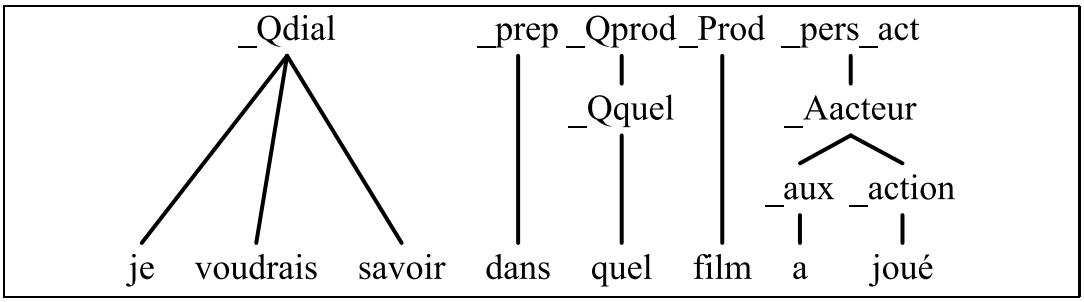
FIG. 4.2 – Résultat de l'adaptation de l'analyse à des langues autres que le français.

dans les énoncés des utilisateurs du système Ritel. Une première étape de ce type de travail est de collecter les co-occurrences impliquant justement ces éléments.

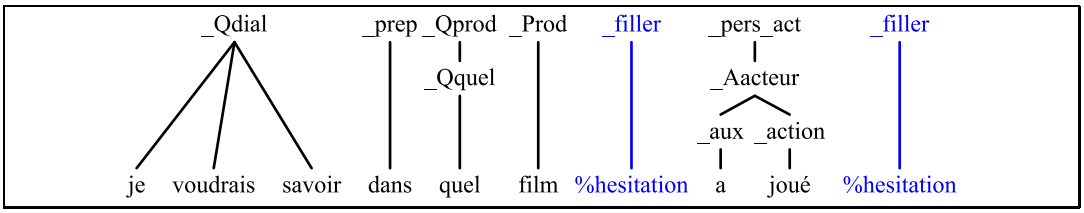
L'extraction des co-occurrences est réalisée sur le corpus annoté avec le système Ritel. Il faut dans un premier temps réinsérer puis annoter les marqueurs d'hésitations qui sont supprimés par l'analyseur standard. Ceci peut être fait par l'application d'une règle très simple telle que :

```
&filler : "%hesitation" | (< ! est) bon | "%respiration" ;
_filler : (&filler) ;
```

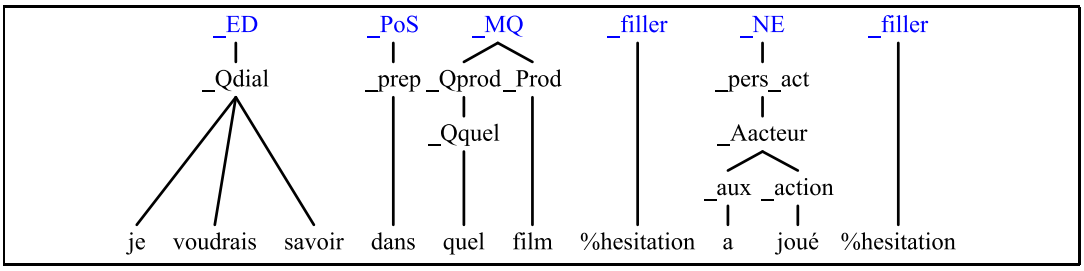
Nous partons de la phrase *je voudrais savoir dans quel film %hesitation a joué %hesitation* qui a été préalablement annotée en :



Après réinsertion des hésitations et leur annotation on obtient alors :



Il est ensuite possible de regrouper dans des catégories plus larges les différents tags, comme toutes les entités nommées et spécifiques sous la catégorie EN, toutes les entités dialogiques sous une catégorie ED etc, donnant le résultat :



Ensuite selon les besoins, il est possible d'extraire les successions à différents niveaux. Par exemple, au niveau des têtes de l'analyse on obtient :

_ED _PoS _MQ _filler _NE _filler

Cette forme a ensuite été utilisée pour mesurer statistiquement les probabilités de co-occurrence entre hésitations et entités nommées ou autres éléments des phrases. L'extraction se fait ici en partant de l'analyse standard utilisée dans Ritel mais elle pourrait se faire depuis n'importe quelle analyse respectant l'un des formats gérés par le moteur.

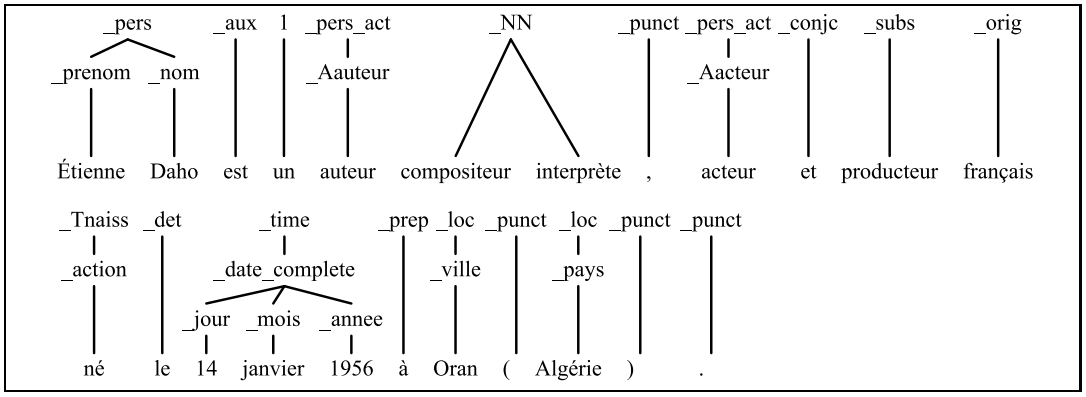
La deuxième expérience porte sur l'extraction de patrons précis d'analyse dans des données *propres* (Wikipedia par exemple) et leur intégration semi-automatique dans l'analyseur (en passe finale par exemple) afin d'améliorer la précision de la représentation des données textuelles (en particulier du Web) et donc le système de Question-Réponse.

En s'appuyant sur les informations présentes dans les infobox, il est possible de récupérer les structures dans le texte principal où ces informations apparaissent et même d'utiliser ces informations pour détecter des contextes permettant d'améliorer l'analyse. Par exemple, la phrase :

Étienne Daho est un auteur compositeur interprète, acteur et producteur français, né le 14 janvier 1956 à Oran (Algérie).

Les éléments auteur compositeur interprète, acteur et producteur français sont présents dans l'infobox de la page, sous la catégorie *Profession*.

La phrase annotée par le système d'analyse standard donne :



Les informations contenues dans l'infobox permettent de détecter automatiquement que *compositeur interprète* n'est pas connu du système en tant que profession (_pers_act) mais est reconnu en tant

que mot composé. Extraire un peu de contexte permet alors de générer une règle de correction qui va rajouter une annotation *_pers_act* au dessus du *_NN*.

```
_pers_act : _pers _aux 1 _pers_act (_NN) _punct _pers_act ;
```

De plus utiliser ces informations et la possibilité offerte par le moteur de réannoter des arbres ou des nœuds permet aussi de préciser certaines annotations. Par exemple, la règle suivante permet de réannoter un mois d'une date de naissance en mois de naissance :

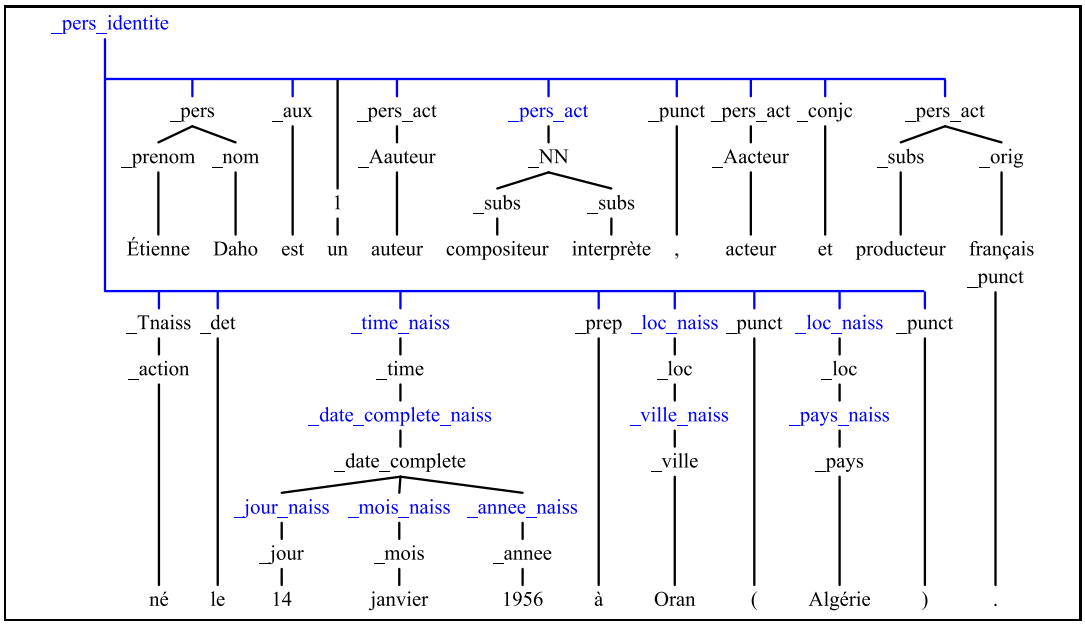
```
%replace_tree(%1, _mois_naiss « %1 ») : _time_naiss « _time « _date_complete « _jour (_mois) _annee » » » ;
```

Collecter automatiquement des patrons de ce type et les conserver sous forme de règles permet d'améliorer la précision de l'analyse et en particulier de détecter des relations entre différents éléments d'une phrase et de les regrouper sous un même nœud. Par exemple, à partir d'un ensemble de patrons constitués à partir de différentes informations comme celles que nous venons de voir, il est possible de constituer une règle du type :

```
_pers_identité : (_pers _aux 1 _pers_act _pers_act _punct _pers_act _conjc _pers_act _Tnaiss _det _time_naiss _prep _loc_naiss _punct _loc_naiss _punct) ;
```

Cette règle est un peu illisible, mais elle a été constituée automatiquement en extrayant les racines des arbres de l'analyse.

Poussée à son terme, ce genre d'approche nous permet d'obtenir une analyse plus poussée :



Malgré la perte de lisibilité due au grand nombre d'annotations, regrouper et préciser les informations permet d'améliorer les résultats fournis par le système Question-Réponse. Il est intéressant de constater que la simplicité du moteur permet d'enrichir l'analyse à partir d'elle-même et de quelques informations ciblées.

4.4 Mesures de performance sur l'analyseur de Ritel

L'analyseur de Ritel présenté section 4.1 et, nous le rappelons, construit non par nous mais par des linguistes, est structuré en 72 transformations successives. 68 de ces passes utilisant la transformation à base de règles, une passe fait des recherches dans le DELAS, une reconstruit les nombres et deux font du filtrage. Le grand nombre de passes facilite la maintenance du système. En effet les règles contenues dans les passes ont tendance à être plus simples et à reconnaître une seule catégorie de structures. Les conflits inter-règles sont ainsi minimisés et leur résolution compréhensible. De plus obtenir des structures hiérarchiques profondes et détaillées est favorisé, la grande majorité des règles ajoutant des annotations plutôt que les modifiant. Les macros et classes nommées permettent de construire des bibliothèques de patterns utiles qui sont incluses au besoin dans les passes, évitant les problèmes classiques de duplication de source. Faciliter l'utilisation du multi-passes est donc un plus pour la qualité du résultat.

Les passes contiennent 1 473 règles utilisant 6 654 macros et 3 588 classes pour un total de 1,8 million de nœuds dans les arbres syntaxiques des expressions régulières. Une partie de ces macros contient de grandes listes d'expressions multi-mots telles que 2 600 noms propres, 500 noms de pays, 185 000 noms de ville, 300 noms de langue, etc. Ces listes sont simplement mises sous la forme d'alternatives.

Cette intégration immédiate dans la syntaxe des expressions régulières permet d'utiliser au besoin toutes les possibilités de ces expressions, et en particulier de gérer certains cas d'ambiguïté. Par exemple dans une des listes de prénoms le mot *fusée* est interdit devant le prénom *Ariane* via un lookahead négatif arrière, réglant le potentiel problème de façon compréhensible et minimale.

La vitesse de l'analyse a été mesurée sur 10 000 phrases (260 000 mots) extraites d'un corpus de journaux et dépêches (corpus CLEF). Chaque phrase est analysée en 2,5ms en moyenne, ce qui représente 10 000 mots par seconde. Une telle vitesse est généralement considérée très bonne pour un système produisant un seul type d'annotation, elle est d'autant meilleure pour un système tel que celui de Ritel qui produit des annotations intégrées entre multiples niveaux linguistiques et sémantiques.

Des 1,8 millions de nœuds seuls 50 000 en moyenne sont visités par phrase, ce qui démontre la capacité de discrimination d'un système se basant sur les mots et l'efficacité des filtrages des règles et alternatives. Chaque nœud est visité en une moyenne de 150 cycles CPU (50ns sur un processeur 3GHz) ce qui montre que le principe de considérer les expressions comme un programme pour un interpréteur de motifs est raisonnable.

Discussion

Les moteurs disponibles permettant à un linguiste d'écrire un système de règles pour ses propres analyses sont rares et, en pratique, assez peu à la mode. Ils sont cependant indispensables pour les problèmes très exploratoires où la définition même des annotations voulues est un sujet de recherche. C'est le cas du domaine qui nous intéresse, l'interaction et la réponse aux questions dans un domaine ouvert, où il n'existe pas à l'heure actuelle de schéma d'annotation mûr pour l'analyse des documents et requêtes. Les moteurs de règles existants ne répondant pas à nos besoins, nous en avons proposé un nouveau.

Deux aspects un peu conflictuels sont à prendre en compte lors de la définition d'un tel moteur. Le premier est l'*expressivité* : le moteur permet-il effectivement d'atteindre les résultats recherchés ? Le second est l'*ergonomie* : le moteur facilite-il la construction du système et ensuite sa maintenance ?

Un des premiers points où tout se décide est le choix de la *représentation interne de l'état de l'analyse*. Elle peut être plus ou moins contrainte, et elle peut avoir une plus ou moins grande couverture. La représentation utilisée par GATE [Cunningham et al. 2002] et UIMA [Ferrucci & Lally 2004], par exemple, n'impose pratiquement aucune contrainte et a une couverture en pratique illimitée. L'inconvénient est que l'absence de structure contrainte rend les règles difficiles à écrire et même souvent indéterministes dans leur application. La représentation de Scol [Abney 1996], en contrepartie, est essentiellement limitée à des vecteurs de symboles, et permet de couvrir des analyses en composants simplifiée sans retour sur décision. Nous avons choisi une représentation structurellement assez contrainte mais avec une couverture relativement large : une forêt de mots ou d'annotations, avec la possibilité d'avoir plusieurs labels par nœud. Cette structure donne la possibilité d'écrire des règles l'exploitant avec une syntaxe claire et sans indéterminisme. Elle permet de couvrir toutes les analyses ne demandant pas des relations à longue distance. C'est une limitation forte et qui gagnerait à être supprimée, ouvrant la porte aux analyses syntaxiques ou sémantiques profondes, mais définir une syntaxe de règles lisible et maintenable capable de manipuler de telles relations parait extrêmement difficile.

Le système de règles lui-même est évidemment très important. Les expressions régulières sont bien souvent la base de tels systèmes car elles ont plusieurs avantages : elles sont bien connues des linguistes et elles ont globalement une bonne expressivité tout en restant relativement lisibles. Nous

avons considéré pertinent de continuer cette tradition. Cependant les expressions régulières ne sont qu'un cadre général, et leur application fait toute la différence. Nous avons fait un certain nombre de choix qui nous paraissent aider grandement à l'ergonomie globale du système. En particulier nous avons choisi de travailler sur des mots et non sur des caractères, et d'aider à la structuration en agissant sur trois axes. Tout d'abord nous permettons la définition de classes et macros nommées, que l'on peut considérer similaires à des fonctions. Ensuite nous favorisons la structuration en multiples passes, permettant à chaque passe de rester simple et modulaire. Et enfin nous autorisons à revenir sur des annotations venant des passes précédentes, ce qui permet de travailler localement en fonction des informations disponibles à un moment donné, sachant que rien n'est définitif et que corrections et précisions sont possibles quand des informations demandant un plus grand contexte sont disponibles.

Toutes ces décisions permettent de construire des analyseurs très structurés et modulaires, facilitant grandement leur développement et leur évolution.

Un autre point que nous considérons important est l'intégration au besoin d'autres approches, qu'elles soient statistiques ou algorithmiques. Les intégrer permet d'avoir une vision d'ensemble de l'analyse.

Enfin un dernier point à ne pas négliger est la vitesse du moteur. En effet un système plus rapide permet plus d'expérimentations. Pouvoir tester ses modifications sur quelques phrases immédiatement et sur une base de documents telle que le corpus de QA@Clef en deux heures permet un retour rapide qui permet d'éviter de s'engager sur de mauvaises pistes.

Mais le vrai test de qualité d'un tel système passe par l'utilisation qui en est faite. Les linguistes l'utilisant pour écrire des grammaires sont globalement satisfaits, et nous avons présenté certaines de leurs réalisations chapitre 4. Une de ces réalisations, une analyse multiniveaux unifiée de la langue, est la base sur laquelle nous nous reposons pour notre approche du problème de Question-Réponse, que nous décrivons dans les parties suivantes.

Deuxième partie

Question-Réponse pour l’interaction

Introduction

Nous nous intéressons, dans cette partie, à la recherche d'une information précise en réponse à une question précise. Cette thématique, plus connue sous le nom de Question-Réponse, a vu ses activités exploser à cause notamment des campagnes d'évaluation qui ont aidé à sa définition. Mais que recouvre ce terme ?

Tout d'abord on veut comprendre une question en langue naturelle. Cette question peut prendre diverses formes, voici par exemples les types de question tels que définis par une campagne d'évaluation récente :
– Questions factuelles : *Quand Gorgoroth a-t-il eu des problèmes avec la police ?*
– Définitions : *Qu'est-ce que le racisme ?*
– Questions oui/non : *L'Aloe-Vera est-il un antioxidant ?*
– Pourquoi : *Pourquoi Michael Jackson a-t-il été poursuivi en justice en 2005 ?*
– Comment : *Comment retirer une tâche de vin rouge ?*
– Listes : *Quels sont les six pays ayant fondé l'Union Européenne ?*
Se pose alors le problème de la définition de ce qu'est une réponse. C'est d'ailleurs un problème majeur. En effet, prenons un exemple : *Qu'est-ce que l'OTAN ?*, qui fait partie de la classe *question de définition*. Peut-on, doit-on attendre une réponse et une seule ? On peut par exemple attendre comme réponse *Organisation du Traité de l'Atlantique Nord*, considérant qu'une question de définition portant sur un acronyme attend en réponse l'expansion de cet acronyme. Mais une réponse telle que *organisation politico-militaire créée à la suite de négociations entre les signataires du traité de Bruxelles, les États-Unis et le Canada ainsi que 5 autres pays d'Europe Occidentale invités à participer* peut constituer une réponse plus informative.

De même, à une question comme *Qui est Robert Torrens ?*, une réponse telle que *un économiste* est considérée correcte. Toutefois on peut là encore s'interroger sur ce que représente une réponse correcte. Est-ce que la correction de la réponse n'est pas liée à une utilisation réelle, à un véritable besoin d'information ? Une réponse comme *un économiste qui a découvert le principe de l'avantage comparatif* serait plus informative dans un tel cadre.

Un autre exemple concerne aussi une question de définition : *Qu'est-ce que le Grenelle de l'environnement ?* Notre système a répondu *machin*, s'appuyant sur le support *certains ont cru en la*

valeur et l'honnêteté du machin appellé Grenelle de l'environnement.... En dehors de tout contexte d'utilisation, cette réponse peut être considérée comme formellement correcte, même si son contenu informationnel est quasi-nul. Mais qu'en est-il réellement ?

Comme nous le voyons, en dehors de toute application réelle, il est difficile de définir ce qu'est une *bonne* réponse à une question. Les différentes campagnes d'évaluation ont toutes essayé de clarifier ce problème, et finalement nous pouvons constater que, hors cadre applicatif précis, elles ont convergé sur les questions *factuelles*, celles se rapportant à des faits précis, et plus précisément les *factuelles simples* où les réponses attendues tiennent en quelques mots désignant une entité précise. En effet définir ce qu'est une bonne réponse est fondamentalement plus simple pour des questions telles que *Que mangent les koalas ?* ou *En France qui est le président ?*, même si pour cette dernière des descriptions telles que *le chef des armées* sont possibles. Des questions certes factuelles mais plus ouvertes telles que *Pourquoi le ciel est-il bleu ?* ou *Comment créer une image ISO sous Linux ?* sont elles bien plus complexes : la réponse peut être constituée de beaucoup d'éléments, de listes, de phrases...

Nous allons donc essentiellement nous intéresser à ces questions factuelles simples qui ont l'avantage dans un système interactif oral de permettre des réponses courtes. Elle ont aussi l'intérêt de permettre relativement aisément d'évaluer automatiquement les sorties du système voire même d'optimiser automatiquement certains des paramètres, en complétant la référence au besoin [Gillard, et al. 2006a]. Cependant, dans le cas de vraies applications, on constate qu'il y a en pratique davantage de questions complexes. [Kato, et al. 2006] ont observé dans le cadre d'un système avec utilisateurs réels que 34 % des questions n'étaient pas des factuelles simples. Ces 34 % consistaient essentiellement en des questions pourquoi, comment et de définition. Lors d'une expérience avec notre plateforme Ritel [Toney, et al. 2008] nous avons observé que plus de 10 % des questions étaient de type oui/non. Il ne faut donc pas non plus les négliger, et la section 9 présentera quelques travaux préliminaires sur ces autres types de question.

Au-delà de la nature des questions se pose aussi le problème de leur forme. Les questions traditionnelles sont des questions écrites dans une langue correcte et bien formée, comme nous avons pu voir dans les exemples. Le cadre interactif demande une plus grande flexibilité sur ce point. Non seulement les questions peuvent être transcrites de l'oral, avec hésitations, reprises, changements d'avis et la syntaxe spécifique de l'oral, mais elles peuvent en plus être incomplètes car faisant référence explicitement ou implicitement à d'autres informations venant de l'historique du dialogue. Ces problèmes de complétion d'historique et de résolution d'anaphore sont du domaine du gestionnaire de dialogue et sortent du cadre de cette thèse. Cependant les approches utilisées doivent permettre d'incorporer des éléments d'informations supplémentaires à la question elle-même que le gestionnaire de dialogue considère pertinents pour la compléter.

Le type et la nature des questions ne sont cependant pas les seuls points à prendre en compte dans le domaine de Question-Réponse. Le type de documents dans lesquelles les réponses vont être recherchées a aussi son importance. Les bases de données, populaires pour les systèmes de dialogue oraux en domaine fermé classiques, sont trop limitées pour de telles questions en domaine ouvert. L'approche normale de Question-Réponse est de chercher les réponses dans des ensembles de docu-

ments en langue peu ou non-structurés. Les évaluations proposées se sont longtemps limitées à des textes journalistiques. Ces textes ont trois avantages : ils peuvent correspondre à un besoin utilisateur réel, sont écrits dans une langue raisonnablement correcte et normalisée, et restent quand même relativement simples au niveau de la langue en comparaison à des œuvres littéraires. Depuis plusieurs alternatives ont été envisagées : les transcriptions de parole, dont la syntaxe est différente de celle de l'écrit, des encyclopédies, en particulier Wikipédia, dont la structure est très spécifique et où la redondance d'information est assez faible, ou encore des documents tout venant du Web, où niveau de langue et qualité typographique sont extrêmement variés. Dans notre cadre interactif où le système Question-Réponse n'est qu'un moyen et non une finalité, nous ne pouvons nous permettre de nous limiter à un type de documents qui risquerait de nous limiter à une catégorie d'informations. Nous préférons donc éviter des méthodologies s'appuyant spécifiquement sur des types ou structures de documents précis.

Indépendamment de ces problèmes de questions et de documents, se pose celui de la langue. Les algorithmes présentés dans cette partie sont essentiellement indépendants de la langue des questions et documents. Cependant les ressources linguistiques disponibles dans chaque langue diffèrent considérablement. Nous nous sommes donc concentrés sur ce qu'il était possible de faire avec les ressources disponibles publiquement en français. Nous verrons cependant dans la partie III que l'ensemble a aussi été évalué sur l'anglais et l'espagnol.

Enfin un dernier point concerne le contrôle des temps de réponse. Un cadre interactif demande une bonne réactivité de l'ensemble du système de dialogue sous peine que l'utilisateur ne se lasse. Nous devons donc choisir des approches permettant de maîtriser autant que possible les temps de réponse maximaux. Nous verrons que ce problème a à ce jour rarement été étudié dans la littérature.

Dans la section suivante nous présentons un état de l'art sur les systèmes Question-Réponse pour questions factuelles dont la discussion donne lieu ensuite à la présentation du plan du reste de la partie. L'évaluation de tout cela est le thème de la partie III.

Chapitre 5

État de l'art

5.1 Présentation générale des systèmes Question-Réponse

La figure 5.1 illustre la structure classique pour un système Question-Réponse. Tout commence par le prétraitement des documents en vue de leur indexation. Ce prétraitement peut être divisé en deux parties : la première, qui n'existe d'ailleurs pas toujours, consiste en une forme d'analyse de la langue qui essaie de pré-extraire de l'information des documents. Cette analyse peut être relativement simple, s'arrêtant aux parties du discours ou aux entités nommées, comme dans [Molla, et al. 2006 ; Molla, et al. 2007 ; Comas, et al. 2007]. Ou elle peut être plus poussée et atteindre des niveaux d'analyse syntaxique ou même sémantique [Laurent, et al. 2006 ; Hickl, et al. 2006 ; Neumann & Wang 2007].

La seconde moitié du prétraitement couvre le formatage des documents pour l'indexation. Bien souvent les créateurs de systèmes préfèrent ne pas travailler avec des documents bruts complets comme unité d'indexation. La tendance est plus au découpage, parfois en phrases, plus souvent en des blocs de quelques lignes [Laurent et al. 2006]. Même pour les cas où il n'y a pas de structure en phrases intrinsèque dans les documents, comme dans le cas de transcriptions de parole, il peut être utile de tenter de reconstruire des blocs équivalents [Kürsten, et al. 2008].

L'indexation des documents ainsi traités est alors effectuée par un moteur de recherche. Lucene [Apache 2007], un moteur de recherche développé par la fondation Apache, est très populaire dans le domaine [Neumann & Wang 2007 ; Comas et al. 2007]. MG (Managing Gigabytes) est aussi utilisé, par exemple dans [Neumann & Wang 2007 ; Comas et al. 2007], mais l'arrêt de son développement depuis quelques années nuit à son succès. Un certain nombre de systèmes utilisent leur propre moteur d'indexation et de recherche, en particulier ceux qui font des analyses linguistiques dans le prétraitement et veulent pouvoir faire des recherches dans les résultats de ces analyses. [Laurent et al. 2006] avec ses analyses profondes et son indexeur spécifique est un bon exemple de ce cas.

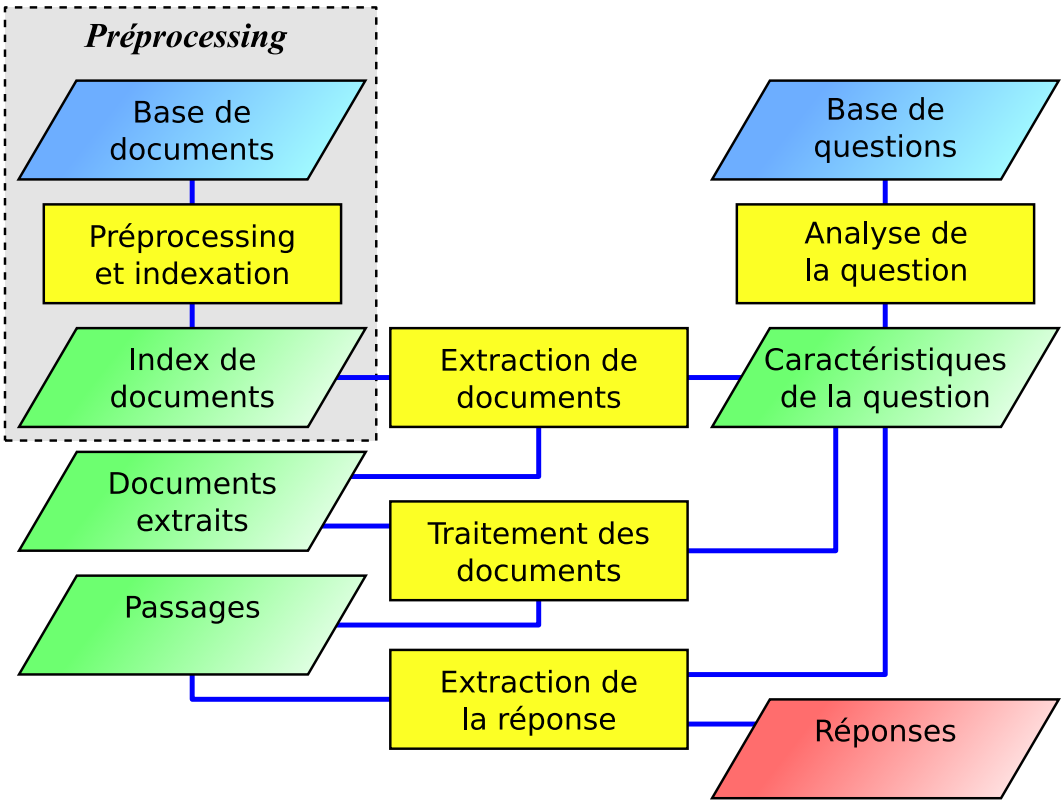


FIG. 5.1 – Architecture générale des principaux systèmes QR (inspiré de [Ligozat 2006]). La partie *préprocessing* est celle effectuée avant d'avoir les questions.

Une fois l'indexation faite, il est temps de s'intéresser aux questions. L'analyse des questions a deux objectifs : le premier est de détecter quelles informations venant de la question doivent être retrouvées dans les documents. Ces informations ont bien souvent la forme de mots-clés et d'entités nommées, mais on rencontre parfois des relations syntaxiques ou sémantiques. Cette partie de l'analyse des questions est généralement assez proche de celle faite pour les documents de façon à pouvoir comparer les résultats obtenus, ce qui est la base des étapes d'extraction de réponse. Le second objectif est la prédiction des types de réponse attendus [Pardiño, et al. 2008]. Ces types sont souvent des types d'entités nommées (personne, lieu, organisation...) mais peuvent être plus précis ou avoir une plus grande couverture quand des taxonomies plus avancées sont utilisées [Laurent et al. 2006].

Les résultats de l'extraction d'informations de la question sont passés au moteur de recherche d'information qui sélectionne alors des documents ou des passages tels que définis par l'indexation. Une analyse complémentaire des documents extraits est souvent effectuée dans la continuité de celle du prétraitement. En pratique, décider quelle partie de l'analyse des documents doit être effectuée avant l'indexation et laquelle après combine un problème d'ingénierie, où entrent en jeu le temps de préprocessing, le temps de réponse, la complexité de l'indexation et les problèmes de passage à l'échelle,

et un problème de recherche où il faut décider quelles informations spécifiques sont nécessaires à l'extraction des documents. Dans tous les cas, l'étape finale extrait les candidats réponse des passages analysés et leur donne un rang. Dans la plupart des cas les candidats sont les mots ou groupes de mots annotés avec les types attendus pour la réponse. Un score est donné à chacun qui peut être basé sur les distances réponse-mot-clé [Pardiño et al. 2008], une mesure générale de densité [Gillard, et al. 2006c ; Comas & Turmo 2008] ou même des similarités syntaxiques et relations de dépendance [Bouma, et al. 2005]. Certains systèmes vont même plus loin en cherchant à confirmer leurs résultats dans d'autres sources, et en particulier le web [Plamondon & Kosseim 2003]. Le système classe ensuite les réponses en fonction des scores obtenus.

Cette structure, très générale, couvre la plupart des systèmes s'appuyant sur des approches linguistiques. Des méthodes alternatives existent dont le but est de ne requérir que peu de connaissances linguistiques. [Berger, et al. 2000] par exemple utilise des modélisations statistiques construites sur des mesures de co-occurrence. [Ittycheriah & Roukos 2002] ajoute à cela une sélection de passages s'appuyant sur un modèle de traduction *IBM model 1* en considérant un bon passage comme étant une traduction de la question. Ils ajoutent aussi une extraction automatique de patrons de réponse. Ces systèmes ajoutent des traits linguistiques simples aux entrées de leurs modèles statistiques. [Whittaker, et al. 2007] tente d'aller encore plus loin en éliminant toute connaissance linguistique en s'appuyant exclusivement sur des modèles statistiques probabilistes.

5.2 Un système très linguistique : le système du LCC

Le système de la *Language Computer Corporation* [Moldovan, et al. 2002b] a participé à de nombreuses évaluations avec de très bons résultats. Il est un bon exemple des systèmes à base de connaissances linguistiques que nous avons présenté précédemment.

Les documents sont indexés tels quels, la totalité de leur analyse étant déplacée après la recherche d'informations. Le vrai travail commence avec les questions. Une première passe consiste en une correction orthographique de la question, en particulier au niveau des noms propres qui sont primordiaux dans les recherches. De plus, au besoin, les questions sont réécrites dans une forme où le mot de question apparaît en premier. Cette forme est ensuite analysée en parties du discours, chunks et syntaxe.

La forme analysée de la question est utilisée à tous les niveaux dans la recherche. En premier lieu les parties du discours sont mises à profit pour sélectionner les mots-clés considérés pertinents pour la sélection des documents. Ces mots-clés sont étendus via des transformations lexicales (abandonné → abandonner), morphosyntaxiques (→ abandon) et sémantiques via WordNet (→ oublié). Ils sont alors passés au moteur de recherche qui extrait les documents contenant les conjonctions de ces mots-clés en relâchant les contraintes jusqu'à obtenir un nombre de documents jugé acceptable (environ 2 000). De ces documents sont extraits des passages en prenant dix lignes avant et après les groupements de mots-clés.

La question entre à nouveau en jeu à ce niveau là. Les dépendances syntaxiques entre les chunks de la question sont extraites et servent à construire un ensemble de contraintes, en particulier temporelles et géographiques. Les passages ne les respectant pas sont éliminés.

Les passages ainsi filtrés, il est temps de passer à la détection des candidats réponse. Le choix du type de réponse attendu (personne, lieu, montant, distance...) est effectué via une hiérarchie de types constituée semi-automatiquement et s'appuyant sur les synsets WordNet associés aux mots de la question [Pasca & Harabagiu 2001]. L'aspect semi-automatique vient du fait que la hiérarchie de types et ses associations avec les synsets est initialement remplie automatiquement via un corpus de paires question/réponses et les résultats sont ensuite filtrés par des humains.

Le type de réponse attendu décidé, il est temps d'extraire et d'évaluer les candidats réponse. Un détecteur d'entités nommées, enrichi par le vocabulaire de WordNet pour les types ne faisant pas partie des entités classiques, extrait les candidats réponse. Leur évaluation est effectuée via un système de raisonnement logique. Une expression logique est construite représentant la question en se basant uniquement sur l'analyse syntaxique [Moldovan, et al. 2002a]. Les mots, décorés de leur partie du discours, deviennent des noms de prédicats prenant comme paramètres des variables référençant les autres mots et chunks avec lesquels ils sont en lien syntaxique. La conjonction de ces fonctions donne une expression logique représentant la question. Les passages sont transformés de la même façon. Le contenu d'Extended Wordnet [Harabagiu, et al. 1999], qui contient l'ensemble des données de WordNet encodées de la même façon, est ajouté comme axiomes. Cet ensemble de prédicats est traité par le moteur d'inférences *Otter* [McCune 1994] pour tenter de trouver les meilleures réponses.

À titre d'ordre de grandeur, ce système a trouvé une réponse correcte pour 71% des questions factuelles à l'évaluation TREC 2005, le classant premier (66% pour le second, 32% pour le troisième), et 58% à TREC 2006, le classant là aussi premier (39% pour le second, 32% pour le troisième).

5.3 Un système purement statistique : le système du Tokyo Institute of Technology

Les systèmes présentés par le *Tokyo Institute of Technology* [Whittaker, et al. 2005a ; Whittaker, et al. 2006] sont un peu l'opposé de ceux du LCC. L'idée est de construire un système purement statistique sans aucune connaissance linguistique. D'un point de vue probabiliste, trouver la meilleure réponse possible à une question consiste à construire un modèle $P(R|Q)$ capable de donner la probabilité d'une réponse étant donnée une question et ensuite de garder, dans l'ensemble des réponses possibles, celle qui a la meilleure probabilité.

$$R = \arg\max_{r} P(r|Q) \tag{5.1}$$

Le modèle proposé décompose l'information donnée par la question en deux parties : l'un, noté

X, représente les élements à rechercher dans les documents. L'autre, noté W, représente le type de réponse attendue.

$$R = \arg\max_r P(r|W, X) \tag{5.2}$$

Après quelques hypothèses simplificatrices (indépendance de W et X pour un r donné, équiprobabilité a priori des candidats réponse), l'équation prend la forme :

$$R = \arg\max_r P(r|X)P(W|r) \tag{5.3}$$

Le problème se décompose ainsi en deux parties : une recherche d'informations, représentée par $P(r|X)$, qui extrait les candidats réponse en rapport avec les éléments de la question et un filtrage, $P(W|r)$, qui sélectionne parmi ces candidats ceux qui correspondent au type de question voulu. À ces deux modèles s'ajoute un algorithme s'appuyant sur des probabilités permettant d'extraire dans les documents les phrases intéressantes. Nous décrivons ces modèles dans l'ordre de leur utilisation.

Étant donnée une question, la première étape consiste à extraire les phrases les plus pertinentes. La méthode est simple : un modèle de langage unigramme est construit pour chaque phrase et document, lissé par *absolute discounting* [Ney, et al. 1994]. Une probabilité peut alors être calculée pour la question étant donné le modèle d'une phrase ou d'un document. La probabilité calculée sur la phrase est combinée linéairement à celle calculée sur le document qui la contient pour obtenir le score final de chaque phrase. La question est notée Q, le document D et la phrase examinée S.

$$\text{Score}(S) = \alpha P(Q|S) + (1 - \alpha)P(Q|D, S \in D) \tag{5.4}$$

Pour l'évaluation QAst 2007 cette extraction a été enrichie par une expansion de requête [Whittaker et al. 2007]. Des classes non-disjointes contenant des ensembles de mots sont construites pour représenter les thèmes possibles. La méthode de construction n'est pas indiquée. [Peat & Willett 1991] donne des exemples de méthodes possibles, incluant certaines ne demandant aucune connaissance linguistique (mesures de co-occurrence, calculs d'information mutuelle...). Les mots de la question sont alors «étendus» en l'ensemble des mots des classes auxquels ils appartiennent avec une probabilité uniforme. Un score étendu peut alors être calculé de la même façon, le calcul étant simplifié par le fait que les modèles ne sont qu'unigrammes. Le score final est une interpolation linéaire entre le score simple et le score étendu. Cette expansion n'est utilisée que pour l'extraction de phrases, et n'a permis d'obtenir qu'un gain très faible en pratique.

Une fois les phrases pertinentes obtenues, le modèle de recherche d'informations $P(r|X)$ tente d'extraire les candidats réponse intéressants indépendamment du type de question. La méthode de sélection des suites de mots représentant des candidats n'est pas précisée. On peut penser que pour chaque phrase toutes les suites de mots jusqu'à une certaine taille limite sont évaluées. Les mots vides (stopwords), choisis comme étant les 50 mots les plus présents dans les documents, sont supprimés de la question. L'ensemble des suites de mots, de toutes tailles, contenues dans la question ainsi simplifiée

constitue l'ensemble X des éléments à rechercher. L'évaluation d'un candidat réponse est calculée comme la moyenne de probabilités élémentaires sur tous les sous-ensembles de X :

$$P(r|X) = \frac{1}{2^{|X|}} \sum_{x \subset X} P(r|x) \tag{5.5}$$

Chaque probabilité élémentaire est calculée par maximum de vraisemblance sur l'ensemble des phrases extraites dans la première étape :

$$P(r|x) = \frac{N(r,x)}{Z(x)} \tag{5.6}$$

$$N(r,x) = \text{count}(S, r \in S \wedge x \subset S) \tag{5.7}$$

$Z(x)$ est choisi pour normaliser le résultat. En pratique le calcul de $N(r,x)$ est légèrement modifié pour tenir compte des phrases autour avec un poids inférieur à 1 nommé λ_{adj}. Notant S^+ et S^- la phrase après et avant une phrase S donnée et avec un λ_{adj} entre 0 et 1 (0,3 pour TREC 2005),

$$N(r,x) = \text{count}(S, r \in S \wedge x \subset S) + \lambda_{\text{adj}}\text{count}(S, r \notin S \wedge (r \in S^+ \vee r \in S^-) \wedge x \subset S) \tag{5.8}$$

Le filtrage des réponses en fonction d'une forme de type de question est un peu plus compliqué [Whittaker, et al. 2005b]. Le principe est de comparer la question demandée à un ensemble de paires question/réponse (environ 290 000) d'un corpus d'apprentissage. Notant E l'ensemble des paires (q,a) d'apprentissage, et r représentant toujours la réponse potentielle examinée, le filtrage est approximé par :

$$P(W|r) = \frac{1}{|E|} \sum_{(q,a) \in E} P(W|q)P(a|r) \tag{5.9}$$

La structure de traits contenue dans W est similaire à X dans le sens où elle est constituée de l'ensemble des suites de mots de toutes tailles présente dans une version filtrée de la question. Le filtrage est cependant différent : un ensemble d'environ 2 500 mots considérés pertinents pour caractériser une question sont conservés, le reste supprimé. Cet ensemble de mots a été choisi statistiquement à partir de E. La probabilité $P(W|q)$, mesurant la ressemblance de la question posée à une du corpus, est estimée comme la proportion des suites de mots de W dans la question du corpus filtrée de la même façon.

L'estimation de $P(a|r)$ se fait via des classes de mots de réponse. Environ 5 000 classes contenant chacune un ensemble de mots sont construites par clustering agglomératif en partant des mots semblant les plus pertinents des questions et en se basant sur des statistiques de co-occurrences calculées sur une grande quantité de documents. De plus, il est pris comme hypothèse simplificatrice que seuls les mots de même indice dans les réponses sont à comparer. Le calcul final est alors, avec r, réponse évaluée et a réponse du corpus d'apprentissage vues toutes les deux comme un vecteur de mots, et

C_A l'ensemble des classes de mots :

$$P(a|r) = \prod_{i=1}^{|a|} \sum_{c \in C_A} P(a_i|c) P(c|r_i) \tag{5.10}$$

Enfin la combinaison des deux probabilités de recherche d'informations $P(r|X)$ et de filtrage $P(W|r)$ est en pratique insuffisante. Les approximations des modèles rendent les magnitudes de leur valeurs différentes et un facteur de correction α, dans la tradition des modèles log-linéaires, aide à compenser cette différence :

$$R = \arg\max_r P(r|X)^\alpha P(W|r) \tag{5.11}$$

En comparaison avec celui du LCC, ce système a trouvé une réponse correcte pour 21% des questions factuelles à l'évaluation TREC 2005, le classant 11e, et 25% à TREC 2006, le classant 9e.

5.4 Un système intermédiaire : le système du LIMSI-LIR

Le système de Question-Réponse du LIMSI-LIR, connu sous le nom de QALC [Berthelin, et al. 2003] pour le travail sur l'anglais et FRASQUES [Grau, et al. 2005a] pour le français, est un exemple de système intermédiaire. Par intermédiaire nous entendons un système qui utilise des connaissances linguistiques mais en variété et couverture limitées par ce qui est disponible. La plus grande partie des systèmes rencontrés dans les évaluations relève de cette catégorie.

Le prétraitement des documents est assez simple : ils sont tout d'abord découpés en paragraphes puis lemmatisés. Le résultat est alors indexé par MG [Belle, et al. 1994] pour l'anglais ou Lucène [Apache 2007] pour le francais.

L'analyse des questions est beaucoup plus poussée. Quatre informations sont extraites. La première est la catégorie de la question, factuelle simple ou définition. La seconde est un *type prédit pour la réponse*. Ce type peut soit être un type d'Entité Nommée soit un type général. Par exemple la question *Dans quelle ville le procès de Giulio Andreotti a-t-il eu lieu ?* est associée au type d'Entité Nommée *lieu-ville* et *Quel contrat a pris place entre 1995 et 2004 ?* appelle le type général *contrat*. La troisième information est l'ensemble des entités nommées de la question, comme la personne *Guilio Andreotti* ou les années *1995* et *2004*. Enfin la dernière information est le *focus* de la question. Cette notion est définie de manière un peu intuitive comme étant l'objet sur lequel porte la question et qui sera attendu à proximité de la réponse, comme *procès* ou *contrat*.

Cette analyse effectuée, des requêtes sont construites à partir du focus et des entités nommées et passées au moteur d'indexation pour obtenir les paragraphes de documents pertinents. Des relâchement de contraintes (suppression d'une partie des mots clefs) et des ajouts de synonymes ont lieu jusqu'à obtenir une centaine de paragraphes.

Le système passe alors à l'analyse de ces paragraphes. La première étape cherche à reconnaître les éléments de la question présents dans les documents. Une comparaison simple des mots est bien évidemment insuffisante, les variations de forme pour exprimer le même fond étant courantes. Fastr [Jacquemin 1996] est utilisé pour générer des variantes des phrases des paragraphes qui sont alors comparées avec les éléments importants de la question, en particulier le focus. Un score de paragraphe est alors calculé en s'appuyant sur ces comparaisons, et les plus pertinents sont conservés (environ 70% du nombre initial). Les paragraphes sont ensuite découpés en phrases. Les phrases qui ne contiennent pas d'élément de la question, sous forme d'origine ou en variante, sont supprimées.

Les phrases pertinentes ainsi extraites, le système passe à l'extraction des réponses. Deux stratégies sont employées suivant la catégorie à laquelle le type prédit pour la réponse appartient. Si ce type appartient à la catégorie entité nommée, les positions dans la phrase des éléments reconnus de la question sont relevées. Leur barycentre est calculé, donnant une nouvelle position dans la phrase. L'entité nommée la plus proche de cette nouvelle position est sélectionnée comme réponse. Sinon, dans le cas des types généraux, un ensemble de patrons d'extraction est appliqué, ces patrons s'appuyant à la fois sur les mots de la phrases et sur une annotation en parties du discours de celle-ci. Un score final de réponse est calculé pour trier les résultats. Ce score est élaboré à partir de plusieurs éléments incluant le score donné au paragraphe par le moteur d'inxation, la présence ou non des éléments de la question dans la phrase et leurs scores Fastr associés ainsi que des poids attribués au différents patrons.

Les systèmes de LIMSI-LIR n'ont pas participé à TREC 2005 ou 2006, empêchant une comparaison directe. Cependant le système QALC (en anglais) a obtenu à TREC 2002 un CWS de 0,497 pour la neuvième place (0,856 pour le premier, là encore le LCC) et le système FRASQUES (en français) a obtenu à l'évaluation Equer un MRR de 0,22, le plaçant troisième (le premier ayant un MRR de 0,58 et le second de 0,25).

5.5 Le problème du temps de réponse

Contrairement aux systèmes de recherche d'information tels que Google, AltaVista ou Exalead, où les temps de réponse sont de l'ordre de la milliseconde, les systèmes de Question-Réponse typiques ont besoin de plusieurs secondes voire minutes pour donner une réponse. Le problème des temps de réponse a rarement été considéré important dans le domaine. En 2001, [Kim, et al. 2001] présentait un système capable de fournir une réponse en une moyenne de 0,029 secondes par question mais la collection de documents était limitée à 60Ko de texte ce qui rend toute comparaison difficile.

L'évaluation CLEF 2006 a proposé une tâche nommée *Real-Time QA Exercise* où les systèmes devaient répondre le plus vite possible à 20 questions en espagnol. Le temps de réponse était pris en compte dans l'évaluation finale. Cinq systèmes ont participé, mais seul les auteurs de Miracle [de Pablo-Sanchez, et al. 2006] décrivent leur expérience. Leur système a une structure standard et emploie une analyse relativement profonde de la langue, combinant entités nommées et annotations sémantiques.

Leur travail dans le cadre des temps de réponse a consisté à déplacer le plus possible de l'analyse des documents de la phase post-extraction des documents vers le préprocessing. Le système initial prenait 198 secondes pour 20 questions, qui sont descendues à 73 secondes après le déplacement. Leur collection de documents était celle de CLEF 2006, soit environ 1Go de textes journalistiques.

Les auteurs de Nexus [Ahn & Webber 2007] considèrent également ce déplacement comme étant un point essentiel. Leur système était capable de donner la réponse à 162 questions sur le corpus Acquaint (3Go de textes journalistiques) en moins d'une minute, avec une moyenne à 0,3 secondes. En plus du déplacement de l'analyse vers le préprocessing, Nexus construit un index contenant toutes les entités nommées, considérées comme des réponses potentielles, et les termes situés dans leur contexte. Les récupérations rapides de réponses permises grâce à cet index sont considérées primordiales par les auteurs pour la performance et la vitesse de leur système.

Dans les deux cas, le moteur de recherche d'informations est générique, Xapian [Xapian 2001] pour Miracle et Lemur [Olgivie & Callan 2002] pour Nexus. La vitesse du moteur ne semble pas avoir été un facteur important.

5.6 Discussion

Nous avons présenté comment, dans le même cadre général des systèmes Question-Réponse, toute une palette de méthodes pouvaient se décliner, de la très linguistique et logique à la pure statistique. Étant donné nos contraintes de flexibilité et de contrôle de vitesse chacune a ses avantages et ses inconvénients. L'approche du LCC, qui donne de très bons résultats, pose plusieurs problèmes. Le premier est le besoin de ressources linguistiques complexes telles que WordNet qui n'ont pas à ce jour d'équivalent en français. EuroWordNet [Vossen 1998] en particulier semble encore considéré insuffisant. De plus les nombreux types d'analyse relativement poussées peuvent poser des problèmes de robustesse face à des questions ou des documents autres que du texte bien écrit. Enfin, d'un point de vue global, ce système a visiblement été construit incrémentalement sur une longue période ce qui donne lieu à un ensemble de modules disjoints dont les différents travaux d'analyse complémentaire sont un peu redondants.

A l'opposé, le système du Tokyo Institute of Technology ne s'appuie sur aucune ressource ou analyse linguistique. Cela lui permettrait en théorie d'être robuste à tout type de questions et de documents. Cependant il nécessite un grand corpus de paires question/réponse d'un type similaire à celui attendu ce qui n'existe pas pour le français, et d'autant moins pour les questions semi-structurées telles qu'obtenues dans une interaction. De plus les performances sont en pratique assez faibles et semblent a priori difficiles à améliorer de façon significative sans réintroduire des approches linguistiques.

Le système du LIMSI-LIR se situe entre les deux en ce qui concerne l'utilisation de ressources et d'analyses linguistiques. Il est en cela plus typique des systèmes généralement rencontrés dans les évaluations officielles. En particulier il utilise un système d'indexation généraliste plutôt qu'un sys-

tème spécialisé. Cela offre l'avantage de profiter de systèmes disponibles de qualité. L'inconvénient est la difficulté de les adapter aux besoins spécifiques de Question-Réponse. Plusieurs difficultés sont à noter. Tout d'abord sélectionner les mots-clés à passer au système d'indexation pour obtenir les documents les plus pertinents est un problème complexe en soi. Il est en plus difficile d'intégrer à ce niveau de la recherche les variations d'expression des éléments de la question. Nous avons pu voir que le système présenté ne le fait qu'une fois les documents sélectionnés. Le reste du système est simple mais performant. Il est à noter que l'utilisation de patrons d'extraction permet d'obtenir souvent une bonne précision de réponse mais ils représentent un gros travail de développement pour obtenir un rappel décent.

Comme nous l'avons vu, les expériences s'intéressant spécifiquement au temps de réponse ont été rares. L'idée principale cependant est simple : préparer au maximum le travail sur les documents avant de recevoir des questions.

En tenant compte de tous ces aspects, nous avons donc décidé d'appuyer notre approche sur une analyse multiniveaux unifiée regroupant autant d'informations que possible. L'analyse des documents est faite avant l'indexation qui prend en compte les résultats pour enrichir son index. Les questions sont analysées de la même manière et les annotations de l'analyse sont les données élémentaires utilisées à tous les niveaux du système, évitant les redondances coûteuses en temps.

Le reste de cette partie porte sur la description des méthodes et algorithmes qui nous ont permis de répondre à ces besoins. Un analyseur de la langue a été développé par d'autres membres de l'équipe en utilisant le moteur présenté dans la partie I. Nous commençons par une description rapide des résultats que produit cette analyse, une description plus complète est donnée à la section 4.1. Un premier système QR simple a été construit s'appuyant sur ces résultats et des ensembles de règles écrites à la main. Il est décrit ainsi que les leçons que nous en avons tirées. Le système final, plus avancé, est alors décrit en détail. Enfin une dernière section présente des approches préliminaires pour traiter des types de questions autres que les factuelles. L'évaluation de l'ensemble est le sujet de la partie III.

Chapitre 6

Description de l'analyse

La première étape de nos systèmes de Question-Réponse est l'analyse de la langue naturelle comme décrit dans [Rosset et al. 2006]. Nous l'avons déjà présentée section 4.1 mais il nous paraît pertinent d'en rappeller les grandes lignes ici.

Par analyse nous entendons l'extraction des informations utiles d'un texte ou d'une question, utiles dans le cadre de la recherche d'information. Ces informations peuvent prendre plusieurs formes. Pour Question-Réponse nous nous basons sur un concept d'entités typées hiérarchiques. Ce sont des paires (type, valeur) où le type est un label fixe et la valeur un groupe de mots du document ou de la question. Les valeurs peuvent être incluses les unes dans les autres, par exemple on peut avoir pour la question "Quelle est la capitale de la France ?" la paire (*pays*, *France*) incluse dans la paire (*ville*, *capitale de la France*), d'où le terme de hiérarchique. Un exemple est donné figure 6.1.

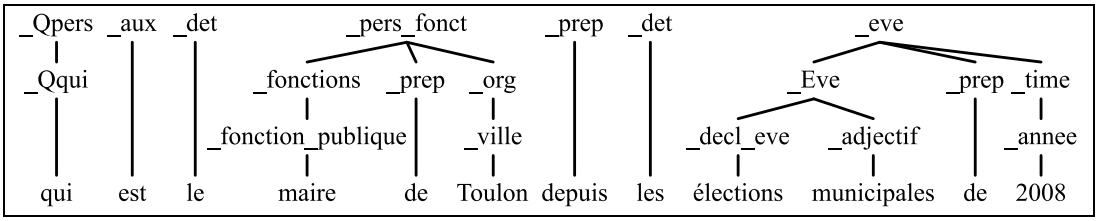


FIG. 6.1 – Exemple d'annotation en entités telle qu'utilisée par Question-Réponse

Ces entités sont regroupées en 4 grandes catégories :

– Entités nommées étendues
– Entités spécifiques
– Entités linguistiques
– Entités couvrant les mots de question

Les entités nommées traditionnelles sont les expressions qui désignent les lieux, personnes, organisations, dates, valeurs et contiennent le *nom* de l'entité et non une périphrase ou description la désignant. Par exemple *Paris* est une entité nommée alors que *capitale de la France* ne l'est pas. En pratique cette définition est trop limitative et il est plus intéressant d'accepter ces expressions ainsi que d'augmenter le nombre de types pour une classification plus fine comme le fait [Sekine 2004].

Les entités spécifiques sont du même genre mais rajoutent des types spécifiques au domaine des documents. Dans le cas de QAst il s'agissait de *système*, *méthode*, *algorithme*, *score*, etc pour les séminaires sur la reconnaissance de la parole ou de *matériaux*, *formes* et *couleurs* pour les réunions sur la conception de télécommandes (cf. [Rosset, et al. 2007]).

Les entités linguistiques servent à couvrir tout le reste. L'idée est d'obtenir un *chunking* du document ou de la question en blocs élémentaires d'information. Le principal type correspond aux noms complexes, comme *imprimante couleur*, *produit de base* ou *couvercle standard*. En complément viennent les noms isolés, adjectifs, adverbes, adjectifs comparatifs, etc.

Enfin une série de types d'entités couvre les mots ou groupes de mots de question comme *qui* ou *quelle méthode*.

Chapitre 7

Une approche préliminaire pour Question-Réponse

Un premier système peut être construit utilisant l'analyse décrite au chapitre précédent. Sa structure générale est présentée figure 7.1. L'approche consiste à prédire le ou les types de réponse possibles étant donné la question, et de chercher la présence d'éléments de ces types dans les phrases des documents contenant les éléments intéressants de la question. Le système se structure en deux niveaux :
- Un ensemble de filtres triant les questions sur les éléments résultants de leur analyse
- Associés à chaque filtre, un ensemble de triplets (clefs de recherche, type de réponse, taille autorisée)

Dans l'ensemble des filtres acceptant la question, le plus prioritaire est choisi. Il lui est associé des triplets décrivant les recherches à effectuer. Ces triplets sont pris séquentiellement. Les paramètres dans les clefs de recherche sont complétés d'après le contenu de la question et l'ensemble des blocs de phrases, leur taille maximale étant donnée par le triplet, contenant toutes les clefs de recherche sont extraits des documents. Les éléments du type de réponse attendu sont extraits et classés en fonction de leur nombre d'occurrences. En l'absence de réponse le système passe au triplet suivant.

Écrire tous ces filtres et triplets à la main est un travail considérable, même avec la généralisation que l'utilisation des types permet. Il nous a cependant été possible de définir une méthodologie pour cette écriture. À partir d'une forme de question, i.e. les types d'entités présentes dedans, la première étape est le *Choix du type de réponse*. Par exemple :

- _Qqui → _pers, _pers_def ou _org
  - Qui a vendu Manhattan ?
- _Qqui + _fonction → _pers
  - Qui est le Pape ?

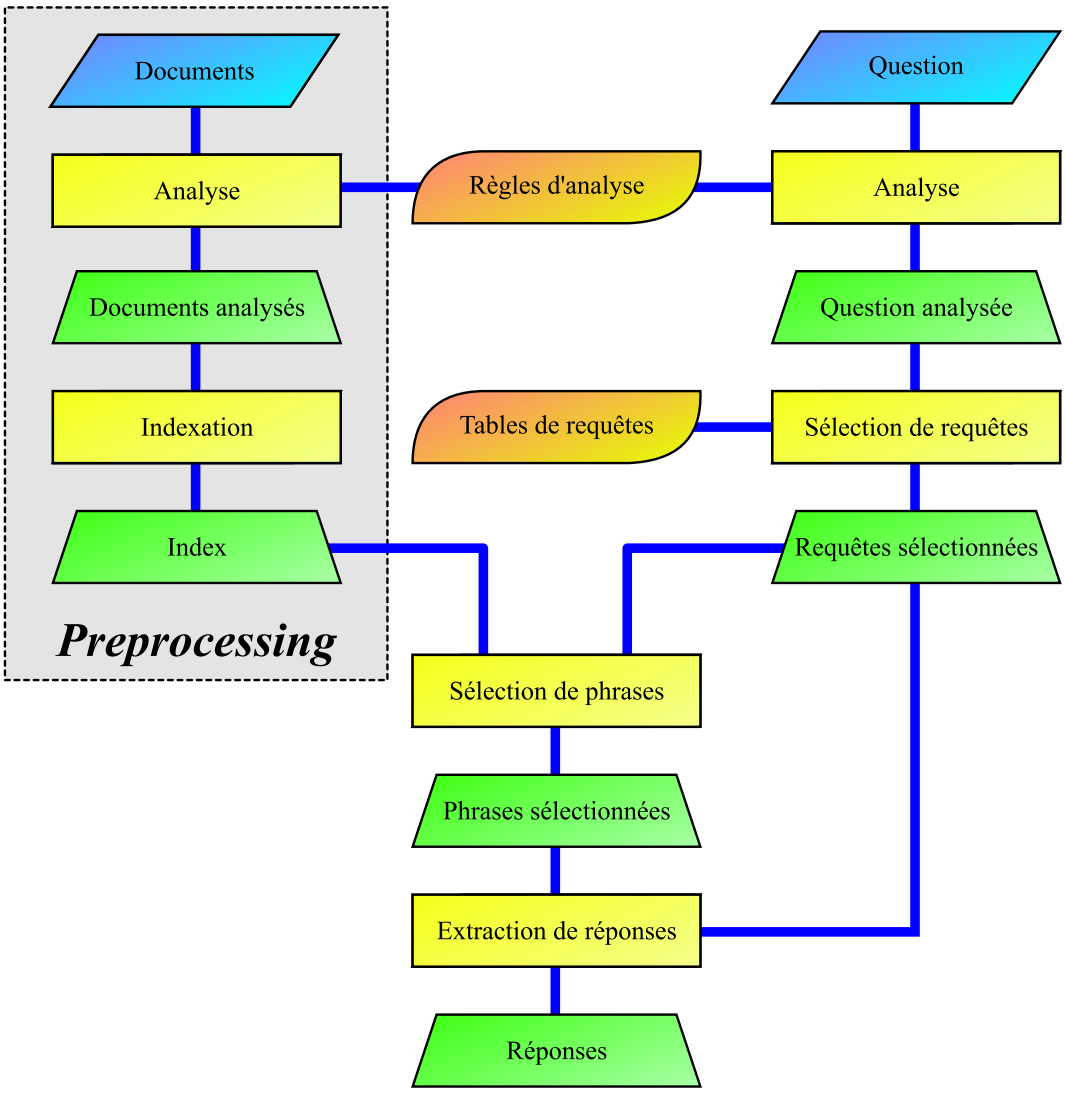


FIG. 7.1 – Structure générale du système Question-Réponse simple. Les parallélogrammes bleus représentent les données fournies. Les boites arrondies orange représentent les règles écrites par un humain. Les rectangles jaunes représentent les modules de calcul. Les trapèzes verts les résultats des modules.

Dans le cas où plusieurs types sont possibles, ils sont classés par ordre de pertinence a priori. La seconde étape est la *Classification des entités de la question par importance*. En particulier les entités nommées sont plus importantes que les noms composés, qui sont eux mêmes plus importants que les verbes, etc. Certaines entités considérées non-pertinentes comme les déterminants sont supprimées de la liste à prendre en considération.

Une fois ces deux étapes accomplies il est possible de construire la liste de requêtes. La première requête contient la totalité des entités, le type de réponse le plus probable et une taille acceptée minimale. Les requêtes suivantes sont produites par relâchements sur la requête initiale :
– Augmentation de la taille maximale acceptée
– Passage aux types de réponse moins probables
– Suppression des entités les moins importantes
– Changement de types pour certaines entités (par exemple _loc → _org)
– Variations de valeurs sur certains types (par exemple Bush → Georges Bush)
Il faut, bien sûr, arrêter les relâchements avant que les requêtes ne deviennent trop générales et perdent toute signification relativement à la question.

Quniv ∧ projet(*nom-projet*)		
Clés de recherche	**Type de réponse**	**Taille autorisée**
projet(*nom-projet*)	univ	2
evaluation(*nom-projet*)	univ	2
np(*nom-projet*)	univ	2
projet(*nom-projet*)	organisation ∨ acronyme	2
np(*nom-projet*)	organisation ∨ acronyme	2

FIG. 7.2 – Exemple de règle de requête

La figure 7.2 montre un exemple d'un tel bloc de requêtes. Une question telle que *Quelle université participe au projet Ears ?* donne comme résultats importants de l'analyse une entité *Quniv* et une *projet* avec comme valeur *Ears*. La première requête cherche donc un nom d'université dans la même phrase que le nom du projet. Si le système n'en trouve pas, il essaie dans la phrase précédente et la suivante (taille maxi=2). Le relâchement suivant permet de trouver les cas où le nom du projet a été classifié comme étant une évaluation (*L'Université de Cambridge participe à l'évaluation Ears.*). Le suivant accepte le nom de projet en tant que simple nom propre. Ears pourrait être vu ainsi s'il n'était pas connu dans les listes de l'analyseur mais simplement grâce aux mots déclencheurs *projet* ou *évaluation*. Enfin les deux derniers relâchements permettent de récupérer les cas où le nom de l'université n'a pas été reconnu en tant que tel mais en tant qu'organisation (Cambridge sans préfixe serait probablement classé ville et organisation) ou même simplement acronyme.

Cette approche a plusieurs problèmes. Le choix de la nature et de l'ordre des relâchement est entièrement à la charge de l'expert linguiste et ces choix sont en pratique assez difficiles car rigides, et du coup arbitraires, en l'absence de scores. De plus il est facile de perdre le fil : dans l'exemple présenté, tiré du système réel, rien ne justifie l'absence du relâchement sur *evaluation* dans la seconde série. Le nombre de requêtes augmente très rapidement (plus de 5000 pour le système QAst 2007), rendant leur maintenance difficile, en particulier pour les synchroniser avec des améliorations et extensions de l'analyseur, et malgré ce nombre leur couverture reste insuffisante et des entités importantes sont ainsi parfois perdues pour la recherche.

Outre ces problèmes de développement et de maintenance, ce système ne permet pas de contrôle du temps de réponse. La vitesse dépend uniquement du nombre de passages retournés. Les passages n'étant ni scorés ni classés, il n'est pas possible de fixer une limite pertinente sur leur nombre. Cela a parfois provoqué des comportements catastrophiques dans un cadre interactif avec des recherches prenant plusieurs minutes. C'est pour répondre à ces observations qu'un deuxième système a été conçu puis implémenté.

Chapitre 8

Un système plus avancé

8.1 Organisation générale

Le système de base présenté dans le chapitre précédent nous a permis de dégager un certain nombre de principes de fonctionnement pertinents pour la recherche :
– Des entités de la question, certaines sont importantes, d'autres moins, et d'autres sont inutiles.
– Les entités de la question peuvent êtres trouvées dans les documents à l'identique, ou dans des formes plus ou moins modifiées, et ce de façon dépendante du type.
– Les entités de la question permettent de décider de types de réponse attendus et pour une question donnée certains types sont plus pertinents que d'autres.
– La distance des candidats réponse aux éléments de la question trouvés dans les documents est à prendre en compte dans un score.
– Il faut pouvoir poser une limite à la quantité de travail effectuée à toutes les étapes de la recherche.
Ces principes généraux n'ont rien de très original en soi et sont sous-jacents à la plupart des systèmes existants. Nous avons cependant décidé de les appliquer plus explicitement que ce n'est fait habituellement. Les trois premiers portent sur l'utilisation des entités de la question dans le cadre de la recherche de la réponse. L'ensemble des décisions associées (niveaux d'importance, variations de formulation, types de réponses attendus) sont regroupées dans une structure abstraite mais compréhensible que nous nommons un *Descripteur De Recherche*. Le système lui-même est décomposé en une série d'étapes élémentaires dont la structure globale est donnée figure 8.1. Les deux étapes critiques pour la performance, extraction des documents et des réponses, ont leur temps de calcul limité par des *paramètres de contrôle de performance*. Enfin la notion de proximité est utilisée à plusieurs niveaux, dans la construction des passages et dans le calcul du score associé aux candidats réponse.

Le *Descripteur De Recherche* est un concept fondamental de ce nouveau système. Cette structure représente la recherche à effectuer. Elle est *complète*, contenant toutes les informations nécessaires à la recherche. Elle est *structurée* et *synthétique*, résumant ces informations efficacement permettant

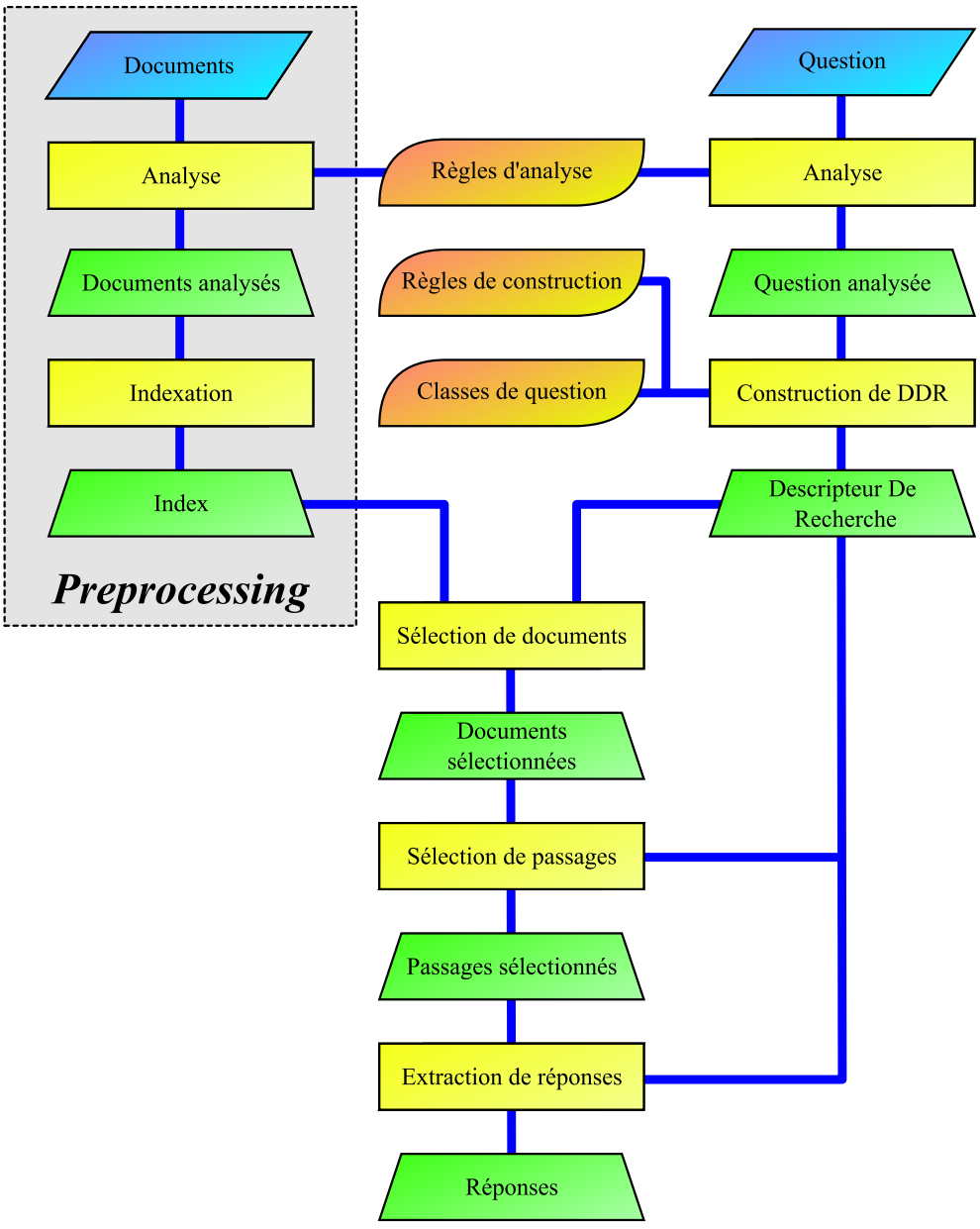


FIG. 8.1 – Structure générale du système Question-Réponse avancé. Les parallélogrammes bleus représentent les données fournies. Les boites arrondies orange représentent les règles écrites par un humain. Les rectangles jaunes représente les modules de calcul. Les trapèzes verts les résultats des modules.

aux algorithmes de s'appuyer directement dessus. Et elle est *lisible*, permettant à un humain de les lire, les comprendre, d'évaluer leur qualité générale et même de les modifier. Le descripteur de recherche

est présenté section 8.2.

Un autre point important est le contrôle des temps de réponse. Cela implique de contraindre deux facteurs principaux :
– Le temps passé en entrées/sorties
– Le temps passé en calcul
Dans le système de base, le temps d'entrée-sorties est dominant, spécifiquement le temps pris par la lecture des lignes individuelles désignées comme intéressantes par l'index. Les disques durs modernes sont capables de grands débits en lecture linéaire mais ont besoin de quelques millisecondes pour changer d'endroit où lire. En conséquence de nombreuses petites lectures en des endroits dispersés et non prévisibles par le système d'exploitation sont particulièrement inefficaces. Il est bien plus efficace de choisir un petit nombre de documents en fonction de la question et de les lire entièrement.

Une fois ces documents choisis et lus, il faut y chercher les réponses possibles. Cependant, les calculs nécessaires pour estimer le score d'une réponse peuvent être compliqués et donc coûteux, et ce coût tend à être proportionnel (ou pire) avec la taille du document. Il est donc intéressant d'essayer de ne garder que les parties potentiellement pertinentes étant donné la question, ce que nous appellons une décomposition en passages.

Enfin, une fois ces passages obtenus, il est temps d'extraire les réponses possibles. À partir du moment où les différents passages obtenus sont classés par un ordre de pertinence, il est là encore possible de contrôler le temps CPU utilisé en limitant le nombre de candidats à examiner.

En résumé, ce nouveau système QA peut être vu comme constitué de trois parties :
– Une abstraction de la question via la construction d'un *Descripteur De Recherche*.
– Une *indexation*, synthétisant les informations pertinentes des documents analysés pour permettre ensuite une recherche efficace.
– Un filtrage multi-niveaux de la base de documents, commençant par une *sélection de documents*, suivi d'une *extraction de passages* et enfin d'une *extraction de candidats réponse*. Chaque étape du filtre nécessite des calculs plus compliqués que la précédente, mais travaille sur moins de données.
Conceptuellement, les étapes de sélection de documents et d'extraction de passages peuvent être considérées comme la partie *Recherche d'Information* au sens traditionnel du terme, la dernière étape représentant plus spécifiquement l'aspect *Question-Réponse*. De plus l'indexation est au service de la recherche, et en particulier de la sélection de documents. Elle sera donc présentée après la méthodologie de sélection de documents.

8.2 Représentation de la recherche

Le rôle du Descripteur De Recherche (DDR) est de représenter de façon synthétique la nature de la recherche à effectuer et des réponses attendues. Il se compose de trois parties :
– Les éléments considérés pertinents à trouver dans les documents

– Les types possibles de réponse attendue
– Des paramètres de tuning

- Élément critique
  - 1,0 *pers* identité(Hans Krasa)
  - 0,2 *pers* expansion(Hans Krasa)
- Élément secondaire
  - 1,0 *action* identité(assassiné)
  - 0,7 *action* lemme(assassiné)
  - 0,5 *action* synonyme(assassiné)
  - 0,5 *subs* verbe_subs(assassiné)
- Types de réponse
  - 1,0 *date_complete*
  - 0,9 *mois_annee jour_mois heure*
  - 0,7 *annee*

FIG. 8.2 – Exemple de Descripteur De Recherche pour la requête *Quand a été assassiné Hans Krasa ?*

La figure 8.2 montre un exemple d'un tel descripteur. La requête associée, tirée de l'évaluation QAst 2008, est *Quand a été assassiné Hans Krasa ?*. Le premier élément, la personne *Hans Krasa*, est considéré comme critique, c'est à dire qu'il doit se situer à proximité de tout candidat réponse examiné. Il peut apparaître soit sous sa forme *Hans Krasa*, identique à la question, ce qui est indiqué par *identité*, soit sous une variante, *expansion*, où seul le prénom ou le nom est présent, mais avec un poids plus faible (0,2 au lieu de 1). Le second élément, le verbe, *action* dans nos tags, est secondaire, indiquant que sa présence est souhaitée et prise en compte au niveau des scores mais n'est pas strictement indispendable. Il peut être rencontré tel quel (*identité*), sous une autre conjugaison du même verbe (*lemme*), sous la forme d'un verbe synonyme (*synonyme*, par exemple tuer) ou en tant que nom suite à une transformation morphosyntaxique de verbe en substantif (*verbe_subs*, par exemple assassinat). Le DDR permet ainsi d'appliquer et de pondérer a priori un certain nombre de formes d'*expansion de requêtes*. Enfin les types de réponse attendus sont présents avec des poids associés, le type le plus complet étant favorisé. Les paramètres de tuning n'ont pas été mis dans l'exemple.

La requête a la forme d'une question analysée et optionnellement d'un ensemble d'éléments complémentaires extraits de l'historique de l'interaction choisis par le gestionnaire de dialogue. La première étape de la construction du DDR est la *classification de la question*. Les éléments de la requête, en particulier, mais pas uniquement, les mots de question (qui, où, quand...) sont utilisés dans un premier classifieur à base de règles pour obtenir un type général attendu pour la réponse tel que *lieu*, *personne*, *nombre*, *organisation*... À chacune de ces classes est associé un ensemble de valeurs de tuning dont l'utilisation est présentée dans la suite du document et qui ont été obtenues par essais systématiques sur un corpus de développement. Une fois la classe de la question obtenue, un second niveau de classification permet d'obtenir l'ensemble des types de réponse attendus et les poids associés.

La sélection des éléments pertinents est algorithmique et s'appuie sur un ensemble de listes de types ayant des propriétés spécifiques. La première étape est une transformation du résultat de l'analyse

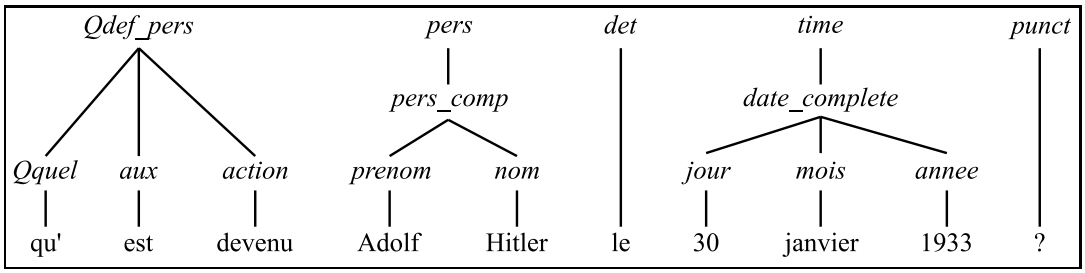


FIG. 8.3 – Analyse de *qu' est devenu Adolf Hitler le 30 janvier 1933 ?*

- *action*(devenu)
- *pers*(*prenom*(Adolf) *nom*(Hitler))
- *time*(*date_complete*(*jour*(30) *mois*(janvier) *annee*(1933)))

FIG. 8.4 – Résultat du filtrage initial des entités

- *action*(devenu)
- *pers*(Adolf Hitler)
- *time*(*date_complete*(*jour*(30) *mois*(janvier) *annee*(1933)))

FIG. 8.5 – Résultat de l'élagage des mini-arbres

de la question (telle que figure 8.3) en un vecteur d'entités qui sont dans notre cas les mini-arbres individuels contruits par l'analyse. À ce vecteur sont ajoutées les entités supplémentaires fournies par le gestionnaire de l'interaction en fonction de l'historique du dialogue s'il y en a. C'est cette décomposition en entités individuelles, qui nous donne la possibilité d'en ajouter, qui fournit la flexibilité de l'entrée nécessaire pour les systèmes interactifs. De ces entités ne sont ensuite conservées que celles de types considérés intéressants (figure 8.4). Il s'agit en l'occurrence des entités nommées générales et spécifiques, de certaines des entités linguistiques et d'un certain nombre d'entités thématiques. Puis ces mini-arbres sont élagués en fonction de plusieurs critères :

- Certains types (comme nom, prénom) ne sont acceptables qu'à la racine d'un mini-arbre et pas dans ses descendants d'où ils sont supprimés.
- Certains types (comme Tnaiss qui regroupe toutes les expressions parlant de la naissance) ne sont intéressants que pour le type lui-même et la valeur associée doit être supprimée.
- Certains types (comme les titres d'ouvrages) doivent être trouvés tels quels et non comme une combinaison de leurs sous-parties et donnent ainsi lieu à une suppression des descendants.
- Certains types (comme Aauteur qui regroupe les verbes et noms indiquant une création d'œuvre) sont *réducteurs*, dans le sens où s'ils sont présents les autres types associés à la même valeur doivent être supprimés (généralement des types linguistiques généraux comme nom ou verbe) ainsi que leurs descendants.

- Élément critique
  - *pers* Adolf Hitler
- Élément critique
  - *time* 30 janvier 1933
  - *date_complete* 30 janvier 1933
    - Sous-élément critique
      - *jour* 30
    - Sous-élément critique
      - *mois* janvier
    - Sous-élément critique
      - *annee* 1933
- Élément secondaire
  - *action* devenu
- Types de réponse
  - 1,0 *pers_act*
  - 1,0 *pers_fonct*
  - 0,5 *fonctions*

FIG. 8.6 – Descripteur De Recherche brut

Certains types font d'ailleurs partie de plusieurs de ces catégories, comme Tnaiss qui est à la fois suppresseur de valeur et réducteur. Une fois tous ces filtrages effectués (figure 8.5), nous avons les entités de base pour la construction du DDR. Il faut ensuite choisir celles qui seront critiques et celles qui seront secondaires. La méthode est simple : sont critiques toutes les entités nommées et les entités linguistiques et thématiques considérées les plus pertinentes (mots composés et substantifs par exemple, mais pas les verbes) ainsi que leurs descendants et le reste est secondaire. Si aucune entité ne répond au critère pour être critique, alors la plus importante est marquée critique. Cette classification d'importance est faite d'après un classement relatif des types d'entités linguistiques et thématiques et en second ordre la distance par rapport au début de la question. Nous avons à ce niveau le DDR brut (figure 8.6).

La dernière étape consiste à étendre pour pouvoir prendre en compte des alternatives d'écriture et de typage des entités. C'est l'étape d'expansion, constituée d'un ensemble de règles ajoutant ces variantes quand c'est approprié (figure 8.7). Par exemple, une de ces règles rajoute sur les entités de type *organisation* la possibilité de les trouver en tant que lieu avec un poids de 0,5. Une autre autorise les noms composés à être trouvés sous forme lemmatisée. Les transformations (lemmatisation, transformations morphosyntaxiques, etc) seront décrites au chapitre suivant.

Nous avons vu que tous les choix faits dans cette génération de Descripteurs De Recherche sont algorithmiques ou basés sur des règles. Il n'existe pas à l'heure actuelle de corpus de questions et réponses associés disponibles suffisamment grands pour envisager des approches stochastiques robustes. Les listes de types, poids, choix d'alternatives doivent ainsi être choisis avec un mélange d'expertise et

- Élément critique
  - 1,0 *pers* identité(Adolf Hitler)
  - 0,05 *np* identité(Adolf Hitler)
  - 0,7 *pers* lemme_simple(Adolf Hitler)
  - 0,015 *np* expansion(Adolf Hitler)
- Élément critique
  - 1,0 *time* identité(30 janvier 1933)
  - 1,0 *date_complete* identité(30 janvier 1933)
  - 0,6 *time* expansion(30 janvier 1933)
  - 0,7 *time* lemme(30 janvier 1933)
  - 0,7 *date_complete* lemme(30 janvier 1933)
  - 0,42 *time* expansion_lemme(30 janvier 1933)
    - Sous-élément critique
      - 1,0 *jour* identité(30)
      - 0,7 *jour* lemme(30)
    - Sous-élément critique
      - 1,0 *mois* identité(janvier)
      - 0,7 *mois* lemme(janvier)
    - Sous-élément critique
      - 1,0 *annee* identité(1933)
      - 0,7 *annee* lemme(1933)
- Élément secondaire
  - 1,0 *action* identité(devenu)
  - 1,0 *subs* verbe_subs(devenu)
  - 0,7 *action* lemme(devenu)
- Types de réponse
  - 1,0 *pers_act*
  - 1,0 *pers_fonct*
  - 0,5 *fonctions*

FIG. 8.7 – Descripteur De Recherche étendu

d'expérimentation sur les quelques données de développement disponibles.

8.3 Les transformations

Une des difficultés de la recherche d'informations est le grand nombre de formes différentes sous laquelle la même information peut apparaître. Par exemple les deux phrases *Barack Hussein Obama naît le 4 août 1961 à Hawaii.* et *Barack Hussein Obama II, né le 4 août 1961 à Honolulu, dans l'État d'Hawaii, est le 44e et actuel président des États-Unis d'Amérique.* donnent la même information

quant à la date de naissance de ce président mais sous des formes subtilement différentes. Le verbe *naître* est à un temps différent, et le nom complet contient un *II* supplémentaire dans un des cas. Les types de variation sont nombreux. On peut citer :
– Problèmes typographiques et orthographiques, et particulièrement des problèmes de majuscules ou d'accents
– Déclinaisons (singulier/pluriel, conjugaisons, etc)
– Variantes morphosyntaxiques (ex : fermer–fermeture)
– Synonymie, hypéronymie, hyponymie
Prendre en compte ces sources de variations a depuis toujours été un problème. Les problèmes de majuscules et d'accents peuvent être supprimés en ne les prenant pas en compte dans les comparaisons. Les déclinaisons sont généralement regroupées via un processus de *lemmatisation* qui supprime la terminaison variable des mots pour ne laisser que la racine [Sparck Jones & Willett 1997] ou remplace le mot par son lemme, en général à partir d'une table telle que présente dans le dictionnaire DELAS [Courtois 1990]. Les variantes morphosyntaxiques sont en partie traitables de la même manière ou encore avec des outils plus poussés tels que Fastr [Jacquemin 1996]. Les problèmes de synonymie, hyponymie et hypéronymie sont bien plus difficiles à gérer en l'absence d'une ressource telle que WordNet. Cependant des dictionnaires peuvent aider.

Mais avoir lemmatiseurs et dictionnaires ne suffit pas. Il faut encore décider comment les intégrer dans la recherche d'informations. La méthode la plus simple, et la plus souvent employée, est de transformer les documents avant indexation vers une forme canonique (lemmatisée ou même encore les mots remplacés par leurs têtes de dérivation). Les mots pertinents de la question sont transformés de la même façon permettant ainsi de trouver plus de documents potentiellement pertinents. Cependant toutes ces transformations font parfois des erreurs, regroupant des termes qui ne devraient pas l'être, ou simplement s'écartant trop du sens initial étant donné le contexte du document (un problème courant avec WordNet, où il est parfois difficile de savoir où s'arrêter dans le suivi des liens sémantiques). Nous avons donc décidé d'une procédure alternative : les transformations possibles sont ajoutées au Descripteur De Recherche avec des poids associés, laissant à l'expert humain la maîtrise du niveau de confiance à leur accorder. Les prendre en compte au niveau du système pose de plus grandes difficultés, et nous détaillerons les solutions proposées dans la section 8.5.

Reste le problème de la représentation de la construction de ces alternatives. Il serait possible de créer un algorithme spécialisé pour chacun des types de variation mais il semble plus intéressant, pour faciliter l'expérimentation, de les unifier autant que possible. Nous avons décomposé ces transformations en une série de transformations élémentaires :
– Suppression des accents.
– Suppression des majuscules.
– Application d'un dictionnaire transformant un mot en un mot.
– Application d'un dictionnaire transformant un mot en plusieurs mots possibles *sans* l'appliquer aux documents.
– Test d'inclusion.
Comparer la valeur d'un élément venant de la question à celui venant d'un document consiste à appliquer la chaîne de transformations aux deux et comparer les résultats. La suppression des accents

et majuscules permet de prendre en compte des problèmes typographiques courants. L'application de dictionnaires mot-mot couvre les cas de lemmatisation (mange → manger) et, à condition de constituer les listes, les cas d'erreurs d'orthographe les plus courants (ethymologie → etymologie) ou de simple variantes d'écriture correctes (tsar → tzar). L'application asymétrique de dictionnaires 1-vers-n permet de traiter les cas de synonymie (abstinence → privation, ascétisme, jeûne, diète) sans aller "trop loin" dans les liens. Par exemple *abdiquer* à pour synonyme (entre autres) *renoncer*, de même que *s'abstenir*. Cependant considérer *abdiquer* et *s'abstenir* en relation de synonymie, ce qu'une application symmétrique impliquerait, serait incorrect. Les relations d'hypéronymie et d'hyponymie (antidépresseur ↔ médicament) peuvent être utilisées de la même façon. Le test d'inclusion, enfin, vérifie si la valeur venant de la question est incluse dans celle venant du document ou vice-versa. Il est adapté aux comparaisons de noms de personnes et permet de retrouver *Barak Obama* quand l'on n'a que *Obama* ou l'inverse.

Définir des chaînes de transformations avec des dictionnaires bien choisis permet ainsi de retrouver de nombreuses variantes des élémentes de la question et d'y associer des indices de confiance a priori.

8.4 Sélection et classement des documents

Une fois la question analysée, l'étape suivante est la *Recherche d'Informations*, dont le but fondamental est d'extraire de la base de documents des sous-parties pertinentes pour répondre à la question. Une grande partie de la performance finale du système est déterminée à ce niveau. Trop peu de sous-parties résultera en un système probablement rapide mais qui aura rarement les informations nécessaires pour répondre. Trop ralentira le système qui aura cependant plus de chances de répondre. Obtenir trop de texte, en plus d'un effet délétère sur la vitesse, noiera le système dans les candidats réponse possibles, diminuant sa capacité à extraire les réponses correctes. C'est donc, comme bien souvent, une question d'équilibre entre précision et rappel de l'extraction.

Deux options sont possibles : utiliser un système de recherche d'informations existant et disponible, tel que Lucene [Apache 2007], ou en créer un spécialisé. Utiliser un système existant impose de suivre ses contraintes, en particulier des recherches sur des mots simples et des formes de transformation en général limités à une lemmatisation. En conséquence, pour pouvoir utiliser nos Descripteurs De Recherche comme clé primaire de sélection, nous avons préféré développer nos propres approches.

Traditionnellement, les systèmes de recherche d'informations retournent une liste de *documents*. Mais la définition de document est très variable. Il peut s'agir d'un des fichiers de la collection, ou d'une sous-partie d'un fichier pré-découpé au moment de l'indexation. Prendre une sous-partie permet d'avoir moins de données à traiter et potentiellement une meilleure précision. Cependant un découpage a priori n'est pas aussi précis qu'un découpage fait en fonction du contenu de la question. De plus, dans l'état actuel des performances des disques durs, le temps de lecture d'un fichier est dominé par le temps de latence : lire un fichier complet, s'il est de taille raisonnable, ne prend pas un temps significativement plus grand qu'en lire une partie. Nous avons donc décidé de procéder en

deux temps : une sélection de documents suivie par une sélection de passages pertinents.

```
Calcul du compte net pour un nœud
function calcule_compte_net(noeud, cc)
        Trie les lignes par ordre inverse de poids
        sort(noeud.lignes[], poids, <)
        for l in noeud.lignes[] do
                l.compte_net = max(0, l.compte_brut - cc)
                cc = cc + l.compte_net
        end
        Propage les comptes sur les dérivations
        for n in noeud.derivations[] do
                calcule_compte_net(n, cc)
        end
end

Calcul du compte net pour les nœuds du haut
for n in ddr.noeud_haut[] do
        calcule_compte_net(n, 0)
end
```

FIG. 8.8 – Algorithme de calcul des comptes nets pour les lignes du DDR à partir des comptes bruts et de la structure arborescente.

La question à se poser est donc comment choisir les documents les plus pertinents de la collection. L'approche que nous avons choisie est de définir un score calculé par document en fonction du DDR, et ensuite de choisir les documents avec les meilleurs scores. Une idée simple pour construire un tel score est de suivre la structure arborescente du DDR. On part des comptes d'occurrences des différents éléments du DDR. Le score d'un nœud individuel est la somme des nombres d'occurrences des lignes individuelles pondérées par leurs poids plus le score de ses dérivations. Le score d'une conjonction d'éléments, ou en d'autre termes une combinaison de nœuds du même niveau, se fait par une moyenne géométrique des scores individuels. L'avantage de la moyenne géométrique est de se rendre indépendant des fréquences moyennes différentes des divers éléments. Cependant, l'analyse produit des annotations hiérarchiques et imbriquées. Il en résulte que dans certains cas, la présence d'une variante d'une entrée du DDR implique la présence des autres variantes. Par exemple, si on a (lieu, France) alors on aura (pays, France) puisque pays est une sous-catégorie de lieu. Il semble préférable d'éviter de compter plusieurs fois le même élément. Donc la première étape consiste à calculer un compte net qui prend en compte ces répétitions en suivant l'algorithme figure 8.8.

Une fois les comptes nets obtenus, chaque nœud reçoit comme score associé à ses lignes la somme de ses comptes nets pondérés par les poids. De plus les nœuds secondaires ont 1 ajouté à leur score.

```
Calcul du score pour les lignes d'un nœud
function calcul_score_lignes(noeud)
        score = 0
        for l in noeud.lignes[] do
                score = score + l.poids * l.compte_net
        end
        return score
end

Calcul du score pour un ensemble de nœuds de même niveau
function calcul_score_groupe(noeuds[])
        score = 1
        for n in noeuds[] do
                score = score * calcul_score_noeud(n)
        end
        Moyenne géométrique
        score = pow(score, 1/noeuds.size())
        return score
end

Calcul du score complet d'un nœud
function calcul_score_noeud(noeud)
        score = calcul_score_lignes(noeud)
        if(noeud.derivations) score = score + calcul_score_groupe(noeud.derivations)
        if(noeud.secondaire) score = score + 1
        return score
end

Calcul du score d'un document
doc.score = calcul_score_groupe(ddr.noeud_haut)
```

FIG. 8.9 – Calcul d'un score de document à partir des comptes nets

L'étape finale remonte les scores jusqu'à la racine :
- Chaque nœud sans descendant a pour score total son score de lignes.
- Chaque nœud avec descendant a pour score total la somme de son score de lignes et de la moyenne géométrique des scores de ses descendants directs.

Le score final du document est le score d'un nœud racine virtuel ayant comme dérivation tous les nœuds de haut niveau. L'algorithme complet est détaillé figure 8.9. Un exemple de calcul complet est donné figure 8.10.

Ligne du DDR	Comptes		Score	Score nœuds		
	Brut	Net	Lignes	Lignes	Dériv.	Total
1,000 identité *pers* Adolf Hitler	5	5	5,000	5,000		5,000
0,700 lemme_simple *pers* Adolf Hitler	5	0	0,000			
0,050 identité *np* Adolf Hitler	0	0	0,000			
0,015 expansion *np* Adolf Hitler	0	0	0,000			
1,000 identité *time* 30 janvier 1933	2	2	2,000	3,800	3,557	7,357
1,000 identité *date_compl.* 30 janvier 1933	2	0	0,000			
0,700 lemme *time* 30 janvier 1933	2	0	0,000			
0,700 lemme *date_compl.* 30 janvier 1933	2	0	0,000			
0,600 expansion *time* 30 janvier 1933	5	3	1,800			
0,420 exp._lemme *time* 30 janvier 1933	5	0	0,000			
1,0 identité *jour* 30	6	1	1,000	1,000		1,000
0,7 lemme *jour* 30	6	0	0,000			
1,0 identité *mois* janvier	8	3	3,000	3,000		3,000
0,7 lemme *mois* janvier	8	0	0,000			
1,0 identité *annee* 1933	20	15	15,000	15,000		15,000
0,7 lemme *annee* 1933	20	0	0,000			
1,000 identité *action* devenu	0	0	0,000	9,800		10,800
1,000 verbe_subs *subs* devenu	0	0	0,000			
0,700 lemme *action* devenu	14	14	9,800			
					Score final	7,351

FIG. 8.10 – Calcul complet d'un score de document

Les n documents avec le meilleur score sont ainsi sélectionnés. Ce nombre est le premier facteur de contrôle de la vitesse du système. Mais calculer ce score demande à pouvoir efficacement trouver pour chaque ligne, et donc triplet (type, valeur, transformation), du DDR le compte d'occurrences associé dans chaque document. C'est le rôle de l'indexation.

8.5 Indexation des documents

Le but de l'indexation est de synthétiser et réorganiser les informations trouvées dans les documents de façon à permettre aux étapes de recherche d'informations et d'extraction de réponse de s'exécuter le plus rapidement possible. Dans notre cas cela demande de :

– être capable de trouver les éléments du DDR dans un document donné en tenant compte des transformations
– réciproquement, être capable de dire quels documents contiennent des éléments du DDR
– être capable de les compter sans avoir à lire les documents

Nous avons construit la réponse à ces besoins autour de plusieurs structures élémentaires. La plus centrale est une table numérotant toutes les paires uniques (type, valeur) trouvées dans les documents. Nos plus gros index à ce jour comptent environ 26 millions de paires différentes, ce qui n'est pas si grand comparé aux capacités mémoires actuelles. En conséquence cette numérotation nous fournit des identifiants numériques permettant non seulement de faire des comparaisons d'égalité beaucoup plus efficaces qu'en comparant des chaînes de caractères, mais aussi de construire des vecteurs indexés par ces nombres. Une simple table de hachage permet de retrouver efficacement l'indentifiant correspondant à une paire (type, valeur) donnée.

Cependant ces identifiants ne peuvent être directement appliqués aux DDR. En effet, il est nécessaire de prendre en compte les transformations. Ceci se fait à travers une *instanciation* du DDR, qui consiste à trouver l'ensemble des identifiants de paires (type, valeur) des documents satisfaisant chacun des triplets (transformation, type, valeur) du DDR. Pour cela les transformations sont compilées sous la forme d'une suite de transformations élémentaires sur les valeurs suivies d'une opération finale de recherche d'identifiants. Les transformations élémentaires correspondent à celles choisies par l'utilisateur (suppression des accents, des majuscules, application de dictionnaires 1-1 ou 1-n). La recherche finale peut être soit une recherche normale en simple comparaison d'égalité, soit une recherche sur test d'inclusion. Il est donc possible, en partant de l'inventaire de valeurs venant des documents de précalculer le résultat des transformations sur ces valeurs puis de les organiser pour permettre des recherches rapides (tables de hachage par type). Au moment de l'instanciation du DDR les mêmes transformations sont appliquées à ses valeurs et les recherches d'identifiants sont faites dans les structures appropriées. Cette recherche peut bien évidemment donner comme résultat plusieurs identifiants différents. De plus l'utilisation de dictionnaires donnant plusieurs mots associés à un seul, comme les synonymes, peut produire plusieurs valeurs différentes. Plusieurs recherches devront dans ce cas être effectuées et les résultats regroupés. Un exemple d'une telle instanciation est donné figure 8.11.

Nous avons donc un identifiant numérique unique pour chaque paire (type, valeur) présente dans les documents et la capacité pour chaque triplet (transformation, type, valeur) de trouver l'ensemble des identifiants lui correspondant. Calculer les scores des documents ne demande plus donc que d'être capable de donner pour un identifiant donné le compte d'occurrences dans chacun des documents. Cela se fait par l'intermédiaire d'un *index inversé*, qui est simplement un tableau, indexé par les identifiants, de listes de paires (identifiant, compte d'occurrences). Ces listes sont ordonnées, permettant de les rapprocher en temps linéraire avec leur taille, et les calculs de scores se font en manipulant de telles listes.

Pour rendre plus efficace les étapes suivantes, présentées dans les sections suivantes, une transformation des documents est faite vers un format que l'on pourrait qualifier de numérique plat. Chaque document devient un tableau de structures de taille fixe, contenant un identifiant de paire (type, valeur), des pointeurs vers l'élément suivant et précédent au même niveau dans les arbres si ils existent, et des pointeurs (implicites) vers le premier de ses descendants et son père, là encore si ils existent. Les pointeurs précédents/suivants sont réutilisés pour lier les éléments de haut niveau des phrases. Cette transformation, associée à des tables de positions de début de ligne dans le tableau de struc-

Ligne du DDR	**Id**	**Élément trouvé**
identité *pers* Adolf Hitler	137021	*pers* Adolf Hitler
identité *np* Adolf Hitler		
expansion *np* Adolf Hitler	600006	*np* Hitler
	5209329	*np* Adolf
lemme_simple *pers* Adolf Hitler	137021	*pers* Adolf Hitler
	13402536	*pers* Adolf HITLER
identité *jour* 30	1638	*jour* 30
lemme *jour* 30	1638	*jour* 30
identité *mois* janvier	710	*mois* janvier
lemme *mois* janvier	710	*mois* janvier
identité *annee* 1933	139776	*annee* 1933
lemme *annee* 1933	139776	*annee* 1933
identité *time* 30 janvier 1933	865131	*time* 30 janvier 1933
identité *date_complete* 30 janvier 1933	865132	*date_complete* 30 janvier 1933
expansion *time* 30 janvier 1933	1637	*time* 30
	5644	*time* janvier
	88431	*time* 30 janvier
	139775	*time* 1933
	865131	*time* 30 janvier 1933
	3080208	*time* janvier 1933
lemme *time* 30 janvier 1933	865131	*time* 30 janvier 1933
lemme *date_complete* 30 janvier 1933	865132	*date_complete* 30 janvier 1933
expansion_lemme *time* 30 janvier 1933	1637	*time* 30
	5644	*time* janvier
	88431	*time* 30 janvier
	139775	*time* 1933
	865131	*time* 30 janvier 1933
	3080208	*time* janvier 1933
identité *action* devenu	7646	*action* devenu
verbe_subs *subs* devenu		
lemme *action* devenu	328	*action* devient
	1863	*action* deviennent
		(37 autres possibilités)

FIG. 8.11 – Élements des documents obtenus par instanciation du DDR de la figure 8.7.

tures, permet des recherches, comparaisons et déplacements très rapides dans les documents. Les mots non-typés sont perdus, mais la présence d'un type *mot inconnu* dans l'analyse assure qu'il n'en existe en pratique pas.

8.6 Sélection et classement des passages

Les documents sont choisis suivant le score défini section 8.4 évalué à partir des informations fournies par l'indexation décrite section 8.5 puis ils sont chargés en mémoire. L'étape suivante est alors de sélectionner des passages pertinents : des blocs de lignes ayant de fortes chances de contenir la réponse cherchée. Le but est double : diminuer la quantité de texte à traiter au moment de l'extraction de la réponse, et diminuer le bruit dû aux candidats réponse situés loin des éléments du DDR.

L'algorithme de sélection des passages s'appuie sur plusieurs concepts. Le premier est celui de *satisfaisabilité* du DDR. Étant donné un Descripteur De Recherche avec sa structure et ses éléments critiques et secondaires, un ensemble d'entités E venant d'un document le valide-t-il ? Un ensemble d'éléments de DDR est validé si tous les éléments critiques sont validés ou, si il n'y a pas d'élément critique, au moins un des éléments secondaires l'est. Un élément seul est validé si au moins un des triplets (transformation, type, valeur) qui le caractérise accepte une des entités de E ou alors, si il a une dérivation, ses éléments dérivés en tant qu'ensemble sont validés. Le DDR est bien évidemment validé si l'ensemble formé par ses éléments de haut niveau est validé. Fondamentalement, cette définition récursive veut simplement dire que les éléments critiques d'en haut doivent être tous présents, soit en tant que tels soit en tant que combinaison de leurs éléments dérivés.

Nous cherchons donc à obtenir des parties de document dont les entités sont capables de satisfaire le DDR. Pour découper ces blocs nous nous appuyons sur un concept de *distance d'influence* des entités. Découpant le document en lignes, correspondant en moyenne à une phrase ou l'équivalent dans un cas oral, nous considérons qu'une ligne satisfait le DDR si toutes les entités nécessaires sont présentes à une distance maximale de *range* lignes de celle observée. Cet ensemble de lignes satisfaisant le DDR permet d'obtenir un premier découpage en passages. Certains des blocs connexes obtenus peuvent cependant être un peu grands. Pour essayer de limiter cela nous tentons de subdiviser les blocs de taille en lignes supérieure à *size*. La méthode consiste à passer temporairement les éléments secondaires du DDR en critique, un par un, et à tester à nouveau la satisfaction du DDR jusqu'à obtenir des sous-blocs suffisamment petits ou qu'il n'y a plus d'élément à transformer. Cette phase fournit ainsi un second découpage en blocs. Ces blocs ont cependant un problème : ils ne contiennent pas toujours les entités ayant justifié leur création. Prenons le cas d'un DDR avec deux éléments critiques dont des entités les instanciant sont disposées sur deux lignes séparées. Si ces lignes sont légèrement plus éloignées que la distance d'influence certaines lignes situées au milieu seront conservées mais les deux lignes contenant les entités ne le seront pas. Pour s'assurer que ces entités sont bien présentes dans les blocs il faut donc dans une dernière passe élargir leur frontières au besoin pour les récupérer.

Algorithmiquement, nous voyons qu'une représentation des entités présentes dans une ligne permettant de tenir efficacement compte des distances d'influences et permettant de calculer rapidement si le DDR est satisfait est indispensable. Une observation fondamentale est que pour calculer une satisfaction de DDR les entités spécifiques rencontrées ou leur nombre d'occurrences dans les lignes individuelles ne sont pas importantes, seuls les nœuds auxquelles elles appartiennent suite à l'instanciation sont utiles. L'ensemble des entités présentes dans une ligne donnée peut ainsi être réduit

```
Préconstruction de la table nœud/entité

– Allouer un tableau d'entiers emask de taille égale au nombre d'entités différentes dans
  les documents initialisé à 0.
– Numéroter les nœuds du DDR en commençant a 0.
for n in ddr.noeuds[] do
        mask = 1 << n.numero
        for l in n.lignes do
                for e in l.entites do
                        emask[e.identifiant] = emask[e.identifiant] | mask
                end
        end
end

Construction des vecteurs de bits de nœuds par ligne

– Allouer deux tableaux d'entiers lmask et gmask de taille égale au nombre de lignes du
  document initialisés à 0.
for l in document.lignes[] do
        for e in l.entites do
                lmask[l.numero] = lmask[l.numero] | emask[e.id]
        end
end
for lnum in 0..document.lignes.size()-1 do
        for ldest in max(0, lnum-range)..min(document.lignes.size()-1, lnum+range) do
                gmask[ldest] = gmask[ldest] | lmask[lnum]
        end
end
Les vecteurs de bits, influence incluse, sont disponibles dans gmask
```

FIG. 8.12 – Algorithme de calcul des vecteurs de bits représentant les présences d'entités par ligne

à un vecteur de bits, un bit par nœud du DDR, indiquant quels nœuds elle valide. Une combinaison de présence d'entités se calcule alors par un simple ou binaire. De plus ces nœuds sont peu nombreux, une quinzaine au grand maximum, permettant de stocker ce vecteur de bits dans un entier du processeur. Cela, combiné aux identifieurs numériques d'entités, permet de construire un algorithme efficace pour calculer l'ensemble des entités, vues en tant que nœud, influençant chaque ligne. Cet algorithme est décrit figure 8.12. La liste des entités validant chaque ligne est obtenue par l'instanciation du DDR décrite section 8.5. L'ensemble des entités présentes par ligne est obtenue efficacement grâce au format numérique plat décrit dans la même section.

Élément du DDR	Numéro	Représentation binaire
pers Adolf Hitler	0	□□□□□■
time 30 janvier 1933	1	□□□□■□
jour 30	2	□□□■□□
mois janvier	3	□□■□□□
annee 1933	4	□■□□□□
action devenu	5	■□□□□□

Expression de validation : $b_0 \wedge (b_1 \vee (b_2 \wedge b_3 \wedge b_4))$

FIG. 8.13 – Numérotation des éléments du DDR et expression logique de statisfiabilité associée

```
Extension d'une frontière
Les paramètres sont la position de la frontière et la direction de déplacement, le résultat la
nouvelle position.
lmask et gmask viennent de l'algorithme figure 8.12
function replace_frontiere(line, direction)
        Récupère l'ensemble des influences agissant sur la frontière
        cur_mask = gmask[line]
        while(cur_mask != 0 && line > 0 && line < doc.line_count-1) do
                Supprime les influences justifiées par les entités de la ligne courante.
                cur_mask = cur_mask & ∼lmask[line]
                Supprime les influences qui n'existent plus dans la ligne suivante,
                et donc venant de l'autre coté.
                cur_mask = cur_mask & gmask[line+direction]
                Si il en reste alors on déplace la frontière et on continue.
                if(cur_mask != 0) line = line + direction
        end
        return line
end
```

FIG. 8.14 – Algorithme d'expansion des frontières des blocs.

Une fois ces vecteurs obtenus il faut ensuite être capable de savoir quelles lignes satisfont le DDR. Cela peut être ramené à un test logique sur le vecteur de bits. En effet :

– Un groupe de nœuds contenant un ou plusieurs éléments critiques est satisfait si tous ses nœuds critiques sont statisfaits (*et* logique).
– Un groupe de nœuds ne contenant que des éléments secondaires est satisfait si au moins un des nœuds est statisfait (*ou* logique).
– Un nœud individuel est satisfait si les entités présentes le valident (test de bit dans le vecteur) ou ses dérivations sont satisfaites en tant que groupe (*ou* logique entre les deux résultats).
– Le DDR lui-même est satisfait si le groupe des nœuds de plus haut niveau est satisfait.

Il est ainsi facile de construire récursivement une expression logique effectuant le test. La figure 8.13

Ligne	Entités	lmask	gmask	DDR	cur_mask	Bloc
0	...	□□□□□□	□□□□□□	□	□□□□□□	□
1	...	□□□□□□	□■□□□□	□	□□□□□□	□
2	...	□□□□□□	□■□□□□	□	□□□□□□	□
3	annee(1933)	□■□□□□	□■□□□□	□	□■□□□□	■
4	...	□□□□□□	□■■■□□	□	□■■■□□	■
5	...	□□□□□□	□■■■□■	■	□■■■□■	■
6	jour(30) mois(janvier)	□□■■□□	□□■■□■	□	□□■■□■	■
7	pers(Adolf Hitler)	□□□□□■	□□■■□■	□	□□□□□■	■
8	...	□□□□□□	□□■■□■	□	□□□□□□	□
9	...	□□□□□□	□□□□□■	□	□□□□□□	□
10	...	□□□□□□	□□□□□□	□	□□□□□□	□

FIG. 8.15 – Exemple d'extraction de bloc. *lmask* indique les nœuds du DDR directement validés par les entités présentes. *gmask* propage ces nœuds sur la distance d'influence (*range*=2). *DDR* indique si gmask satisfait le DDR, nous donnant le bloc initial. *cur_mask* indique l'ensemble d'influence restant à justifier suivant l'algorithme d'expansion de frontière. *Bloc* indique les lignes finalement conservées.

donne un exemple de numérotation des éléments du DDR et l'expression de satisfiabilité associée. La première construction des blocs ainsi que leur subdivision si nécessaire par passage d'éléments de secondaire à critique ne pose plus alors de difficulté algorithmique particulière. Reste l'expansion finale pour retrouver les entités manquantes. L'algorithme de cette passe s'appuie sur les deux tableaux *lmask* et *gmask* qui représentent les entités des lignes et leur influence. Les frontières vont alors être déplacées jusqu'à ce que toutes les sources des influences les atteignant aient été indentifiées ou que les entitées concernées se trouvent vers l'intérieur du bloc. L'algorithme exact est donné figure 8.14. Un exemple de déroulement complet de ces algorithmes est figure 8.15.

Les passages obtenus, il suffit alors de leur donner un score. Nous avons choisi de le calculer de la même manière que les scores de documents à partir des comptes d'occurrences des entités individuelles. Seule la procédure de calcul des comptes nets est changée. En effet avoir le texte des passages permet de mesurer exactement les inclusions des entités les unes dans les autres. Chaque entité validant plusieurs lignes dans les nœuds du DDR n'est ainsi comptée qu'une seule fois, pour la ligne qui a le plus grand poids, et seulement si elle n'est pas incluse dans une autre entité validatrice. Cependant la relative petite taille des passages rend ces scores un peu abrupts. Ils ont besoin d'être lissés avec les scores de document. Nous avons choisi une moyenne géométrique pondérée par un coefficient w pour cela :

$$S_{\text{lisse}} = S_{\text{brut}}^{w} S_{\text{document}}^{1-w} \tag{8.1}$$

Les passages peuvent ensuite être examinés par ordre de score par l'étape suivante, l'extraction des réponses.

Cet ensemble d'algorithmes est contrôlé par 3 valeurs, *range*, *size* et *w*. Nous appellons ces valeurs

des *variables de tuning*. Leurs valeurs sont fixées par essais systématiques sur des données de développement, comme décrit section 8.8.

8.7 Extraction et classement des réponses

Une fois les passages obtenus et leurs scores calculés, il est temps d'examiner les candidats réponse. Est considéré comme réponse possible à la question posée toute entité présente dans un passage dont le type fait partie des types prédits pour la réponse. Sont cependant supprimées toutes les entités validant des lignes du DDR ou leurs descendants. En effet ces entités sont bien évidemment extrêmement favorisées par la recherche, et les questions qui contiennent leur propre réponse sont rares dans un besoin réel.

Nous avons décidé de construire un score se basant sur une mesure de distance entre le candidat et les entités validant le DDR situées dans le même passage. Cela nous rend un peu similaire à [Plamondon & Kosseim 2002] où un score de proximité est utilisé. Plus une entité validant le DDR est proche du candidat, plus elle ajoutera au score final. Ce genre de distance est souvent mesuré en nombre de mots, mais dans notre cas, nous pouvons faire mieux. En effet les entités de l'analyse forment des blocs de mots, pas très lointains de la notion de chunk, représentant chacun un objet ou concept spécifique. Il est donc plus intéressant de compter les entités de haut niveau traversées plutôt que les mots, permettant d'obtenir le même coût unitaire pour traverser *Sarkozy* ou *président de la République Française*. Nous définissons donc $d(e, a)$, la distance entre un candidat réponse et une entité validant une ligne du DDR. Mais nous ne voulons pas non plus que toutes les entités du passage ajoutent au score, seulement celles qui semblent les plus pertinentes vis-à-vis du candidat réponse. L'analyse ne permet cependant pas de savoir quelles entités sont en relation sémantique avec le candidat réponse. Une approximation simple pour choisir ces entités pertinentes est de prendre le sous-groupe maximisant le score final tout en évitant les redondances au niveau du DDR. Étant donné un ensemble E de paires (e, l) contenant une entité d'un passage et la ligne qu'elle valide (la même entité pouvant apparaître dans plusieurs paires). Un tel ensemble est considéré non-redondant si :

– Deux paires différentes de E ne valident pas une ligne du même nœud (redondance interne aux nœuds).
– Deux paires différentes de E ne valident pas deux nœuds dont l'un est descendant de l'autre (redondance structurelle).

On associe à chacune de ces paires (entité du passage, ligne du DDR) un score à partir de la distance entre l'entité et le candidat $d(e, a)$, le poids associé à la ligne du DDR $w(l)$ et une variable de tuning α . Le score pour E est la somme des scores individuels.

$$S(E, a) = \sum_{(e,l)\in E} \frac{w(l)}{(1 + d(e, a))^{\alpha}} \tag{8.2}$$

Il est nécessaire de rajouter 1 à la distance pour éviter des divisions par zéro. Pour tous ces ensembles E non-redondants possibles, nous choisissons celui donnant le meilleur score. Multiplier ce score par

le poids associé au type de a en tant que réponse nous donne un premier score individuel pour a :

$$S_1(a) = w(a) \max_E \sum_{(e,l)\in E} \frac{w(l)}{(1 + d(e,a))^\alpha} \tag{8.3}$$

Ce score brut n'est pas suffisant. En particulier il ne prend pas en compte l'adéquation du passage ou du document au thème de la question, et ne tient pas non plus compte de la redondance (même réponse à plusieurs endroits différents). Nous commençons par le lisser avec le score de passage S_p :

$$S_2(a) = S_1^{1-\gamma} S_p^\gamma \tag{8.4}$$

Pour tenir compte de la redondance, nous considérons que toutes les instances de la même paire (type, valeur) représente la même réponse r. Notant A_r l'ensemble des instances de candidat réponse ayant r comme (type, valeur), son score global primaire est la somme des scores individuels :

$$S_1(r) = \sum_{a\in A_r} S_2(a) \tag{8.5}$$

Mais cette addition favorise trop les entités fréquentes. Donc une dernière compensation est nécessaire pour obtenir le score final. Partant des comptes d'occurrences $C_d(r)$ de r dans les documents et $C_p(r)$ dans les passages, le score final est :

$$S(r) = \frac{S_1(r)}{C_d(r)^\beta C_p(r)^\delta} \tag{8.6}$$

β et δ sont là encore des variables de tuning. Regroupant tout, cela nous donne comme équation complète :

$$S(r) = \frac{\sum_{a\in A_r} (w(a) \max_{E_a} \sum_{(e,l)\in E_a} \frac{w(l)}{(1+d(e,a))^\alpha})^{1-\gamma} S_p(a)^\gamma}{C_d(r)^\beta C_p(r)^\delta} \tag{8.7}$$

En résumé les entités validant le DDR participent chacune au score à hauteur du poids de la ligne du DDR associée divisée par leur distance au candidat. Ce score est ensuite pondéré par le poids associé au type de réponse, puis lissé avec le score de passage. Les scores individuels des différentes instances de la même paire (type, valeur) sont additionnés, puis partiellement compensés par leurs comptes d'occurrences dans les documents et dans les passages.

En pratique, examiner la totalité des candidats réponse peut poser des problèmes de performances dans certains cas. En particulier des types très courants comme *substantif* sont acceptables en réponse pour les questions considérées vagues par l'analyse, ou simplement hors typologie précise. Du coup le nombre moyen de candidats réponse par ligne de passage est extrêmement variable d'une question à l'autre et la limitation sur le nombre de documents à charger est un contrôle insuffisant. Nous nous retrouvons à devoir choisir entre un écroulement des performances sur les questions "vagues" du point de vue de l'analyse et une incapacité de répondre sur les questions "précises" par manque de candidats. Nous avons donc un second niveau de contrôle qui est le *nombre maximal de candidats à examiner*. Les passages sont pris un par un dans l'ordre de leurs scores et les candidats présents

examinés. L'extraction s'arrête quand le nombre de candidats examinés atteint ou dépasse cette limite. Il est à noter que la limite n'est consultée que quand on passe d'un passage à un autre, ce qui rend l'implémentation légèrement plus simple et efficace sans perte de vitesse détectable.

D'un point de vue algorithmique calculer ces scores est plus simple qu'il n'y paraît au premier abord. La sélection de l'ensemble E d'entités validant des lignes du DDR optimal pour un candidat donné respecte le principe d'optimalité de Bellman [Press 1957] grâce à sa construction à partir d'additions et de maximum entre valeurs strictement positives et son indépendance entre nœuds. Il est donc possible d'appliquer la Programmation Dynamique, en calculant le score par remontée des feuilles aux éléments de haut niveau, sélectionnant la meilleure possibilité à chaque fois. La seule difficulté reste le calcul des comptes d'occurrence, mais ceux-ci sont disponibles dans l'index pour les documents et calculables au vol pour les passages.

8.8 Optimisation des paramètres de tuning

Nous avons vu que nos différents algorithmes s'appuient sur une série de variables dites de *tuning*. La sélection de passages s'appuie sur deux variables, *range* et *size*, pour diriger sa sélection de lignes des documents et sur la variable *w*, coefficient de lissage entre son score brut et celui du document. Le calcul du score des candidats s'appuie sur 4 variables de plus, α, γ, β et δ, agissant respectivement sur la prise en compte des distances entre les entités et le candidat, le lissage avec le score de passage et la prise en compte des comptes d'occurrences dans les documents d'un côté et les passages de l'autre. De plus deux variables contrôlent la vitesse globale du système, le *nombre maximal de documents à lire* et le *nombre maximal de candidats à étudier*. Ces deux dernières variables sont plus faciles à étudier et font l'objet du chapitre 14.

Modéliser ces 7 variables de tuning et leurs interactions est a priori très difficile et n'est, heureusement, pas indispensable. En effet tout ce que nous voulons c'est un ensemble de valeurs donnant de bons résultats. En l'absence de résultats quant à la convexité de l'espace de recherche, qui permettraient d'envisager des algorithmes plus évolués, ces valeurs peuvent être obtenues par essais systématiques sur un corpus de développement. Cela représente un nombre d'essais assez important, 1,2 million pour 4 paires (range, size) et toutes les valeurs de 0 à 1 par pas de 0,1 pour les autres sauf pour δ que nous faisons varier entre -1 et 1 ou encore 57 024 essais en choisissant un pas de 0,2. La structure du système permet cependant de factoriser une grande partie de ces essais. En effet pour une question donnée la sélection des documents est identique quelles que soient les valeurs. La sélection de passages varie, mais une fois celle-ci effectuée et les scores de passages calculés les candidats réponse examinés restent les mêmes. Les scores peuvent ainsi être calculés en même temps pour toutes les valeurs de α, β, γ et δ, accélérant grandement la recherche globale.

Cette systématisation et automatisation de l'optimisation de ces variables nous a permis de constater qu'il était intéressant de les varier d'une question à l'autre. Nous n'avons cependant pas de méthode capable de trouver des valeurs optimales pour chaque question individuelle, et quelques méthodes

de clustering automatique n'ont rien donné de pertinent, probablement dû à un manque de données d'apprentissage. Nous avons cependant constaté qu'utiliser le résultat du classifieur sélectionnant le type général de réponse attendu (section 8.2) fournit une classification des questions pertinente pour le choix des valeurs de tuning. Nous utilisons donc une série de valeurs par type général.

Chapitre 9

Autres types de question

Répondre à des questions factuelles est un problème intéressant, mais cela est loin de couvrir l'ensemble des questions qu'un utilisateur peut avoir envie de poser. Nous avons fait des travaux préliminaires pour traiter d'autres types de question. Aucune évaluation sérieuse n'en a encore été faite, mais ils ont l'interêt de montrer comment tout ou partie de l'approche proposée pour les questions factuelles peut être réutilisé en support d'une analyse adaptée pour traiter d'autres problèmes.

Le premier type de question est les *listes fermées*. Ces questions, telles que *Quels sont les cinq pays premiers producteurs de pétrole ?*, attendent une liste de plusieurs réponses et précisent le nombre attendu explicitement. Nous avons aussi décidé de nous limiter dans un premier temps aux cas où la liste réponse apparaît dans une courte partie d'un document, voulant éviter les difficultés liées à la fusion multi-documents. La méthode est alors simple : après une extension de l'analyse pour la rendre capable de détecter ce type de question et d'isoler le compte, un Descripteur De Recherches est construit de la manière habituelle. Le compte, n, est extrait indépendamment et ne fait pas partie des éléments du DDR. Les candidats réponse sont évalués, donnant pour chaque réponse possible un score global. Une seconde passe a alors lieu sur les passages avec une fenêtre glissante d'une taille en lignes décidée à l'avance. Chaque ensemble de candidats réponse contenu dans une telle fenêtre devient alors un candidat de réponse liste. Les listes candidates de plus de n éléments sont coupées à n en gardant seulement les éléments ayant le score global le plus élevé. Finalement chaque liste est évaluée en lui donnant comme score la somme des scores de ses éléments individuels. Comme d'habitude, celle avec le meilleur score est considérée la meilleure. Cette méthode a donné des résultats tout à fait raisonnables sur les données de développement et le test. Les échecs, particulièrement en anglais, étaient plutôt dus à des échecs d'analyse des éléments de la liste dans les documents, et donc pas fondamentalement causés par la demande de liste.

Deux types de questions que nous avons traitées de façon similaire sont les questions *pourquoi* et *comment* telles que *Pourquoi le ciel est-il bleu ?* ou *Comment peler les tomates ?*. La encore nous nous limitions à des réponses simples, devant tenir dans une phrase, ou ligne de document. Notre approche

a reposé sur la détection de marqueurs linguistiques du type de phrase que nous cherchions, tels que *parce que* ou *est le résultat de* pour les questions pourquoi ou encore *en + participe présent* pour les comment. Ces marqueurs se voient attribuer un type spécifique et ce sont eux qui sont recherchés en tant que candidat réponse. Le calcul de score est légèrement modifié : les candidats individuels sont considérés indépendants même s'ils ont la même valeur, il n'y a pas d'additions de scores pour obtenir un score global, et les comptes d'occurrence et scores de passage sont ignorés ($\beta = \delta = \gamma = 0$). Les autres variables de tuning (*range*, *size*, *w* et α) sont fixées de manière empirique. De ce score modifié ressort un meilleur candidat que les autres, et la réponse donnée est la phrase complète qui le contient. En l'occurrence nous extrayons les phrases *Le bleu du ciel est le résultat de la diffusion de la lumière solaire par les composants de l'atmosphère* et *Peler les tomates en les plongeant quelques secondes dans une casserole d'eau bouillante*, qui sont plutôt pertinentes.

Un autre type de question qui demande l'extraction de phrases est les questions *oui/non*, telles que *Est-ce que l'Afrique du sud fait partie des sièges permanents au conseil de sécurité des Nations Unies ?*. Cependant il n'y a pas de marqueurs linguistiques dans les documents associés aux réponses à ce type de question. Nous nous sommes donc appuyés sur l'extraction de passages pour extraire une phrase pertinente en forçant *range* a 0 et *size* à 1. Les quelques phrases extraites répondent bien souvent effectivement à la question. Cependant la réponse attendue est un oui ou un non, et estimer la valeur de ces phrases n'est pas facile. Notre méthode brutale consistant à détecter la présence d'une négation est nettement insuffisante, comme pour la phrase *Le Brésil, l'Inde et l'Afrique du Sud intensifieront leur campagne pour l'obtention de sièges permanents au conseil de sécurité des Nations Unies quand leurs dirigeants se rencontreront à Pretoria, la capitale sud-africaine*. En effet il n'y a pas de négation dans cette phrase alors qu'elle indique clairement que l'Afrique du Sud n'est pas encore membre permanent.

Le dernier type sur lequel nous avons fait des travaux préliminaires sont les questions de *définition*, comme *Qu'est-ce que le cabernet typique ?*. C'est là où nous nous sommes le plus éloignés des algorithmes présentés. Une analyse spécifique détecte les structures linguistiques caractéristiques d'une définition, *nom est-un expression-multi-mots* par exemple, et marque l'objet défini et la définition. Ces définitions détectées sont ensuite récupérées pour constituer une base de données. Cette base sert ensuite à répondre aux questions. Pour arriver à trouver les définitions même face à des variations d'écriture, le principe de transformation décrit chapitre 8.3 est utilisé sur la clé de recherche. Nos premières expériences montrent que ce principe de constitution de base de données est à approfondir. Si nous sommes capables d'en constituer sur divers types de question ou d'information avec une grande précision nous devenons capables de répondre à certaines questions très rapidement. Cela revient fondamentalement à construire des *bases de connaissances struturées* à partir des documents. Néanmoins il faut pouvoir évaluer la qualité de ces entrées, ce dont nous ne sommes pas encore capables.

Discussion

Nous avons présenté une approche générale et un ensemble d'algorithmes dont le but est de répondre au problème de la réponse à des questions factuelles simples sous deux contraintes fortes issues du domaine de l'interaction :
– documents en domaine ouvert aux formats libres
– flexibilité de l'entrée qui ne se limite pas à une simple question, des entités complémentaires pouvant être fournies par le gestionnaire de dialogue
– contrôle de la vitesse du système, pour assurer une bonne interactivité

Répondre à ces contraintes est passé par deux originalités. La première est que l'ensemble des algorithmes reposent sur une analyse unifiée multi-niveaux qui structure l'ensemble des textes et questions en entités individuelles ayant la forme d'arbres de types avec les mots originaux au niveau des feuilles. Ces entités sont utilisées à la fois comme clés de recherche, réponses potentielles et structuration générale du texte. La seconde originalité est la construction d'un *Descripteur De Recherche* (DDR), qui représente ce que le système a compris de la demande. Ce descripteur, abstrait mais compréhensible par un humain, permet de s'affranchir de la forme de l'entrée. Nous avons présenté un algorithme pour le générer à partir d'une phrase optionnellement accompagnée d'entités complémentaires, mais l'on peut imaginer que de tels descripteurs soient directement construits à partir de résultats de systèmes de raisonnement ou autres approches algorithmique. Essentiellement, le DDR peut être vu comme similaire à *select* dans une base de données, qui serait les documents dans lesquels les recherches sont effectuées. Nous envisageons d'ailleurs de détecter certains DDR caractéristiques de recherches en domaine spécifique pour les rediriger vers de vraies bases de données. Les recherches de noms d'acteurs ou de réalisateurs de films pourraient par exemple donner lieu à une requête sur une base en ligne telle que *IMDB* plutôt que dans une base de documents libres. Obtenir l'information est après tout le but et la recherche dans des documents libres n'est qu'un moyen. Notre concept de Descripteur De Recherches semble être un outil de communication intéressant entre un ou plusieurs systèmes de recherche d'informations au sens large et un système tel qu'un système de dialogue intéressé uniquement par les résultats.

La chaîne de traitement cherche donc à extraire les meilleures réponses possibles étant donné un tel descripteur. À première vue, l'approche est assez traditionnelle : une passe de Recherche d'Informations sélectionne des documents, qui sont ensuite découpés en passages dans lesquels la réponse est recherchée. On peut d'ailleurs noter un certain désaccord dans la littérature pour savoir si le dé-

coupage en passages fait partie de la Recherche d'Informations ou non. Nous avons cependant une certaine différence de philosophie. L'idée traditionnelle est, en stéréotypant un peu, de chercher *le* document où se trouve *la* phrase qui contient *la* bonne réponse. [Litkowski 2001] par exemple mesure à quel rang à la sortie de leur recherche se trouve le premier document contenant la bonne réponse, et s'appuie sur ces résultats pour limiter leur analyse aux 20 premiers. Nous considérons que dans le cadre de questions posées par des utilisateurs réels, et sur une base de documents suffisamment grande et adaptée, les bonnes réponses apparaissent plusieurs fois, voire même un grand nombre de fois. Cela est connu sous le terme général de redondance. Notre but a donc été non pas tellement de trouver le bon document ou la bonne phrase mais plutôt d'augmenter le *taux de présence* de la bonne réponse par rapport à la quantité de texte à traiter. Nos approches sont fondamentalement plus des approches de *filtrage*, où nous cherchons à supprimer le texte non-pertinent. Il est ainsi possible de voir notre sélection de documents comme comprenant deux étapes : une suppression de tous les documents qui ne peuvent pas répondre à la question, puis dans un deuxième temps un calcul de score pour faire une sélection liée aux besoins en temps de réponse. De même la sélection de passages supprime toutes les lignes trop lointaines des éléments décrits dans le DDR, avant de calculer un score présent là encore pour la gestion de la vitesse.

Cet aspect est aussi visible dans le calcul des scores de réponse. Il est très similaire dans le principe à des scores de compacité présentés dans [Gillard, et al. 2006b]. Cependant il intègre dans sa définition la notion de redondance, en tenant compte de toutes les instances d'une réponse candidate et en intégrant des mesures compensatoires du biais qu'une utilisation aveugle de la redondance pourrait provoquer, en favorisant trop les entités naturellement fréquentes dans la langue. Ce score, qui est un des points les plus importants dans la qualité globale des réponses du système, reste encore très ad-hoc. Il mériterait d'être construit sur des bases plus solides, et un travail est en cours pour le redéfinir sur des bases probabilistes, un peu dans l'esprit de [Gillard, et al. 2007].

Enfin le dernier point à mettre en valeur dans le système proposé est la facilité avec laquelle il permet l'expérimentation. Le Descripteur De Recherche, tout d'abord, par sa représentation synthétique et complète de la recherche à effectuer permet un bon niveau de diagnostic en voyant directement l'impact des modifications sur ses règles de génération ou sur les règles de prédiction des types attendus. Mais il est aussi possible de modifier directement son contenu pour mesurer l'impact de corrections sur les étapes suivantes. De même ces étapes suivantes, choix des documents, des snippets et extraction des réponses ont des fonctionnalités et fonctionnements relativement simples à comprendre permettant de savoir, en comparant DDR et documents, pourquoi un document ou un passage spécifique a été perdu à un niveau donné et d'agir en conséquence. Il est là de même possible de changer ces résultats intermédiaires (liste de documents ou de passages) pour étudier l'effet de corrections ou variations sur la suite. Tout cela est mis en exergue par la vitesse élevée de l'ensemble, permettant un retour rapide sur toute modification.

Troisième partie

Évaluation du système Question-Réponse

Introduction

Rendre un ordinateur capable de répondre à n'importe quelle question qui lui soit posée est un vieux rêve de l'Intelligence Artificielle. Cependant c'est en soi un problème un peu vague. À la fin du siècle dernier, la Recherche d'Informations consistait à devoir retourner un ensemble de documents en réponse à une requête. Les spécialistes du domaine ont alors ressenti un besoin de se rapprocher plus de cet idéal de l'IA en tentant de donner des réponses plus précises que des documents. Étant des habitués de l'évaluation, ils ont cherché à définir une *tâche*, ou en d'autres termes une sous-partie bien définie du problème qui paraisse faisable et évaluable. Paraphrasant [Voorhees & Tice 1999], la tâche devait être ni trop simple ni trop difficile de façon à ce que les résultats permettent d'apprendre quelque chose sur la viabilité des différentes approches envisagées. Depuis, toutes les évaluations du domaine ont eu à essayer de maintenir cet équilibre entre intérêt applicatif, scientifique et difficulté.

Concevoir une telle campagne d'évaluation pose des problèmes. Nous rentrerons plus dans les détails au chapitre suivant mais les questions essentielles à se poser sont :
– À quels types de questions voulons-nous nous intéresser ?
– Sommes-nous capables d'en créer ?
– Sommes-nous capables d'évaluer les réponses ?
Il n'y a pas fondamentalement de bonne ou de mauvaise réponse à ces questions. En dehors d'un contexte applicatif direct ces choix ne peuvent être qu'arbitraires, même si des critères pertinents de faisabilité et de l'évaluation et des systèmes sont pris en compte.

Face à ces questions, les campagnes d'évaluation ont beaucoup évolué d'une année sur l'autre et chaque instance a eu ses caractéristiques propres. Nous présentons donc un panel représentatif de ces campagnes, en mettant en avant les points les caractérisant. Nous présentons aussi à cette occasion les métriques utilisées dans le domaine pour quantifier la qualité des systèmes.

Une fois ce tour d'horizon fait, nous nous intéressons aux campagnes spécifiques auxquelles nos systèmes ont participé. Nous présentons les caractéristiques spécifiques de ces campagnes et les résultats que nous avons obtenus.

Cependant regarder des résultats bruts ne donne qu'une partie de l'information. Nous avons donc examiné le comportement du système suivant plusieurs aspects. Le premier a été de quantifier l'impact

sur les résultats de la taille du corpus de questions d'entraînement. Ce corpus agit à deux niveaux : il sert à estimer les paramètres de fonctionnement optimaux, mais aussi il donne des exemples pertinents pour les linguistes travaillant sur l'analyse de la langue et la prédiction de type attendu pour les réponses. Notre étude séparera les deux contributions.

Un autre aspect est une analyse modulaire. L'analyse de la langue, la prédiction des types possibles pour la réponse et l'extraction de passages sont des modules séparés pour lesquels il nous a paru intéressant d'évaluer les contributions individuelles et en combinaison.

Enfin un dernier aspect est le contrôle de la vitesse. Nous avons prévu dans les algorithmes deux paramètres pour contrôler la vitesse de réponse du système. Il s'agit du nombre maximum de documents examinés et du nombre maximum de candidats réponse extraits. Le dernier chapitre avant la discussion globale sera consacré à une analyse du comportement du système en fonction de variations sur ces paramètres.

Chapitre 10

Les campagnes d'évaluation Question-Réponse

10.1 Présentation générale des campagnes d'évaluation en Question-Réponse

L'évaluation internationale pionnière du domaine Question-Réponse est TREC (Text REtrieval Conference), organisée par le NIST aux Etats-Unis [Voorhees & Harman 2005]. Elle a introduit la première tâche Question-Réponse en 1999 et en a organisé une chaque année jusqu'en 2007. Pour 2008, la conférence, renommée TAC (Text Analysis Conference), s'est tournée plus vers des problèmes d'extraction d'opinion qui sont hors de notre cadre. En Europe, Clef (Cross-Language Evaluation Forum) a été créée en 2000 pour être le pendant pour les langues européennes de TREC qui se consacre à l'anglais [Peters & Braschler 2001]. Question-Réponse y a été introduit en 2003 et donne là aussi lieu à des évaluations annuelles. Une sous-tâche qui nous intéresse spécifiquement, QAst (Question Answering on Speech Transcripts), est présente depuis 2007 [Turmo, et al. 2007]. Un équivalent asiatique existe se nommant NTCIR (NII Test Collection for IR Systems), créé en 1999 et où Question-Réponse est apparu en 2002 [Fukumoto, et al. 2002]. Cela fait le tour des évaluations internationales les plus influentes. En France à ce jour deux évaluations Question-Réponse ont eu lieu. La première est EQueR (Evaluation des systèmes Question-Réponse) en 2005 [Ayache, et al. 2006] dans le cadre du projet EVALDA. Ce projet n'a pas encore connu de suite. Enfin la seconde a eu lieu dans le cadre du projet Quaero [Quaero 2008 ; Quintard 2009], un grand projet franco-allemand centré sur le contenu numérique, et en particulier l'extraction d'informations, leur analyse et classification, et en général leur exploitation. Cette campagne d'évaluation est à l'heure actuelle limitée aux partenaires du projet. Nous faisons cependant partie des participants ce qui nous permet de nous y référer. Le tableau 10.1 synthétise les caractéristiques principales de ces nombreuses évaluations. Les sections suivantes vont développer plus avant ces caractéristiques.

	TREC									QA@Clef Main Track							QAst		NTCIR					EQueR	Quaero
	1999	2000	2001	2002	2003	2004	2005	2006	2007	2003	2004	2005	2006	2007	2008	2009	2007	2008	2002	2004	2005	2007	2008	2005	2008
Factuelles	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Définitions simpl.		●	●								●	●	●	●	●							●		●	
Définitions					●													●				●	●		●
Pourquoi																						●			●
Comment											●											●			●
Oui/non																								●	●
Listes ouvertes					●	●	●	●	●				●	●	●				●		●	●	●		
Listes fermées			◇	◇									●	●	●				●		●	●		●	●
Enchaînements			◇				●	●	●					●	●				●	◇	●				
Thématisation						●	●	●	●																
Information						●	●	●	●																
Journaux	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				●	●	●	●	●	●	
Parole																	●	●							
Politique																								●	
Médical																								◇	
Juridique																●									
Wikipédia														●	●										
Blogs									●																
Web en général																									●
Classe donnée					●	●	●	●	●										●	●					
Réponses multiples	●	●	●							●			●				●	●	●	●	●	●	●	●	●
Réponses longues	●	●	●							●	●													●	●
Support												●	●	●	●									●	●
Translingue										●	●	●	●	●	●	●					◇	◇	●		
Docs. alignables																●									
Restr. temporelles												●	●	●	●										
Timecodes																		●							

◇ : tâche à part

Sources :

– TREC : [Voorhees & Tice 1999 ; Voorhees 2000 ; Voorhees & Tice 2001 ; Voorhees 2002 ; Voorhees 2003 ; Voorhees 2004 ; Voorhees & Dang 2005 ; Dang, et al. 2006 ; Dang, et al. 2007].
– QA@Clef Main Track : [Magnini et al. 2003 ; Magnini, et al. 2004 ; Vallin, et al. 2005 ; Magnini, et al. 2006 ; Giampiccolo, et al. 2007 ; Forner, et al. 2008] et des messages de la mailing-list pour la définition de la tâche 2009.
– QA@Clef QAst : [Turmo et al. 2007 ; Turmo et al. 2008] et des messages de la mailing-list pour la définition de la tâche 2009.
– NTCIR : [Fukumoto et al. 2002 ; Fukumoto, et al. 2004 ; Kato, et al. 2004 ; Kato, et al. 2005 ; Sasaki, et al. 2005 ; Sasaki, et al. 2007 ; Fukumoto, et al. 2007 ; Mitamura, et al. 2008].
– EQueR : [Ayache et al. 2006].
– Quaero : [Quintard 2009].

TAB. 10.1 – Tableau résumant les caractéristiques des principales évaluations Question-Réponse

10.2 Les types de questions

Le premier point à décider quand on définit une campagne d'évaluation Question-Réponse est le type de questions que l'on envisage de traiter. Le type de question le plus habituel est la question *factuelle*, telle que *"Qui est le président de la France ?"* où une réponse précise de peu de mots, souvent une entité nommée ou assimilée, est attendue. De ce type de question est dérivé le type *Liste* qui attend plusieurs réponses de ce type, qui se décline en *Liste fermée* où le nombre d'éléments attendus est précisé, comme pour *"Nommez les sept nains."*, et en *Liste ouverte* où ce nombre n'est pas indiqué *"Qui ont été les présidents élus de la cinquième République en France ?"*. Ces questions posent peu de difficultés au niveau de l'évaluation, savoir si une réponse donnée par un système est correcte ne donne en général pas lieu à ambiguïté.

Un autre type de question traité couramment dans ces évaluations concerne les questions de *Définition*. Le problème des définitions dans le cas général, comme pour *"Qu'est-ce qu'un parachute doré ?"*, pose un gros problème d'évaluation. En effet il est difficile de dire quand une définition est correcte et suffisamment complète. Le NIST a proposé dans [Voorhees 2003] une méthodologie générale pour les évaluer : faire une liste de l'ensemble des informations élémentaires présentes sur le sujet dans les documents de référence et les classer entre indispensables dans une définition et optionnelles. Cependant même en suivant cette méthodologie ils ont noté une importante variance dans les résultats d'un évaluateur humain à l'autre, essentiellement due à des désaccords sur quelles informations sont indispensables. Pour faciliter cette évaluation un sous-type a été défini dans QA@Clef nommé *Définition simplifiée* [Vallin et al. 2005]. Pour ces questions, telles que *"Qui est Patrick de Carolis ?"*, une réponse simple est demandée, du même ordre de complexité que celles associées aux questions factuelles. N'importe quelle réponse donnant au moins une caractéristique pertinente de l'objet est considérée correcte. Par exemple *PDG de France Télévision* ou encore *fils de Dominique de Carolis* seraient acceptés.

Deux types de questions plus avancés parfois rencontrés sont les questions *Pourquoi* et *Comment*, telles que *"Pourquoi le ciel est-il bleu ?"* ou *"Comment retirer une tâche de vin rouge ?"*. Décider de la validité d'une réponse peut poser de gros problèmes (le système doit-il fournir une procédure ? Une synthèse de diverses informations ?) ce qui incite les campagnes d'évaluation à poser des contraintes spécifiques. Par exemple la campagne QA@Clef 2005 [Vallin et al. 2005] autorisait les questions *Comment* mais limitait leurs réponses à une entité simple. De même, la campagne Quaero autorisait les deux types mais exigeait que la réponse tienne en une phrase extraite d'un document. Le problème de l'évaluation des réponses à ces types de questions dans le cas général est loin d'être résolu.

En apparence plus simples, les questions *Oui/Non*, ou encore *Questions fermées*, telles que *"Est-ce que Saturne est une planète gazeuse ?"* posent le problème de la justification. En effet il n'est dans ce cas pas raisonnable de répondre simplement oui ou non. Avec une probabilité de 50%, il serait difficile de faire la part entre qualité du système et chance sauf à avoir un très grand nombre de questions. Les évaluations traitant de ce type de questions doivent donc demander aux systèmes de fournir la ou les phrases et documents les ayant poussé à prendre leur décision.

Dans l'idée de se rapprocher plus d'une application interactive, une forme de question a été introduite impliquant des *Enchaînements*. Les questions sont organisées en séries tournant autour du même sujet et des anaphores implicites ou explicites sont possibles dans les questions. Par exemple, cette série est traduite de NTCIR [Kato et al. 2005] :
– *A quel genre littéraire appartient la série Harry Potter ?*
– *Qui est l'auteur ?*
– *Qui sont les personnages principaux dans cette série ?*
– *Quand le premier volume a-t-il été publié ?*
– *Quel était son titre ?*
– *Combien de titres ont été publiés en 2001 ?*
– *Dans quelles langues a-t-il été traduit ?*
– *Combien d'exemplaires ont été vendus au Japon ?*
Cette série a été construite via une interaction réelle entre un humain et un système Wizard of Oz. Elle montre bien en particulier les variations de focus qui s'affine, s'élargit et se déplace au long des questions. Elle est en cela très similaire avec des interaction réelles entre un humain et un système. Dans QA@Clef, où ce type de question est également présent, l'approche est beaucoup plus limitée : la première question fixe le thème via son focus ou sa réponse et les questions suivantes ne peuvent faire référence qu'à la première question et ce de façon explicite. Par exemple cette série est tirée de l'édition 2008 :
– *Qui a été ambassadrice de l' UNICEF entre 1988 et 1992 ?*
– *Quelles langues parlait-elle ?*
– *Qui épousa-t-elle en 1969 ?*
– *Où est-elle morte ?*
Le résultat est bien moins naturel, ressemblant bien plus à une *interrogation écrite* qu'a une vraie recherche d'informations par quelqu'un qui ne connait pas la réponse. Ce type de question est cependant plus facile à traiter. Une évaluation TREC a aussi eu lieu avec la même organisation de questions que QA@Clef, mais les résultats ont été considéres non probants scientifiquement car trop contraints par la sélection initiale de documents [Voorhees & Tice 2001].

En alternative aux enchaînements, la campagne d'évaluation TREC a proposé une *Thématisation* des questions. Un thème général est donné et les questions tournent toutes autour de lui [Voorhees 2004] :
– Thème : *Comète Hale Bopp*
  – FACTUEL - *Quand la comète a-t-elle été découverte ?*
  – FACTUEL - *Avec quelle fréquence approche-t-elle la Terre ?*
  – LISTE - *Dans quels pays était-elle visible lors de son dernier retour ?*
  – AUTRE
Le principe de devoir résoudre les anaphores reste le même, la résolution étant cependant évidente vu qu'il s'agit à chaque fois de l'entité thème de la série. Avec ce type de questions thématiques on simule une application où l'utilisateur est un expert cherchant un ensemble d'informations sur un sujet dont il a déjà une connaissance vague lui permettant de savoir a priori les questions à poser. Dans la même optique ils ont introduit le notion de question *Autre*, notée dans le tableau sous le nom *Information*, qui n'est pas une question en soi, mais demande à fournir l'ensemble des informations sur le thème qui n'ont pas encore été demandées par les autres questions. La méthodologie d'évaluation par

informations élémentaires qu'ils utilisent est une extension de celle qu'ils avaient développée pour les questions de définition.

10.3 Types de documents

Un autre aspect de la définition d'une évaluation Question-Réponse est l'ensemble des documents dans lesquels les réponses doivent être cherchées. Le tableau 10.2 donne les caractéristiques de quelques unes des collections de documents utilisées dans nos évaluations.

	QAst 2008	QA@Clef	Ritel	Quaero
Type	Parole	Journaux	Web	Web
Années	2004	1994-95	2004	2008
Nombre de documents	12	200K	63K	500K
Nombre de phrases	2,3K	3M	29M	82M
Nombre de mots	87K	70M	380M	840M
Nombre de caractères	460K	400M	2,4G	4,2G
Phrases/document	200	17	450	170
Mots/phrase	37	25	13	10
Caractères/mot	5,3	5,4	6,9	5,3

TAB. 10.2 – Types et tailles de plusieurs collections de documents utilisés dans des évaluations QR en français. QAst 2008 contient des transcriptions d'émissions de radio. QA@Clef contient les années 1994 et 1995 du journal *Le Monde* et de l'*Agence Télégraphique Suisse*. Ritel et Quaero sont des collections de pages du Web.

Nous pouvons constater que d'une évaluation à l'autre ces caractéristiques varient fortement. La principale est le type de documents. Les documents journalistiques (journaux, dépêches), sont les plus courants et ont des avantages certains : les documents sont plutôt factuels, la densité d'informations est assez élevée et avec une certaine redondance, les mêmes évènements étant abordés sur plusieurs jours, la qualité de la langue est plutôt bonne sans être trop recherchée, les documents sont en général monothématiques et les sujets abordés sont intéressants pour tout un chacun, rendant entre autres la création des questions plus facile. Ils ont par contre deux inconvénients : la couverture des informations tend à être limitée à l'actualité et couvre peu les questions demandant des connaissances de type encyclopédique. Le second est le problème du coût. Ce type de documents a un coût commercial assez élevé et nécessite des négociations avec les propriétaires des droits. En conséquence les documents proviennent en général de peu de sources (2 pour QA@Clef), sont assez vieux (1994-1995 dans le cas de QA@Clef) et sont en quantité relativement faible (quelques années).

En complément les évaluations QA@Clef récentes (2007-2008) ont rajouté l'ensemble de Wikipedia. La quantité de données disponibles ainsi est très importante et variée et le niveau de langue reste en général bon. Cependant la difficulté est en pratique bien plus élevée qu'avec les documents journalis-

tiques à cause de trois problèmes principaux : le nombre de concepts rencontrés tend à être bien plus élevé que dans l'actualité, une encyclopédie ayant, par définition, une plus large variété de sujets, la redondance d'informations est assez faible et la structure des documents est très spécifique, avec de nombreuses anaphores référant au titre de la page ou aux titres de section.

Un alternative explorée de temps en temps sont les documents de domaines spécifiques. Documents politiques (débats du Sénat, évaluation EQueR), médicaux (articles scientifiques du domaine médical, EQueR) ou juridiques (JRC-Acquis, le Journal Officiel de l'Union Européenne, évaluation QA@Clef 2009). Ces tâches sont intéressantes et ont des possibilités applicatives évidentes, mais demandent à traiter une langue spécifique d'un domaine dans lesquels les personnes travaillant sur l'analyse de la langue peuvent ne pas être compétentes (terminologie et concepts spécifiques en particulier). De plus l'évaluation en est rendue plus difficile avec le besoin de recourir à des spécialistes pour créer les questions et parfois même évaluer les réponses.

L'évaluation QAst a été organisée pour travailler spécifiquement sur une modalité de documents : la parole. Sous la forme de transcriptions manuelles ou automatiques, elle peut venir de bien des sources avec des propriétés spécifiques. En 2008 par exemple étaient proposées des transcriptions de séminaires (un locuteur principal) et de réunions de travail (plusieurs intervenants) en anglais, des transcriptions d'émissions de radio en français, et des transcriptions de débats du parlement européen en anglais et espagnol. Les niveaux de langue, couverture conceptuelle et complexité syntaxique varient donc beaucoup d'une sous-tâche à l'autre, mais reste commune la très faible quantité de documents du au coût des transcriptions manuelles et de la difficulté d'obtenir des transcriptions automatiques.

Enfin la dernière source de documents étudiée dans ces évaluations est bien évidemment le Web. Ce peut être des parties spécifiques (Wikipedia comment nous avons vu, ou encore des blogs dans le cadre de TREC) où l'on peut espérer une certaine uniformité de qualité de langue. Mais les dernières expériences ont lieu sur des gros corpus de pages choisies via un moteur de recherches (Altavista pour notre corpus interne Ritel, Exalead pour l'évaluation Quaero) sur une série de thèmes (cas de Ritel) ou de recherches par mots-clés faites par de vrais utilisateurs (cas de Quaero). Le Web tout venant est particulièrement difficile à traiter à cause d'une absence totale d'uniformité dans les contenus. Les niveaux et qualité de la langue sont extrêmement variables, la sélection de la langue des pages parfois incorrecte, la structuration souvent assez obscure rendant l'extration du texte peu fiable. De plus des pages existent contenant par exemple des listes de mots ou de noms. Ces mots et noms peuvent apparaître dans les questions et poser problème au moment de la recherche. Pire encore sont les pages, souvent pornographiques ou publicitaires, comportant des listes de mots-clés pour essayer d'attirer les moteurs de recherches et qui attirent ainsi très bien les passes de sélection de documents et de passages des moteurs de Question-Réponse, noyant ainsi les documents pertinents. De plus, ces collections ont beau être importantes leur taille est en pratique relativement faible par rapport aux nombres de thèmes abordés, résultant en une redondance en pratique assez faible. Tout cela rend les documents tout-venant du web les plus difficiles à traiter.

10.4 Autres caractéristiques des campagnes d'évaluation

En plus des caractéristiques primaires que sont les types de questions et de documents de nombreuses caractéristiques secondaires donnent à chaque évaluation sa propre personnalité. Nous en avons sélectionné certaines qui semblent particulièrement intéressantes.

La première touche les questions elles-mêmes. La présence de plusieurs catégories de questions pose le problème, pour les systèmes, de détecter à quelle classe une question appartient. Cette classification est souvent simple mais peut dans certains cas poser problème. Par exemple un système ne peut savoir a priori que *Qui a écrit Good Omens ?* a pour réponse deux auteurs et non un seul. Considérant que ce problème ne faisait pas partie de la tâche, certaines évaluations, en particulier TREC, ont décidé d'indiquer pour chaque question à quelle catégorie (factuelle à réponse unique, définition, liste...) elle appartient. Cette option n'est cependant pas pertinente dans un cadre interactif. En effet il ne paraît pas raisonnable de demander à l'utilisateur la classe de la question qu'il vient de poser.

Un ensemble de caractéristiques touche les réponses attendues de la part du système. Un point de contentieux dans la définition d'une tâche est le nombre de réponses attendues, ou plus spécifiquement si une seule réponse (une seule entité, une seule définition, une seule liste d'entités...) est attendue ou, alternativement, le système peut proposer plusieurs réponses de la plus à la moins probable. Dans le premier cas, les organisateurs considèrent que Question-Réponse, en tant que version plus précise de la recherche de documents, implique devoir donner *la* réponse et rien d'autre. Le fait qu'une question a rarement une seul réponse correcte n'est pas pris en compte. Dans l'autre cas, ils considèrent que donner plusieurs réponses permet une évaluation un peu plus fine des systèmes eux-mêmes, et que de toutes façons d'un point de vue applicatif présenter plusieurs réponses possibles à une utilisateur est pour certaines modalités comme une interface web parfaitement raisonnable. Les deux points de vue se défendent, mais dans un cadre de développement de système avoir plusieurs réponses semble effectivement permettre de constater des changement de score plus progressifs au moment des réglages fins optimisants les performances du système, en particulier en utilisant la métrique MRR décrite section 10.5.

Une autre de ces caractéristiques touchant les réponses attendues est historique : le concept de *réponse longue*. Le problème ayant été initialement conçu comme un raffinement de la sélection de documents les premières évaluations ne demandaient pas encore une réponse précise mais un passage d'un nombre limité de caractères (50 ou 250 suivant les évaluations). La réponse était considérée correcte si le passage contenait la réponse précise attendue. TREC a abandonné cette approche en 2002 pour exiger à la place les réponses précises. Cette méthode d'évaluation a cependant été reprise de temps en temps en alternative optionnelle aux réponses précises pour permettre aux participants de mesurer la qualité de leur extraction de réponse.

Une variante de la notion de réponse longue utilisée à l'origine dans le cadre de QA@Clef est la notion de *support*. Un passage de 250 caractères est fourni en plus de la réponse et son but est de convaincre l'évaluateur de la validité de la réponse. L'idée sous-jacente est de pouvoir proposer une

interface utilisateur similaire aux moteurs de recherche actuels où un ensemble de documents sont désignés par leur titre et des passages courts en sont extraits permettant d'un coup d'œil de décider de leur pertinence. Ce n'est pas à proprement parler une justification. En particulier le passage peut contenir des anaphores référant à des entités de la question situées en dehors de la plage des 250 caractères. Le passage se doit juste d'être convaincant.

Un dernier grand axe concerne la multilingualité. La tâche Question-Réponse a initialement été définie comme un problème monolingue : questions et documents sont dans la même langue. Cependant, pour en particulier une question de quantité d'informations disponibles voire de biais éditorial, il est intéressant de pouvoir faire des recherches dans des documents d'une langue que l'on ne parle pas. De la vient la notion d'évaluation *translingue*. Les questions sont dans une langue, la langue source, et les documents dans une autre, la langue cible. Les réponses sont données dans la langue cible, le problème de leur traduction étant considérée comme ne relevant pas de la tâche. NTCIR organise ainsi des campagnes d'évaluation translingues entre japonais et anglais ou chinois et anglais. Pour QA@Clef une matrice est construite avec les langues européennes principales et des ensembles de questions construits pour chacune des langues. La combinatoire a cependant donné lieu à une trop grande dispersion des efforts, avec une grande partie des paires de langues n'ayant qu'un ou deux participants pour l'évaluation associée, rendant toute comparaison sérieuse illusoire. En 2009 une approche alternative va être tentée avec des *documents alignables*, c'est-à-dire des documents disponibles dans toutes les langues couvertes et dont les versions sont des traductions strictes les unes des autres, comme les instances européennes en fournissent beaucoup. Les questions auront la même propriété, un pool de questions traduites strictement dans toutes les langues, et les participants seront libres de sélectionner les langues de questions et de documents qu'ils veulent, voire de croiser les réponses entre plusieurs sorties de systèmes. Les résultats d'évaluation obtenus devraient du coup être beaucoup plus comparables.

En 2005, dans le but d'augmenter graduellement la difficulté des questions posées, la campagne d'évaluation QA@Clef a décidé de codifier la notion de *restriction temporelle*. Il s'agit d'une catégorie couvrant les questions contenant une clause sélectionnant une plage temporelle spécifique dans laquelle le réponse est attendue. Par exemple pour *Qui est le président de la France ?* n'importe quel nom de président fourni par les documents serait correct, mais pour *Qui était le président de la France en 1994 ?* seul *François Mitterand* est correct. Les informations temporelles nécessaires peuvent être explicites dans les documents, c'est à dire présentes dans le texte donnant la réponse, ou implicite via la date du document, généralement disponible dans le cas journalistique. Une partie des questions a alors été définie comme devant comporter de telles restrictions, et les systèmes ont dû être adaptés en conséquence pour les prendre en compte. On peut imaginer qu'une évaluation future prendra en compte des restrictions géographiques du même ordre.

Enfin la recherche des réponses dans des documents sortis de systèmes de transcription automatiques a ses spécificités. Les systèmes de transcription font des erreurs par rapport à ce qui a été réellement dit mais en contrepartie fournissent la position temporelle de chacun des mots reconnus. Du coup les systèmes de recherches d'informations peuvent tenter de reconstituer ce qui avait été réellement dit en s'appuyant sur les entités de la question et des connaissances sémantiques haut niveau absentes

Question : *What is the Vlaams Blok ?*

Transcription manuelle : *the Belgian Supreme Court has upheld a previous ruling that declares the Vlaams Blok a criminal organization and effectively bans it .*
Réponse : *criminal organisation*

Extrait d'une **transcription automatique (format CTM) :**
(...)
20041115_1705_1735_EN_SAT 1 1018.408 0.440 Vlaams 0.9779
20041115_1705_1735_EN_SAT 1 1018.848 0.300 Blok 0.8305
20041115_1705_1735_EN_SAT 1 1019.168 0.060 a 0.4176
20041115_1705_1735_EN_SAT 1 **1019.228** 0.470 criminal 0.9131
20041115_1705_1735_EN_SAT 1 **1019.858 0.840** organisation 0.5847
20041115_1705_1735_EN_SAT 1 1020.938 0.100 and 0.9747
(...)
Réponse : 1019.228 1020.698

FIG. 10.1 – Question *What is the Vlaams Blok ?* et réponse dans une transcription manuelle (haut) et automatique (bas) transcripts. Les colonnes du format CTM sont nom de document, numéro de canal, position temporelle, durée, mot, indice de confiance

dans le système de transcription d'un côté, et sur une transcription phonétique des mots reconnus de l'autre. Le résultat peut alors prendre la forme d'un intervalle dans le signal où la réponse est dite, et que l'on peut du coup rejouer à l'utilisateur. On parle alors de *timecodes*. Dans cette optique l'évaluation QAst 2008 a proposé l'utilisation de ces intervalles temporels comme réponse attendue du système, comme on peut le voir figure 10.1.

10.5 Les métriques

Plusieurs métriques ont été développées au cours du temps. Nous allons voir dans un premier temps celles définies pour le cas le plus courant où les réponses peuvent être classées de façon binaire correcte ou incorrecte, puis nous regarderons ce qui a été proposé pour évaluer les réponses dans les cas de listes, de définitions complexes ou de recherche d'informations complémentaires.

La métrique la plus simple est la *précision* (accuracy), le ratio entre nombre de réponses correctes et nombre total de questions. Dans le cas où le système peut donner plusieurs réponses par question on ne considère que la première. En notant CR_i le rang de la première réponse correcte pour la question i, prenant pour valeur $+\infty$ si aucune réponse correcte n'a été trouvée :

$$\text{précision} = \frac{\#\mathrm{CR}_i = 1}{\#\text{questions}} \tag{10.1}$$

Cette mesure donne directement la probabilité que le système soit correct quand il donne une réponse, ce qui est utile dans un cadre applicatif. Cependant d'un point de vue de développement de système cette information manque de finesse. Il est intéressant de connaître la densité de réponses correctes parmi les n premières réponses proposées et ne pas se limiter à la première La mesure la plus immédiate pour les prendre en compte est le *top-n*, la précision en acceptant les réponses correctes de rang 1 à n :

$$\text{top-}n = \frac{\#\text{CR}_i \leq n}{\#\text{questions}} \tag{10.2}$$

Cette mesure n'est en pratique utilisée qu'avec n égal au nombre maximal de réponses autorisé pour le système, ce qui en fait une sorte de *rappel*. Le système essaie cependant de mettre les réponses les plus sûres en premier. Pour mesurer plus la qualité de ce classement, le *Mean Reciprocal Rank* (Moyenne des Réciproques des Rangs), ou *MRR* est souvent utilisée. La réponse correcte la mieux classée est pondérée par l'inverse du rang auquel elle a été proposée. Une absence de réponse correcte correspond à un rang infini et donc à une contribution nulle. Le score final est la moyenne de ces contributions :

$$\text{MRR} = \frac{\sum \frac{1}{\text{CR}_i}}{\#\text{questions}} \tag{10.3}$$

Les mesures de précision, MRR et top-n forment un ensemble de valeurs croissantes utiles pour avoir une idée de la qualité d'un système et de ses possibilités d'évolution et aussi pour comparer la résultats de plusieurs versions du même système. Ce sont les métriques que nous avons utilisées dans nos expérimentations.

Les questions de listes, de définitions au sens large et les demandes d'informations complémentaires ont la particularité de donner lieu à des réponses que l'on peut considérer plus ou moins correctes. Une simple classification binaire correct/incorrect telle qu'utilisée pour les questions factuelles semble insuffisante. L'idée est alors de donner un score entre 0 et 1 à chaque réponse, score qui pourra ensuite être intégré directement dans les métriques standard.

Pour les listes, le problème est de comparer une liste d'éléments fournis par le système à une liste de référence. On notera C le nombre de corrects, i.e. le nombre d'éléments corrects entre les deux listes, L le nombre d'éléments de référence et S le nombre d'éléments donnés par le système. Les évaluateurs de TREC ont proposé dans [Voorhees 2003] d'utiliser comme score la F-mesure, qui est définie en fonction de la précision et du rappel :

$$P = \frac{C}{S} \qquad R = \frac{C}{L} \qquad F = \frac{2 \times P \times R}{P + R} \tag{10.4}$$

Alternativement, l'évaluation Quaero a proposé d'utiliser une métrique inspirée de la reconnaissance vocale où les réponses correctes font gagner des points et les erreurs en perdre ($S - C$ est le nombre d'erreurs) :

$$Q = \max(0, \frac{C - (S - C)}{L}) \tag{10.5}$$

Il n'y a pas à l'heure actuelle d'étude comparant ces deux métriques.

Le problème des définitions au sens large et celui des demandes de compléments d'informations a été unifié d'un point de vue évaluation par TREC. Pour une question donnée un ensemble d'informations élémentaires qui lui sont lié est constitué à partir des documents et des réponses des systèmes. De plus une sous-partie de ces informations sont annotées comme vitales. Par exemple, pour la question *Qu'est-ce qu'un parachute doré ?* [Voorhees 2003] ces informations peuvent être :
– contrat entre une entreprise et ses hauts dirigeants (vital)
– dédommagement pour les cadres perdant leur emploi (vital)
– dédommagement généralement très élevé (vital)
– encouragement pour les cadres à ne pas résister aux rachats bénéfiques pour les actionnaires
– méthode pour les enterprises pour attirer certains cadres
– dédommagement non-imposable au titre des cotisations sociales
Le nombre d'informations vitales r et non-vitales mais correctes a présentes dans la réponse du système est compté. Le rappel est estimé directement sur les informations vitales. Notant V le nombre d'informations vitales dans la référence :

$$R = \frac{r}{V} \tag{10.6}$$

La précision est plus difficile à estimer en soi, un extrait pouvant contenir plusieurs informations. Les évaluateurs se sont du coup tournés vers une notion de compacité. Un nombre de caractères maximal A est autorisé en fonction du nombre d'informations présentes et le dépasser baisse la précision. Notant T le nombre de caractères de la réponse du système :

$$A = 100 \times (r + a) \qquad P = \min(1, 1 - \frac{T - A}{L}) \tag{10.7}$$

Le score final est obtenu par une F-mesure pondérée entre précision et rappel :

$$F = \frac{10 \times P \times R}{9P + R} \tag{10.8}$$

Cette méthodologie d'évaluation est très intéressante mais demande un gros travail pour les évaluateurs, et il a été constaté que des problèmes d'accord inter-annotateurs sont courants dans le choix de quelle partie des informations devait être considérée vitale. Elle reste une approche à garder à l'esprit si l'on veut aller un jour vers des réponses à des questions complexes construites par synthèse entre plusieurs documents.

Enfin certains systèmes ont la capacité de fournir des niveaux de confiances avec les réponses qu'ils donnent, en d'autres termes la probabilité estimée que la réponse soit juste. Deux métriques existent pour aider à estimer la qualité de ces niveaux de confiance. La première, *K* [Herrera, et al. 2004], essaie d'estimer la qualité de ces niveaux de confiance en faisant gagner des points proportionnellemant à la confiance sur les réponses correctes, et en faisant perdre dans la même proportion pour les réponses fausses. Les réponses dupliquées, où le système a répondu plusieurs fois la même chose sans s'en apercevoir sont éliminées. Pour un ensemble de Q questions, on note pour la question i :

– S_i le nombre de réponses données par le système
– R_i le nombre de réponses différentes correctes trouvées dans les documents
– $e_{i,j}$ l'évaluation de la j-ème réponse du système à la question i, +1 si correct et -1 si faux
– $c_{i,j}$ la confiance du système en sa j-ème réponse à la question i

Les valeurs R_i correspond aux comptes de réponses de référence, trouvées à la main par les évaluateurs et complétées après évaluation des sorties des systèmes. Alors la mesure K est définie comme :

$$K = \frac{1}{Q} \sum_{i=1}^{Q} \frac{\sum_{j=1}^{S_i} e_{i,j} c_{i,j}}{\max(S_i, R_i)} \tag{10.9}$$

Cette mesure donne une valeur entre -1 et 1. Dans le cas où les réponses ne sont pas strictement correctes ou incorrectes (listes, définitions...), mais donnent lieu à un score entre 0 et 1 tel qu'obtenu par les F-mesures ou le score Q présentés précédemment, on peut imaginer utiliser le mapping linéaire :

$$e = 2 \times score - 1 \tag{10.10}$$

Une variante de cette mesure nommée *K1* est définie pour les cas où les systèmes doivent donner une et une seule réponse. R_i est forcée à 1, simplifiant la formule en :

$$K1 = \frac{1}{Q} \sum_{i=1}^{Q} e_i c_i \tag{10.11}$$

L'autre mesure, nommée *Confidence Weighted Score*, essaie de prendre en compte la difficulté relative des questions estimée par le système. Utilisée dans les cas où le système doit donner une réponse par question avec un taux de confiance, la première étape est de classer les paires question/réponse dans l'ordre de la confiance la plus élevée à la plus faible. Le score est alors calculé en donnant un plus fort poids aux premières questions de la liste qu'aux dernières. En prenant comme hypothèse que i, numéro de question, suit l'ordre obtenu, on note $score_i$ le score entre 0 et 1 obtenu pour la question i. CWS est défini comme :

$$CWS = \frac{1}{Q} \sum_{i=1}^{Q} \frac{\sum_{j=1}^{i} score_j}{i} \tag{10.12}$$

En réorganisant un peu les termes, et notant H_n est le nombre harmonique n, $H_n = \sum_{i=1}^{n} \frac{1}{i}$, $H_0 = 0$, on peut réécrire la formule sous la forme :

$$CWS = \frac{1}{Q} \sum_{i=1}^{Q} (H_Q - H_{i-1}) score_i \tag{10.13}$$

On voit bien alors qu'il s'agit d'une pondération sur les réponses.

L'ensemble de ces métriques peuvent être regroupées sous la forme de la figure 10.2. Tout commence par une sortie de système sous la forme de triplets question, liste de réponses et optionellement niveaux de confiance associés. La première étape est l'évaluation humaine de ces résultats. A chaque

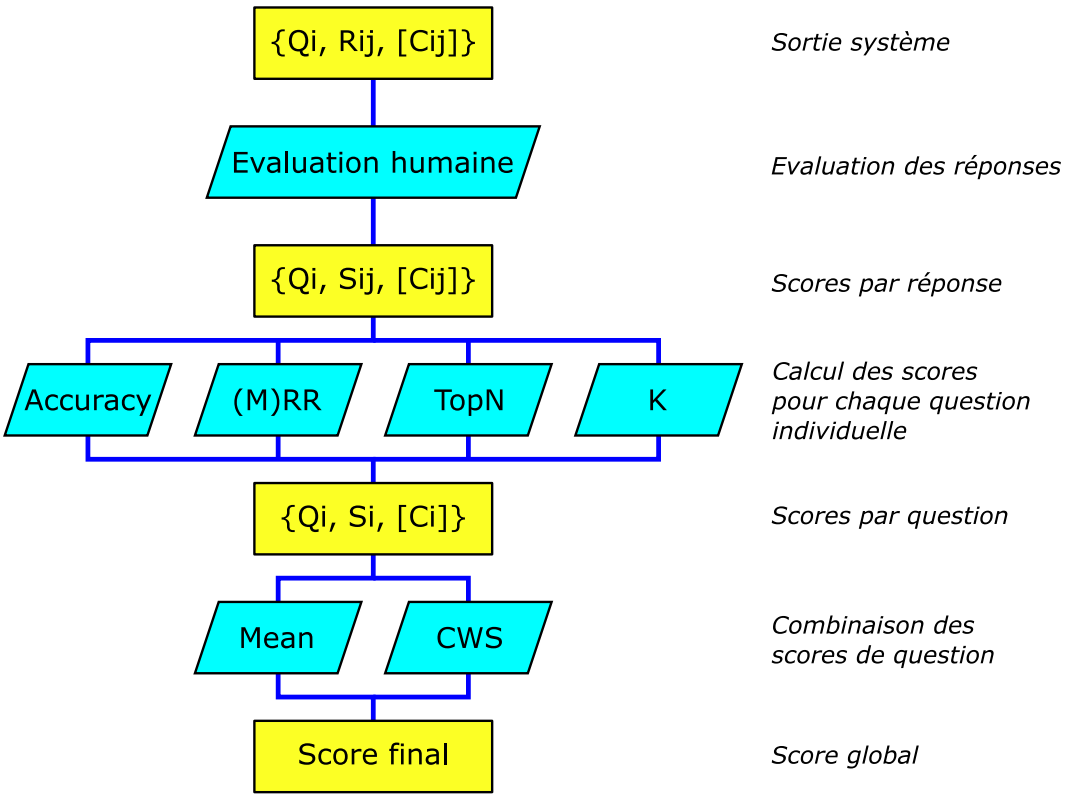


FIG. 10.2 – Structuration générale des métriques d'évaluation en Question-Réponse

réponse individuelle un score est associé, 0 pour les réponses fausses, 1 pour les correctes, et une valeur entre 0 et 1 pour les cas intermédiaires tels que se produisant sur les questions de listes ou de définitions complexes. Une fois cette étape franchie, il est alors possible de choisir les calculs à effectuer pour mettre tel ou tel point en valeur.

La première étape des calculs est l'estimation d'un score par question. Quatre alternatives sont possibles : l'*Accuracy*, où le score de la première réponse est pris, le *top-n* où la réponse avec le meilleur score est retenue, le *Reciprocal Rank* où chaque score est divisé par son rang et le meilleur est choisi, et enfin, si des niveaux de confiances individuels sont disponibles, la mesure *K* vue équation 10.9 en la limitant à une seule question. Chacune met l'accent sur un point particulier :

– l'*Accuracy* permet d'estimer la qualité d'un système censé retourner une seule réponse, comme c'est le cas des systèmes interactifs oraux
– le *Top-N* s'applique plus au cas d'une présentation type moteur de recherche web où une page de résultats est retournée. Il tente d'indiquer si la bonne réponse est dans la page.
– le *Reciprocal Rank* s'applique dans le même cadre et favorise la présence de la bonne réponse dans le haut de la page
– la mesure *K* cherche à estimer la qualité des niveaux de confiance fournis par le système

Enfin la seconde étape combine les scores par question en un score global. La méthode la plus courante est la simple moyenne, parfois pondérée en fonction des types de question. *CWS* est une méthode alternative intéressante quand des niveaux de confiance par question sont disponibles. Elle a pour but de mesurer la capacité du système à estimer la difficulté des questions les unes par rapport aux autres.

Ainsi, en choisissant les calculs élémentaires à effectuer il est possible de mettre en avant les points qui nous intéressent. En l'absence de niveaux de confiance, nous nous en tiendrons aux trois principaux, *Accuracy*, *MRR* et *top-n* avec la moyenne comme combinaison inter-questions.

Chapitre 11

Résultats aux campagnes d'évaluation officielles

11.1 La campagne d'évaluation QAst

L'évaluation QAst, *Question-Answering on Speech Trancripts* (Question-Réponse sur transcriptions de parole), a été créée en 2007 pour étudier le problème de la recherche d'informations précises dans la parole [Turmo et al. 2007]. Cependant la parole n'est qu'une modalité, et divers genres de documents peuvent être concernés. À l'heure actuelle quatre genres ont été étudiés.

Le premier genre est le séminaire. Une personne seule parle pendant que les autres présents dans la salle écoutent et interviennent parfois mais rarement. Le corpus *CHIL* [CHIL 2007] est constitué de 25 séminaires transcrits manuellement par ELDA et automatiquement par le système du LIMSI [Lamel, et al. 2005]. Ces séminaires sont en anglais, sur le thème du traitement de la langue et de la parole, et le plus souvent donnés par les locuteurs non-natifs.

Le second est la réunion de travail. Plusieurs personnes parlent ensemble, et souvent en même temps, se coupent la parole, etc. Le corpus *AMI* [AMI 2005] contient 168 réunions qui ont là encore été transcrites manuellement et automatiquement. Le système de transcription avait été conçu par l'Université de Edinbourg [Hain, et al. 2007]. Les réunions sont en anglais et portent sur la conception de télécommandes de télévision.

Ces deux premiers genres de données étaient les seuls disponibles pour l'évaluation 2007. L'année suivante deux genres supplémentaires ont été ajoutés. Le premier correspond aux émissions d'informations de la radio. Le corpus *ESTER* [Galliano, et al. 2006] en contient 10 heures, enregistrées de sources francophones variées (France Inter, Radio France International, Radio Classique, France Culture, Radio Télévision du Maroc) et transcrites manuellement par ELDA. Ces données ayant

été initialement constituées pour une évaluation des systèmes de reconnaissance vocale [Galliano et al. 2006] trois sorties de systèmes sont disponibles à trois taux d'erreurs de mots différents (11,0%, 23,9% et 35,4%).

Enfin le dernier genre est les sessions du Parlement Européen en anglais et espagnol. Chaque parlementaire fait à son tour un discours préparé pendant quelques minutes, le président de séance s'assurant du bon déroulement de la session. Pour chaque langue trois heures ont été transcrites dans le cadre du projet *TC-STAR* [TC-Star 2004-2008], manuellement là encore par ELDA et automatiquement dans le cadre d'une évaluation nous permettant d'avoir trois sorties par langue à des taux d'erreur de 11,5%, 12,7% et 13,7% pour l'espagnol et 10,6%, 14,0% et 24,1% pour l'anglais.

Tous ces genres de parole ont les mêmes caractéristiques de surface, en particulier les hésitations, bruits de respirations, erreurs de prononciation, correction, faux départs... Cependant la parole dans le cas des émissions d'informations et du Parlement Européen est moins spontanée que pour les séminaires et réunions car elle est en général préparée à l'avance et est proche structurellement des textes écrits. Cela pousse à qualifier les séminaires et réunions de *parole spontanée* et les émissions et débats de *parole préparée*. Les séminaires, monologues, ont une syntaxe assez différente de celle de l'écrit, avec phrases à rallonge et structuration locale, et les réunions, multilogues, rajoutent à cela un certain niveau de déstructuration dû aux nombreuses interruptions mutuelles, discussions parallèles et en général à la complexité des tours de parole.

	Développement				Test			
	# d.	# m.	# h.	# q.	# d.	# m.	# h.	# q.
T1	10	68 541	8h45	50	15	57 133	5h45	100
T2	50	281 454	21h15	50	120	692 957	50h40	100
T3	6	35 328	2h15	50	12	87 147	5h40	100
T4	3	11 568	1h00	50	4	22 514	1h50	100
T5	1	13 355	1h10	50	4	20 007	1h40	100

TAB. 11.1 – Les corpus QAst : **# d. :** nombre de documents ;**# m. :** nombre de mots ; **# h. :** durée de parole. ; **# q. :** nombre de questions

Ces corpus ont été divisés en deux parties, une pour les questions de développement et une pour les questions de test. Les tailles de ces corpus sont indiquées tableau 11.1.

Chaque type de donnée, avec séparation par langue, correspond à une tâche, nommées T1 et T2 pour 2007 et de T1 à T5 pour 2008. En résumé :
– T1 correspond à Question-Réponse dans des séminaires en anglais.
– T2 correspond à Question-Réponse dans des réunions en anglais.
– T3 correspond à Question-Réponse dans des émissions d'informations radiophoniques en français.
– T4 correspond à Question-Réponse dans des sessions du Parlement Européen en anglais.
– T5 correspond à Question-Réponse dans des sessions du Parlement Européen en espagnol.
Afin de pouvoir estimer l'impact sur les résultats des erreurs de la transcription automatique, on consi-

dère pour chaque tâche deux *sous-tâches*, la sous-tâche *a* ayant lieu sur les transcriptions manuelles et les *b* sur la ou les transcriptions automatiques[1].

Type	Exemple
personne	Quel est le réalisateur du film Holy Lola ?
lieu	Où le Maréchal Lannes est-il né ?
organisation	À quel parti politique Ariel Sharon appartient-il ?
langue	En quelle langue la chaîne Al-Jazira est-elle diffusée ?
méthode/algorithme	Which windowing method is used for acoustic pre-processing ?
mesure	Combien de personnes sont mortes du SRAS ?
data/heure	Quand John Lennon fut-il assassiné ?
couleur	Quelle est la couleur du ciel autour du golfe du Lion ?
forme	What shape could the joystick have ?
matériau	What material can be flexible ?
définition	Qu'est-ce que le Patriot Act ?

TAB. 11.2 – Exemples de questions QAst

QAst 2007 traitait uniquement de questions factuelles dont les réponses devaient être des entités nommées de types prédéfinis : *personne*, *lieu*, *organisation*, *langue*, *méthode/algorithme*, *mesure*, *date/heure*, *couleur*, *forme* et *matériau*. L'édition 2008 rajoute les questions de définition simple. Le tableau 11.2 donne des exemples de ces questions. Les systèmes pouvaient donner jusqu'à 5 réponses par question, et environ 10% des questions n'avaient pas de réponse dans les documents.

Les réponses attendues doivent contenir l'identifiant de la question, le rang de la réponse, l'identifiant du document dans laquelle elle a été trouvée, son texte et optionnellement son indice de confiance. De plus, depuis 2008 et dans le cas des transcriptions automatiques, la réponse attendue n'est pas seulement un texte mais aussi un intervalle temporel du signal audio original contenant la réponse. Cet intervalle est considéré correct si ses bornes ne sont pas trop éloignées des bornes du ou des intervalles de référence. Cette référence est établie par alignement de la transcription manuelle et du signal audio d'origine, assurant ainsi des mesures précises. La figure 11.1 donne un exemple de question et de réponse attendue dans une transcription manuelle et une transcription automatique.

La campagne d'évaluation QAst 2007 a été l'occasion de notre première participation à une évaluation ouverte internationale. Nous y avons utilisé deux systèmes. Le premier, décrit dans la chapitre 7, *Une approche préliminaire pour Question-Réponse*, s'appuie sur des listes de requêtes possibles écrites à la main. Le second, décrit dans le chapitre 8, *Un système plus avancé*, s'appuie sur une abstraction de la recherche à effectuer et des algorithmes s'appuyant dessus, et constituait la toute première version du système actuel. Le tableau 11.3 donne les résultats obtenus au test officiel avec pour comparaison le résultat sur les questions de développement. La figure 11.2 permet de comparer ces résultats à ceux

[1]La terminologie utilisée pour QAst 2007 était en réalité 4 tâches nommées T1, T2, T3 et T4 correspondant respectivement à T1a, T1b, T2a et T2b. La généralisation en tâche/sous-tâche a été introduite en 2008. La terminologie 2008 sera utilisée pour les deux pour éviter toute confusion.

Question : *What is the Vlaams Blok ?*

Transcription manuelle : *the Belgian Supreme Court has upheld a previous ruling that declares the Vlaams Blok a criminal organization and effectively bans it .*
Réponse : *criminal organisation*

Extrait d'une **transcription automatique (format CTM) :**
(...)
20041115_1705_1735_EN_SAT 1 1018.408 0.440 Vlaams 0.9779
20041115_1705_1735_EN_SAT 1 1018.848 0.300 Blok 0.8305
20041115_1705_1735_EN_SAT 1 1019.168 0.060 a 0.4176
20041115_1705_1735_EN_SAT 1 **1019.228** 0.470 criminal 0.9131
20041115_1705_1735_EN_SAT 1 **1019.858 0.840** organisation 0.5847
20041115_1705_1735_EN_SAT 1 1020.938 0.100 and 0.9747
(...)
Réponse : 1019.228 1020.698 criminal organisation

FIG. 11.1 – Exemple de requête *What is the Vlaams Blok ?* et réponse dans une transcription manuelle (haut) et automatique (bas). Le format CTM est un format colonne contenant *identifieur de document*, *numéro de canal*, *position temporelle*, *durée*, *mot* et *score de confiance*.

des autres participants à l'évaluation.

Tâche	Système	Acc.	MRR	Top5	Acc. dev
T1a	Simple	32,6%	0,37	43,8%	74%
T1a	Avancé	39,7%	0,46	57,1%	94%
T1b	Simple	20,4%	0,23	28,5%	24%
T1b	Avancé	21,4%	0,24	28,5%	34%
T2a	Simple	26,0%	0,28	32,2%	28%
T2a	Avancé	26,0%	0,31	41,6%	72%
T2b	Simple	18,3%	0,19	22,6%	20%
T2b	Avancé	17,2%	0,19	22,6%	32%

TAB. 11.3 – Résultats de l'évaluation QAst 2007. *Acc.* est l'accuracy, *MRR* le Mean Reciprocal Rank, *Top5* (le "rappel"), le taux de réponses correctes quelles que soit leur rang. *Acc. dev* donne pour comparaison l'accuracy sur les données de développement.

L'amélioration du Top5 (9-13% absolus) observée sur les transcriptions manuelles montre bien que l'extraction explicite de passages suivi d'une extraction des candidats réponses qui y sont situés permet une bien meilleure couverture des réponses potentielles que la simple extraction de lignes des documents sur requête. Les variations d'accuracy sont moins claires, et en particulier nettement moins tranchées que sur les données de développement, allant d'un gain pour T1a à une perte pour T2b. L'hypothèse qui semble la plus probable est que la taille des données de développement était insuffisante, donnant lieu à une sur-spécialisation du système sur ces données et en conséquence une perte

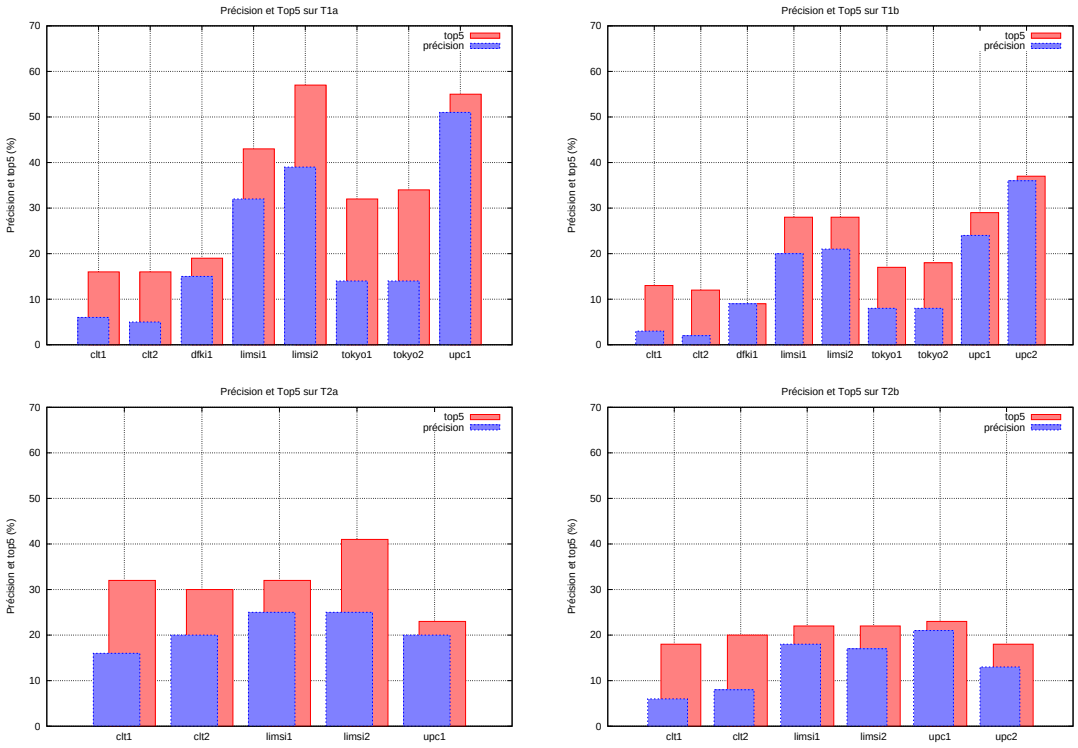


FIG. 11.2 – Résultats officiels pour la campagne d'évalution QAst 2007

de robustesse. Une étude plus poussée de ce genre de problèmes est décrite section 12 mais nous avons remarqué par exemple que toutes les questions de développement portant sur des systèmes ou des méthodes contenaient le mot *system* ou *method* ce qui n'était le cas d'aucune de celles du test. Du coup les règles de prédiction de types de réponses attendues s'appuyaient sur ces mots comme indices, ce qui a nuit à la qualité des résultats. Ce genre de problème est moins probable avec une plus grande variété de questions sur lesquelles s'appuyer.

Les résultats ont été bien meilleurs sur l'édition 2008 de cette évaluation. Il sont présentés dans le tableau 11.4. Comme nous avons vu précemment 16 sous-tâches étaient proposées. Nous étions les seuls participants pour les 4 sous-tâches du français (T3) et sommes arrivés premiers pour 8 des 12 autres.

Ces résultats ont été obtenus par la combinaison d'un gros travail d'amélioration de la qualité de l'analyse et de la génération des descripteurs de recherche, et en particulier la prédiction des types de réponses attendus, une augmentation de la taille des corpus (cf. chapitre 12) et de l'ajout dans les transformations du support des synonymes. Cet effort a été grandement facilité par la vitesse élevée du système permettant de nombreuses expériences comparatives, mesurant immédiatement l'impact de toute modification. De plus, le concept des *Descripteurs de Recherche* est à la fois puissant et

Tâche		Prec.	Meilleur
T1	manuel	41%	-
	ASR	27%	31% UPC
T2	manuel	33%	-
	ASR	16%	18% UPC
T3	manual	45%	-
	ASR A	41%	-
	ASR B	25%	-
	ASR C	21%	-

Tâche		Prec.	Meilleur
T4	manuel	33%	34% UPC
	ASR A	21%	30% INAOE
	ASR B	20%	-
	ASR C	19%	-
T5	manuel	33%	-
	ASR A	24%	-
	ASR B	19%	-
	ASR C	23%	-

TAB. 11.4 – Résultats officiels de l'évaluation QAst 2008. La colonne *Meilleur* indique le meilleur résultat et le système l'ayant obtenu le cas échéant.

simple à comprendre et facilite le diagnostic. Il est généralement possible pour une question donnée de comprendre pourquoi le système a privilégié une réponse donnée en comparant descripteurs et passages, donnant des idées de points sur lesquels agir au niveau de l'analyse de la langue pour améliorer les résultats.

Ces résultats montrent que les algorithmes et méthodes proposées ont des performances tout à fait honorables et sont d'autant plus mis en valeur que le travail spécifique à la langue est de qualité.

	ASR_A		ASR_B		ASR_C		MAN
	Prec.	WER	Prec.	WER	Prec.	WER	Prec.
T3	41%	11,0%	25%	23,9%	21%	35,4%	45%
T4	21%	10,6%	20%	14,0%	19%	24,1%	33%
T5	24%	11,5%	19%	12,7%	23%	13,7%	33%

TAB. 11.5 – **Résultats comparatifs sur transcription manuelle et automatique** pour T3, T4 et T5. *Prec.* : % taux de réponses correctes au premier rang. *WER* : Word Error Rate, taux d'erreur de mots

Un des buts de l'évaluation QAst était d'étudier l'impact des erreurs de la reconnaissance automatique sur les systèmes de Question-Réponse. Les résultats sont regroupés dans le tableau 11.5. Les mêmes systèmes étaient utilisés pour les transcriptions manuelles et automatiques, ces résultats nous donnent des indications sur la robustesse des systèmes. Là encore la plus grande maturité du système en français est évidente, avec une perte de seulement 4% absolus (9% relatifs) pour un taux d'erreur de mots de 11%, contre 9% absolus (27% relatifs) pour l'espagnol et 12% absolus (36% relatifs) pour l'anglais. Globalement un lien qualitatif apparaît entre taux d'erreur de mots et réussite du système, plus d'erreurs de transcription donnent généralement lieu à de plus mauvais résultats, ce qui était attendu. Cependant prévoir quantitativement la perte en fonction du taux d'erreur semble difficile. Il est probable que 100 questions et un petit nombre de documents sont insuffisants pour mesurer des différences fines.

11.2 La campagne d'évaluation Quaero

Le projet Quaero [Quaero 2008], qui inclut entre autres une tâche Question-Réponse, a organisé dans son premier semestre d'existence une évaluation *baseline* des systèmes des participants [Quintard 2009]. La qualification de baseline signifie que les systèmes devaient être testés en l'état autant que possible, une adaptation à la tâche restant cependant nécessaire. Le but était de savoir d'où nous partions afin de mesurer les progrès via les évaluations des années suivantes.

	Quaero	
Type	Web	
Langue	Français	Anglais
Nombre de documents	500K	500K
Nombre de phrases	82M	92M
Nombre de mots	840M	921M
Nombre de caractères	4,2G	4,9G
Phrases/document	170	180
Mots/phrase	10	10
Caractères/mot	5,3	5,3

TAB. 11.6 – Types et tailles de la collection de documents pour l'évaluation baseline Quaero.

L'organisation de l'évaluation a suivi des lignes traditionnelles. Elle a commencé par une définition du corpus de documents. Le but à terme étant d'aller plus loin dans le domaine des moteurs de recherche grand public, le corpus se devait d'être constitué de documents du Web. Exalead, qui est une entreprise française partenaire du projet et qui possède un moteur de recherche grand public en ligne[2], s'est chargé de cet aspect. Un ensemble de requêtes utilisateur a été collecté, et environ 500 000 documents potentiellement pertinents pour y répondre ont été sélectionnés. Le texte de ces documents a été extrait automatiquement pour constituer la collection. La même opération a été menée pour le français et l'anglais. Les caractéristiques finales de ces deux collections sont données tableau 11.6.

La définition des types de questions a été un compromis entre les types de questions qu'un humain pose naturellement et ce que nous pouvons évaluer. Comme nous l'avons déjà indiqué, [Kato et al. 2006] a observé dans le cas d'utilisateurs réels que 34% des questions consistaient de questions comment, pourquoi et définitions. Nous mêmes avons constaté que plus de 10% des questions étaient de type oui/non [Toney et al. 2008]. Finalement les partenaires se sont entendus sur 6 types de questions :

– Questions factuelles : *Quand Gorgoroth a-t-il eu des problèmes avec la police ?*
– Définitions : *Qu'est-ce que le racisme ?*
– Questions oui/non : *L'Aloe-Vera est-il un antioxidant ?*
– Pourquoi : *Pourquoi Michael Jackson a-t-il été poursuivi en justice en 2005 ?*
– Comment : *Comment retirer une tâche de vin rouge ?*

[2] http ://www.exalead.fr/

– Listes fermées : *Quels sont les six pays ayant fondé l'Union Européenne ?*

Ces différents types ont semblé un bon équilibre entre les besoins utilisateurs et ce que nous étions capables d'évaluer. Une période d'*adjudication* avait cependant été prévue après l'évaluation pour pouvoir rectifier les résultats de l'évaluation, et nous n'avons pas constaté de problèmes de désaccord quand à la correction ou non d'une réponse. Cela ne veut pas dire qu'il n'existait pas de réponse que l'on pourrait considérer limite dans les documents, juste que les systèmes ne les ont pas retournés comme réponse correcte. Le problème pourra donc se produire plus souvent à l'avenir avec l'amélioration des systèmes. De plus, toujours dans l'idée d'un application réelle, les systèmes devaient fournir une *justification* avec chaque réponse, une extrait de texte de 250 caractères maximum devant convaincre un utilisateur humain du bien fondé de la réponse.

Environ 130 questions par langue ont été fournies avant l'évaluation comme données de développement par un effort commun entre participants et évaluateurs. Les évaluateurs ont ensuite créé environ 250 questions par langue pour l'évaluation à partir des logs utilisateur qui avaient servi à construire la collection de documents. Au final, après adjudication, 256 questions ont été conservées pour le français et 242 pour l'anglais.

		Notre système	Système B	Système C
Français	Précision	19,3%	30,9%	11,9%
sans	MRR	0,204	0,338	0,143
justif.	Top3	23,1%	37,7%	19,6%
Français	Précision	18,9%	29,7%	11,9%
avec	MRR	0,200	0,304	0,139
justif.	Top3	22,7%	36,0%	19,1%
Anglais	Précision	9,1%	24,7%	14,1%
sans	MRR	0,114	0,266	0,152
justif.	Top3	15,2%	27,2%	17,9%
Anglais	Précision	9,1%	24,3%	14,1%
avec	MRR	0,110	0,259	0,152
justif.	Top3	15,2%	27,2%	17,9%

TAB. 11.7 – Résultat de l'évaluation Quaero, avec et sans prise en compte de la justification. Les systèmes B et C sont les systèmes des deux autres participants.

Les résultats, tableau 11.7, sont assez mitigés. Notre système était celui de l'évaluation QAst 2008 dont les résultats étaient présentés à la section précédente. Nous y avions ajouté les algorithmes présentés chapitre 9 pour pouvoir répondre aux questions autres que les factuelles simples. Nos résultats très moyens ont à première vue plusieurs causes. La première est un certain nombre de bugs dans le système, en particulier au niveau de la détection de type de question et au niveau de la prédiction de type de réponse qui nous ont coûté cher, particulièrement en anglais. Une seconde est l'absence en pratique de redondance. Les questions, construites à partir de logs utilisateurs mais en se référant aux documents, tendaient à être assez précises. La réponse se trouvait du coup à un très petit nombre d'endroits dans la collection. Des documents journalistiques par exemple ont tendance à se répéter

beaucoup plus, d'un jour sur l'autre et d'un journal à l'autre, tout en variant les formulations, rendant la recherche beaucoup plus facile. Mais la raison principale des faibles résultats reste la qualité des documents.

La collection de documents est, rappelons-le, un ensemble de pages web choisies à partir des logs utilisateurs. Exalead n'a effectué aucun filtrage particulier dans les pages choisies sauf celui de la langue, et même là de nombreuses erreurs sont présentes en particulier en anglais. En effet ne sont pas rares les pages ayant tout ou partie de leur navigation en anglais mais le contenu effectif dans une autre langue, trompant le système de sélection. Or une grande partie du web actuel consiste en des spams ou en général en des sites cherchant à tromper les moteurs de recherche et attirer les utilisateurs en contenant des listes de mots-clé les plus larges possibles. La collection de documents n'a pas été filtrée à ce niveau là. Et nos scores, en particulier les scores de documents et de passages, sont essentiellement basés sur des comptes d'occurrence des termes de la question. Ils sont donc particulièrement sensibles à ce genre d'action. Ils devront donc être modifiés pour pouvoir y résister.

L'autre aspect de la qualité des documents est la typographie. Dans des documents propres le placement des majuscules en particulier est un indicateur fort de la présence de noms propres et d'acronymes. Mais les documents venant du web sont, en moyenne, tout sauf propres. Nous allons devoir travailler plus avant sur le problème de la *normalisation*, qui cherche à régulariser les conventions typographiques utilisées dans les documents.

Chapitre 12

Impact de la taille des corpus de questions

Une de nos conclusions de l'évaluation QAst 2007 est que la robustesse du système, et donc ses performances au test officiel, était très dépendante de la taille et de la couverture des données de développement et que celles proposées étaient insuffisantes ([Rosset, et al. 2008] ainsi que section 11.1). L'édition 2008 de cette évaluation ne proposant là encore que 50 questions par tâche pour le développement (cf. tableau 11.1), nous avons décidé d'agir sur ce point.

La première étape a été de demander à des locuteurs natifs de construire des *reformulations* des questions de développement. Les questions d'origine, construites par des humains à partir des documents, ont tendance à reprendre les mots et tournures exactes qui y sont trouvés. Construire des reformulations sans consulter les documents permet de s'en éloigner et du coup de limiter ce biais. Les réponses restent en théorie les mêmes que pour les questions initiales, même si en pratique de légers glissements de sens demandent parfois à les adapter. Par exemple reformuler *En quelle année la Lituanie est-elle devenue indépendante ?* en *Quand la Lituanie est-elle devenue indépendante ?* ne change pas fondamentalement le sens mais demande de rajouter les dates complètes (11 mars 1990 vs. 1990 seul) comme réponses correctes.

Nous avons utilisé ces corpus de questions de développement (nommé *OffDev* par la suite) et de reformulations (*RefCorp*) directement pour l'amélioration des analyses, le choix des différents poids et le réglage des paramètres de fonctionnement du système en général. Cela en fait, dans le vocabulaire de la construction des systèmes statistiques, des corpus d'*entraînement*. Pour pouvoir avoir une idée des performances réelles, un vrai corpus de *développement* est nécessaire, qui n'est utilisé que pour contrôler l'effet général des évolutions et non pour des réglages fins. Nous avons donc dû construire un tel corpus, que nous avons nommé *Blind Corpus* ou *BlCorp*. Pour les tâches T1 et T2, déjà présentes en 2007, nous avons utilisé le test 2007 comme corpus. Pour T3 à T5 nous avons récupéré des documents similaires à ceux présents dans les corpus officiels et avons demandé aux mêmes natifs de

créer de nouvelles questions et reformulations à partir de ces documents. Les tailles finales de tous ces corpus sont données dans le tableau 12.1.

	OffDev				RefCorp			
	# q.	# d.	# m.	# h.	# q.	# d.	# m.	# h.
T1	50	10	68 541	8h45	565	10	61 025	5h45
T2	50	50	281 454	21h15	587	50	281 454	21h15
T3	50	6	35 328	2h15	350	6	35 328	2h15
T4	50	3	11 568	1h00	277	3	11 568	1h00
T5	50	1	13 355	1h10	217	1	13 355	1h10

	BlCorp				Test			
	# q.	# d.	# m.	# h.	# q.	# d.	# m.	# h.
T1	100	15	63 526	8h30	100	15	57 133	5h45
T2	100	118	692 957	50h40	100	120	692 957	50h40
T3	248	3	44 048	3h00	100	12	87 147	5h40
T4	186	3	11 568	1h00	100	4	22 514	1h50
T5	36	2	47 721	2h25	100	4	20 007	1h40

TAB. 12.1 – Les corpus : *OffDev* : données de développement officielles ; *RefCorp* : questions de développement reformulées ; *BlCorp* : données de test 2007 pour T1 et T2, données construites à la main sur d'autres documents pour T3 à T5 ; *Test* : données officielles de test. **# q. :** nombre de questions ; **# d. :** nombre de documents ;**# m. :** nombre de mots ; **# h. :** durée de parole.

De façon à mesurer l'impact effectif de ces corpus sur les résultats nous avons repris le système d'analyse et la génération de DDR correspondante d'avant que le corpus de reformulations soit constitué et l'avons comparé avec le système final, en séparant l'aspect optimisation automatique de paramètres numériques, permettant de différencier les parties liées à l'expertise humaine (analyse et DDR) des parties automatiques. Les résultats sont regroupés dans le tableau 12.2.

Ces résultats montrent que l'impact de l'amélioration de l'analyse et de la génération des DDR est très significative, avec un gain absolu sur le test de 9% (T5) à 24% (T1). Avoir un plus grand nombre d'exemples avec lesquels travailler est en pratique aussi important pour un humain écrivant des règles à la main que pour un système statistique. L'intuition a des limites.

La forte perte entre le *Blind Corpus* et le test officiel pour les tâches T1 et T2 s'explique par un mismatch entre les données de développement et de test. Le blind corpus suit les catégories de questions des données de développement, mais celles-ci n'avaient pas changé depuis 2007. Or les catégories, elles, avaient changé, en particulier avec l'addition de nombreuses questions de définitions (environ 25% des questions du test). Ces changements manquaient donc à la fois dans les données de développement et dans le blind corpus. Le travail s'est donc fait suivant les lignes de l'évaluation 2007 et l'utilisation du blind corpus n'a pas permis de s'apercevoir du problème.

	T1		T2		T3		T4		T5	
	BlCorp	Test	BlCorp	Test	BlCorp	Test	BlCorp	Test	BlCorp	Test
Original	45,9%	29%	37,5%	29%	29,0%	40%	15,1%	25%	11,1%	22%
Paramétrage sur tout	54,1%	33%	45,8%	30%	29,0%	41%	15,1%	31%	11,1%	24%
Analyse sur tout	64,3%	44%	49,0%	33%	40,3%	45%	24,2%	28%	25,0%	29%
Tout sur tout	64,3%	41%	49,0%	33%	41,5%	45%	26,9%	33%	36,1%	33%

TAB. 12.2 – Résultats comparatifs obtenus sur le *Blind Corpus* (BlCorp) et le test officiel (Test). Le système original est construit uniquement sur le corpus de développement officiel. À partir de là l'optimisation du paramétrage pour être effectué sur l'ensemble dev+reformulations, l'analyse et les poids dans la génération des DDR peut être travaillé sur le même ensemble, ou les deux à la fois. Les valeurs sont les précisions (accuracy).

En contrepartie pour T3 à T5 nous pouvons voir que ces blind corpus sont légèrement plus difficiles que le corpus de test officiel. Cela en fait des bons guides pour notre travail. L'utilisation de reformulations dans leur construction a assuré que la plupart des questions ne reprenaient pas directement les formulations des documents. La légère perte observée sur T5 peut probablement être expliquée par la relative petitesse de ce corpus spécifique.

En ce qui concerne l'optimisation des paramètres, nous pouvons voir qu'en général plus de données résulte en un gain, comme attendu. T1 est une exception, qui est probablement expliquée par le mismatch précédemment décrit. La magnitude du gain est cependant peu prévisible. Cela laisse à penser que la méthode d'estimation des scores de réponse gagnerait à être retravaillée pour assurer une meilleure robustesse.

Chapitre 13

Résultats individuels par modules

13.1 Impact de l'analyse et de la Recherche d'Informations

Le système Question-Réponse s'appuie sur 5 étapes principales : l'analyse de la langue, la génération d'un Descripteur De Recherche et en particulier la prédiction des types de réponse, la sélection des documents, l'extraction des passages et finalement l'évaluation des candidats réponses. Il est intéressant de mesurer pas à pas l'impact de ces différents modules. Le tableau 13.1 regroupe un ensemble de résultats individuels sur les données QAst 2008 (transcription manuelle) et Quaero en français, obtenus sur les questions factuelles ayant une réponse dans la collection uniquement (d'où les pourcentages différents des résultats officiels).

Rappelons les caractéristiques de ces évaluations. Les tâches T1 à T5 de QAst s'appuient sur des transcriptions de parole de natures diverses. T1 et T2 s'intéressent à l'anglais technique spontané, dans le cadre de séminaires pour T1 (un seul locuteur) et de réunions de travail pour T2 (locuteurs multiples). T3 couvre le français préparé d'émissions d'informations radiodiffusées. Enfin T4 et T5 couvrent la parole préparée des débats du parlement européen, en anglais pour T4 et espagnol pour T5. L'évaluation Quaero s'appuie sur une collection de documents (500 000) récupérés du Web.

Pour interpréter ce tableau, il faut tout d'abord se rappeler que seules les entités annotées par l'analyse sont considérées comme réponses potentielles. Par exemple si une réponse attendue est *centre marocain de la gouvernance* mais que l'analyseur ne détecte que *centre marocain* la réponse attendue ne sera jamais candidate. C'est ce que mesure la ligne *Entité*, qui permet ainsi de voir le maximum absolu en performance que l'on peut obtenir pour une analyse donnée. L'algorithme d'extraction fait cependant aussi un filtrage sur le type de l'entité : seules sont prises en compte celles d'un des types prédits pour la réponse. Par exemple, pour la question *De quel organisme Driss Abbudi est-il président ?* le système prédit que la réponse doit être une organisation. Si, par erreur, l'analyse a annoté *centre marocain de la gouvernance* comme un lieu, le système ne pourra trouver la bonne réponse.

	QAst					Quaero
	T1	T2	T3	T4	T5	
Entité	83,3%	90,9%	92,0%	88,8%	85,4%	87,8%
En+Type	60,0%	59,1%	87,4%	71,9%	66,3%	63,4%
Document	100,0%	100,0%	100,0%	100,0%	100,0%	75,6%
Passage	77,8%	72,7%	86,2%	66,3%	65,2%	56,7%
Pass+En	67,8%	67,0%	80,5%	66,3%	59,6%	51,2%
Pass+En+Type	53,3%	54,5%	75,9%	58,4%	49,4%	36,0%
Précision	38,9%	31,8%	49,4%	34,8%	33,7%	23,8%

TAB. 13.1 – Évaluation modulaire sur questions factuelles uniquement. **Entité :** la réponse attendue est en une seule entité pour l'analyse. **En+type :** la réponse attendue est en une seule entité et son type fait partie des types prédits pour la réponse dans le DDR. **Document :** la réponse attendue apparaît dans les documents choisis. **Passage :** la réponse attendue apparaît dans les passages extraits. **Pass+En :** la réponse attendue apparaît en une seule entité dans les passages. **Pass+En+Type :** la réponse attendue apparaît en une seule entité dans les passages avec un type prédit dans le DDR. **Précision :** la réponse attendue est donnée au premier rang.

Cette deuxième condition est mesurée dans la ligne *En+Type*.

Le système fonctionne globalement par filtrage. La première étape est une sélection d'un sous-ensemble de documents. Dans le cas de QAst cette sélection n'est pas faite, le nombre de documents étant très faible. Mais dans le cas de Quaero il s'agit de sélectionner un maximum de 300 documents parmi une collection en contenant 500 000. La ligne *Document* indique le niveau de succès de cette étape en vérifiant si la réponse est présente dans les documents sélectionnés. Après la sélection des documents, des passages en sont extraits. La qualité de ces passages est mesurée suivant le même critère dans la ligne *Passage*.

Les deux lignes suivantes, *Pass+En* et *Pass+En+Type* croisent l'extraction de passages avec l'analyse. La première, croisant *Passage* et *Entité* indique si la réponse existe en tant qu'entité dans les passages extraits. La seconde croise *Passage* et *En+Type* et indique si la réponse existe dans les passages non seulement en tant qu'entité mais aussi d'un type attendu pour la réponse.

Enfin la dernière ligne, *Précision*, donne la performance finale du système complet.

Que nous indiquent ces chiffres ? Le plus simple est de regarder les pertes successives causées par chaque étape, partant d'un idéal de 100%. En premier lieu, on peut constater que le choix des frontières d'entités par l'analyse occasionne une perte de 8% (T3) à 16,7% (T1) (ligne *Entité*), ce qui est relativement peu et justifie de ne considérer que les entités comme réponses potentielles. La prédiction de type de réponse fait perdre de 4,6% (T3) à 31,8% (T2) de plus (ligne *En+Type*). Le meilleur résultat de ces deux étapes (8% et 4,6% pour un total de 12,6%) est obtenu pour les questions sur

les informations en français (T3). Les données sont relativement propres et le français est la langue pour laquelle l'analyseur a été développé depuis le plus longtemps. Dans des conditions similaires, les analyseurs anglais (T4) et espagnol (T5), plus récents, atteignent 28,1% et 33,7% de perte cumulée respectivement. Il est à noter que ces pertes combinent trois types d'erreurs : problèmes de frontières d'entités, problèmes de typages d'entités et problèmes de prédiction de type attendu.

Toujours pour *EN+Type*, la perte cumulée de 40% et 40,9% pour T1 et T2 (séminaires et réunions en anglais), plus importante d'environ 12% que T4 (anglais général) tout en étant la même langue, peut s'expliquer par deux facteurs : couverture de l'analyseur et mismatch développement/test. Le vocabulaire technique du traitement de la langue dans T1 a entraîné des problèmes de frontières et de typage. Il en est de même dans T2 pour les locutions variées désignant des formes et couleurs. Les questions très différentes du test par rapport au développement, qui avaient été constituées à un an d'écart par deux personnes différentes et dans un anglais parfois approximatif ont mis à rude épreuve la prédiction de type de réponse attendue.

Enfin dans le cas de Quaero la perte est là aussi importante à 36,6%, et est très probablement due à la nature des documents. Le Web tout-venant est particulièrement inconsistant à tous les niveaux, qualité de langue, orthographe, typographie, qualité informationnelle, etc. Cela pose bien des problèmes et en particulier au niveau du typage des entités. Des efforts sont d'ailleurs prévus au niveau de la normalisation des documents pour améliorer la situation.

La qualité de la recherche d'informations, ligne *Passage*, dépend, dans le cas des données QAst (où le faible nombre de documents fait qu'ils sont tous conservés), de la capacité du système à reconnaître les éléments importants de la question dans le voisinage de la réponse. Les résultats n'ont rien de surprenant, même s'ils gagneraient à être améliorés. Le système le plus abouti (T3, français) a les meilleurs résultats, les systèmes plus récents (T4, anglais, et T5, espagnol) sont moins bons. La perte plus faible pour T1 et T2 (anglais technique) par rapport à T4 (anglais général) s'explique probablement par la spécificité des thèmes abordés donnant lieu à l'utilisation de termes et locutions du domaine se retrouvant à l'identique dans les questions.

Dans le cas de Quaero, la recherche d'informations dépend d'un facteur de plus : la capacité du système à choisir les documents les plus pertinents dans la grande masse disponible (environ 500 000). Ce facteur, cumulé avec les problèmes de typage d'entité déjà évoqués, rend la reconnaissance des entités des questions plus difficile. Nous pouvons constater que 24,4% de pertes sont dues à la sélection des documents (ligne *Document*). À cela s'ajoutent 18,9% de pertes dues à la sélection des passages. En comparaison T3, informations en français sans sélection de documents, perd 13,8% au niveau de la sélection de passages. Cette sélection est particulièrement difficile dans le cas de Quaero où sont présentes dans la collection de documents de nombreuses pages non pertinentes pour la question qui essaient cependant d'attirer les moteurs de recherches (pages de sites pornographiques ou publicitaires en particulier) avec des listes de mots ou de noms. L'algorithme de sélection des pages, s'appuyant sur de simples comptes d'occurrences, y est particulièrement sensible. Des efforts devront être faits pour améliorer cela, une piste est de mesurer la distance de la page à des données journalistiques via des mesures de perplexité calculées avec des modèles de langages appropriés et de favoriser

les pages proches de ce type de langue, les considérant généralement plus informatives.

La dernière étape est l'évaluation des candidats réponse, qui nous amène au score final, la *précision*. La perte associée, successive à l'étape précédente *Pass+En+Type*, varie de 30% à 40%. Deux facteurs entrent en jeu : la capacité du système à reconnaître les éléments de la question mais aussi l'algorithme de score lui-même. La méthode de calcul proposée a ses limites. En effet la présence de deux entités proches n'indique pas toujours une relation positive entre elles, et une simple notion de distance est dans tous les cas trop limitée. On peut imaginer que des informations linguistiques plus avancées telles que des relations sémantiques entre les entités pourraient permettre d'obtenir de meilleures performances sans influer de manière notable sur la vitesse du système.

13.2 Impact des erreurs de transcription sur les différents modules

La table 13.2 présente les résultats par composant en suivant la même approche que la section précédente sur les transcriptions automatiques de QAst 2008 en comparaison à ceux obtenus sur les transcriptions manuelles. Ces mesures sont là encore limitées aux questions factuelles.

		WER	Entité	En+Type	Passage	Pass+En+Type	Précision
T3	Manuel		92,0%	87,4%	86,2%	75,9%	49,4%
	ASR_A	11,0%	97,7%	86,2%	59,8%	55,2%	41,4%
	ASR_B	23,9%	97,7%	86,2%	37,9%	35,6%	24,1%
	ASR_C	35,4%	97,7%	78,2%	29,9%	27,6%	19,5%
T4	Manuel		88,8%	71,9%	66,3%	58,4%	34,8%
	ASR_A	10,6%	97,8%	67,4%	40,4%	33,7%	20,2%
	ASR_B	14,0%	97,8%	64,0%	34,8%	30,3%	18,0%
	ASR_C	24,1%	98,9%	61,8%	30,3%	25,8%	16,9%
T5	Manuel		85,4%	66,3%	65,2%	49,4%	33,7%
	ASR_A	11,5%	95,5%	66,3%	51,7%	38,2%	23,6%
	ASR_B	12,7%	92,1%	61,8%	53,9%	38,2%	20,2%
	ASR_C	13,7%	95,5%	64,0%	46,1%	33,7%	22,5%

TAB. 13.2 – Évaluation modulaire sur questions factuelles uniquement sur transcriptions manuelles et automatiques. **WER :** taux d'erreur de mots des transcriptions automatiques. **Entité :** la réponse attendue est en une seule entité pour l'analyse. **En+type :** la réponse attendue est en une seule entité et son type fait partie des types prédits pour la réponse dans le DDR. **Passage :** la réponse attendue apparaît dans les passages extraits. **Pass+En+Type :** la réponse attendue apparaît en une seule entité dans les passages avec un type prédit dans le DDR. **Précision :** la réponse attendue est donnée au premier rang.

La valeurs élevées de la colonne *Entité* pour les transcriptions automatiques sont un artefact de la méthode d'évaluation. Le système doit donner un intervalle temporel comme réponse et une marge

d'erreur est autorisée sur les bornes (cf. section 11.1). En conséquence n'importe quelle suite de mots de la transcription automatique commençant et finissant à l'intérieur des marges doit être considérée correcte. Du coup la réponse peut souvent être réduite a un simple mot, parfois même un mot vide (déterminant, préposition, ...), qui est annoté en tant que tel par l'analyse. Le score *Entité* est ainsi artificiellement augmenté.

Ajouter la corrélation avec les types d'entité attendus pour la réponse, colonne *En+Type*, donne des informations plus intéressantes quand aux pertes dues aux erreurs de transcription automatique. Nous pouvons voir que les systèmes français et espagnols sont plutôt robustes, arrivant à annoter les entités avec les types corrects attendus la plupart du temps même en présence d'erreurs dans les entités ou leur contexte (environ 1% d'erreur pour environ 11% de WER). Le système anglais est légèrement moins robuste ce qui explique sa perte plus élevée (environ 4% pour un WER similaire).

La plus grande perte a lieu au niveau de l'extraction des passages, colonnes *Passage* et *Pass+En+Type*. Extraire les passages, et ensuite évaluer les candidats réponse, dépend, comme nous l'avons vu, de la capacité du système à reconnaître les éléments des questions dans le document, avec des possibilités de transformations entre les deux pour améliorer la couverture. Même si les résultats *En+Type* montrent que les types sont en grande partie conservés, les valeurs elles changent avec les erreurs, faisant échouer les correspondances. Cette conservation des types laisse à penser qu'il serait intéressant d'ajouter une tranformation basée sur la phonétique permettant des correspondances plus lâches, quitte à lui donner un poids plus faible au niveau des DDR.

Ces résultats montrent que le lien augmentation du WER et augmentation des pertes reste présent à tous les niveaux. Pour le cas étrange des ASR B et C en espagnol, où la différence entre les résultats (20,2% contre 22,5% de précision) est inversée par rapport à la différence de taux d'erreur (12,7% contre 13,7% de WER), un indice semble être présent dans la colonne *En+Type*, avec des valeurs de 61,8% contre 64,0%. Il semblerait indiquer que les réponses attendues aux questions du test se sont trouvées être plus touchées par les erreurs de reconnaissance du système B que du système C. Cependant, le faible nombre de questions (100) semble être insuffisant pour tirer de ces relativement faibles différences des conclusions significatives statistiquement.

Chapitre 14

Equilibre vitesse - performance

Un des buts du système de Question-Réponse que nous avons décrit est de pouvoir contrôler la vitesse de réaction du système. Cela se fait via deux paramètres, le nombre maximal de documents retenus et le nombre maximal de candidats réponse évalués. Nous allons évaluer l'influence de ces deux paramètres sur les performances du système, en termes de MRR pour la qualité des réponses et de temps moyen par question pour la vitesse.

Nous avons fait ces expérimentations sur deux jeux de données :
– Clef : la collection QA@Clef en français et les 400 questions des évaluations 2004 et 2005.
– Web : Documents récupérés du Web et 100 questions construites à partir du corpus Ritel [Rosset & Petel 2006].

La collection de documents Clef comprend les années 1994 et 1995 du journal *Le Monde* ainsi que les dépêches de l'agence de presse suisse *ATS*. Les documents portent chacun sur un seul sujet et leur taille varie de 3 octets à 95Ko suivant une courbe exponentielle inverse (figure 14.1). La taille totale est légèrement en dessous de 400Mo pour 177 000 documents. Les questions et réponses de référence sont celles des évaluations Question-Réponse de Clef des années 2004 et 2005.

La collection Web a été construite en 2006 en récupérant les (approximativement) 1 000 premières pages retournées par Altavista sur une série de requêtes construites autour d'environ 10 thèmes. La taille de ces pages HTML, une fois le texte extrait, varie de un octet à 33Mo, avec une distribution en loi de puissance tel que nous pouvons voir figure 14.2. Nous avons supprimé de la collection tous les documents plus grand que 1Mo pour ces évaluations, conservant cependant une taille moyenne bien plus élevée que pour Clef avec un total de 5Go de texte pour environ 62 000 documents. Comparant avec la collection Quaero, construite en 2008 par des moyens similaires avec le moteur d'Exalead, nous avons constaté que les documents de notre collection Web étaient en moyenne beaucoup plus propres et moins spammés. Cela s'explique probablement par deux facteurs : la collection Quaero a été constituée par Exalead eux-mêmes sans appliquer leurs filtres anti-spam, alors que ce type de filtre est appliqué par l'interface web d'Altavista que nous avions utilisé, et le web français en 2006 était

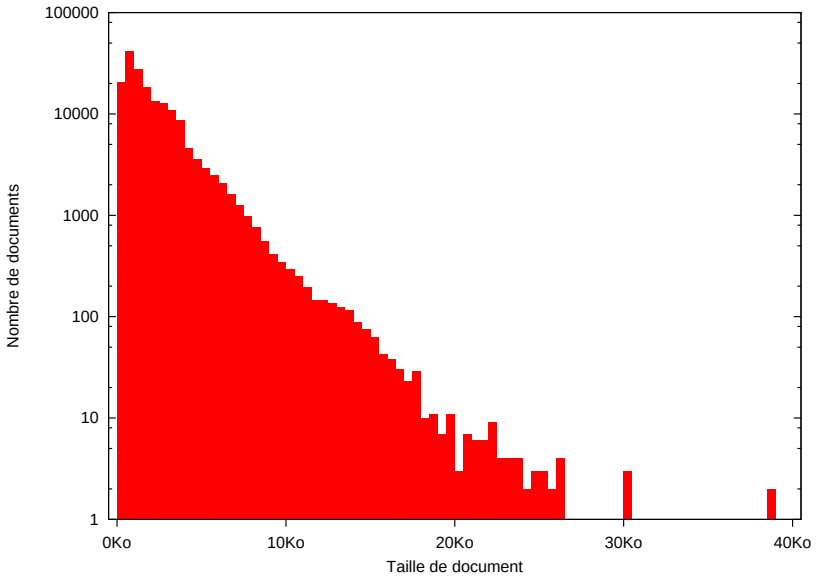


FIG. 14.1 – Distribution des tailles de documents pour la collection Clef

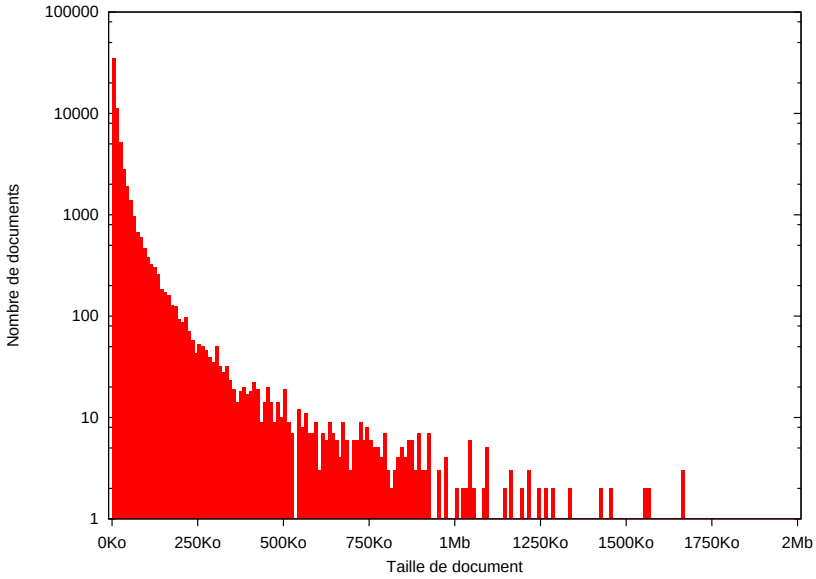


FIG. 14.2 – Distribution des tailles de documents pour la collection Web

probablement intrisèquement plus propre, mais moins riche, qu'en 2008, son développement étant rapide. Les questions associées sont des vraies questions utilisateurs, parfois reconstruites d'après l'historique du dialogue, qui ont été posées oralement et spontanément par des utilisateurs testeurs.

Nous avons ensuite cherché à la main les réponses dans les documents. Cette procédure demande beaucoup de travail mais assure que le biais habituel apparaissant quand des questions sont construites à partir de documents soit cette fois-là absent.

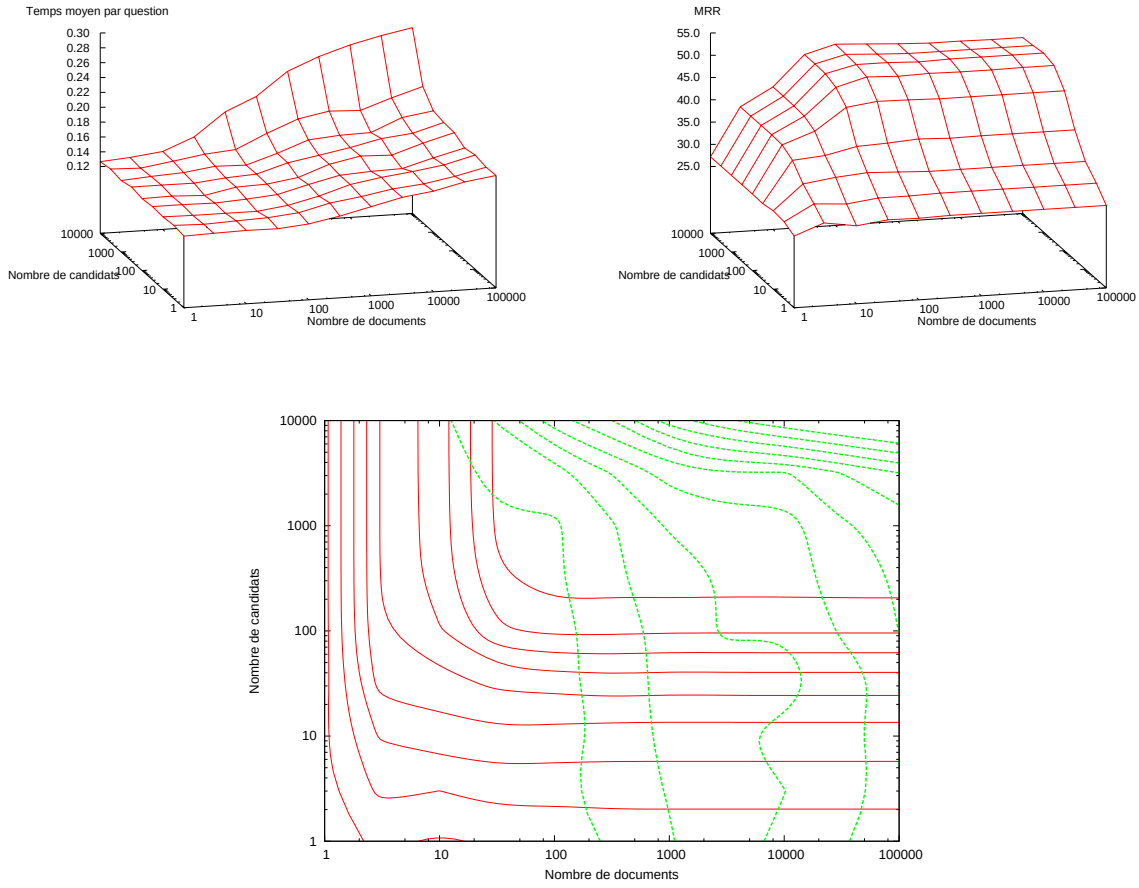


FIG. 14.3 – Clef : Temps par question en secondes, MRR et lignes de contour. Dans le graphe de contours, chaque ligne rouge correspond à une perte de 5% absolus par rapport au MRR maximal, et chaque ligne verte à un ralentissement de 10%.

Pour chacune des collections nous avons évalué le temps moyen par question et le MRR en fonction des valeurs des deux paramètres contrôlant la vitesse, le nombre maximal de documents retenus et le nombre maximal de candidats réponse examinés. Les courbes pour la collection Clef sont figure 14.3 et celle de la collection Web figure 14.4. Les courbes de MRR sont similaires pour les deux collections : elle grimpent rapidement vers un plateau qui représente la performance maximale du système. Les courbes de temps sont très raisonnables : elles démarrent à un niveau non-compressible qui représente le temps pris par la sélection des documents. On voit d'ailleurs là une différence fondamentale entre les deux collections. Les documents Clef sont courts et monothématiques. En conséquence le nombre de documents contenant les éléments importants d'une question donnée est en général

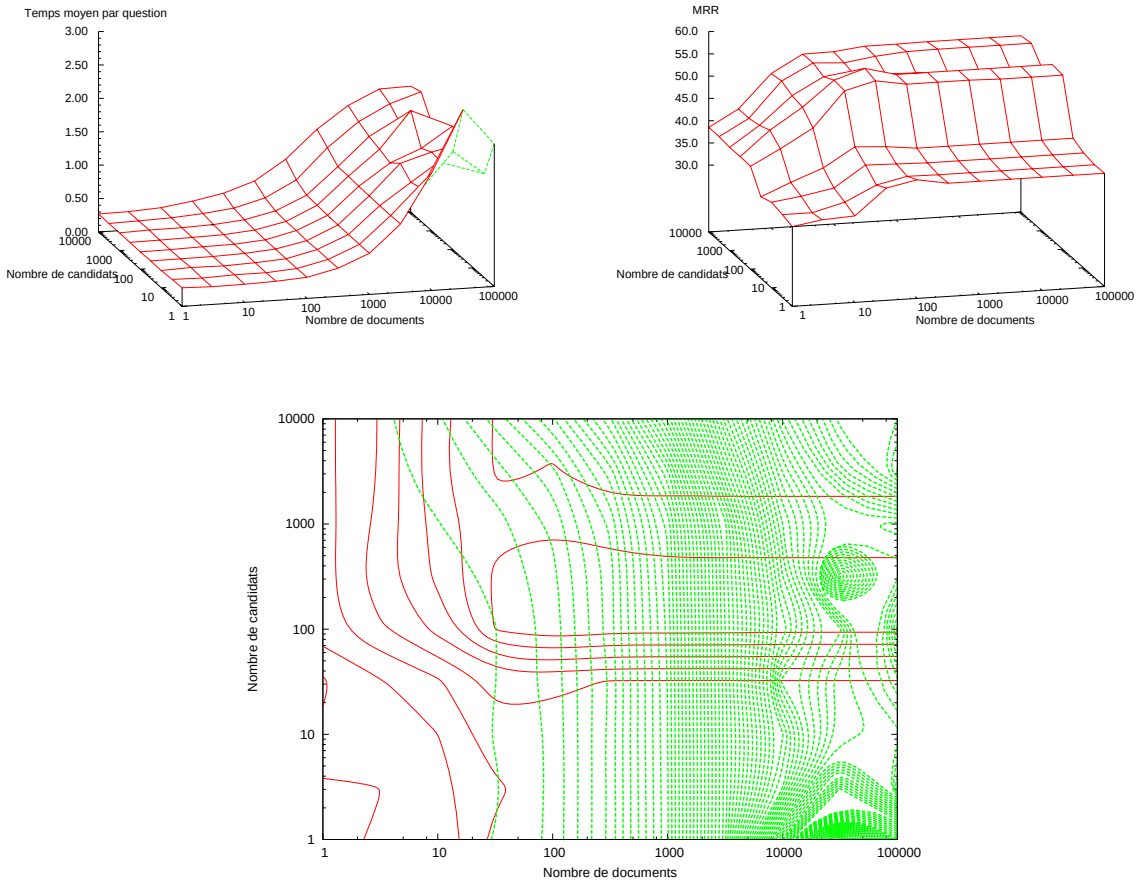


FIG. 14.4 – Web : Temps par question en secondes, MRR et lignes de contour. Dans le graphe de contours, chaque ligne rouge correspond à une perte de 5% absolus par rapport au MRR maximal, et chaque ligne verte à un ralentissement de 10%.

relativement faible. Par contre la collection Web est constituée de gros documents, parfois multithématiques. En conséquence beaucoup plus de documents sont pertinents pour une question donnée. Cela rend la sélection des documents deux fois plus lente pour le Web que pour Clef bien que la collection contienne trois fois moins de documents. Ces courbes de temps augmentent relativement lentement ensuite avec le nombre de documents et le nombre de candidats examinés. Dans le cas du Web le temps est dominé largement par le nombre de documents, ce qui n'est pas étonnant vu la taille moyenne bien supérieure des documents.

Le point important, montré par les courbes de contour, est que le plateau du MRR est atteint avant que les temps augmentent de façon significative, permettant de choisir pour chaque collection un point de fonctionnement où les résultats sont maximaux alors que le temps moyen n'est que 5 à 15% au dessus du minimum absolu. Les temps résultants, environ 0,3s par question pour le corpus

Web et 0,13s pour Clef, sont raisonnables pour une système interactif réactif. Ces résultats montrent cependant que l'amélioration de ces temps devra passer par une accélération de l'étape de sélection des documents, ce qui n'est pas a priori simple.

Discussion

Nous avons présenté des évaluations de systèmes construits sur la base des algorithmes proposés suivant plusieurs angles. Les évaluations Question-Réponse officielles permettent d'obtenir une vue globale des performances de l'ensemble du système. L'étude de l'impact de la taille des corpus donne une idée de la quantité de données nécessaire pour l'apprentissage non seulement des paramètres du système mais aussi des linguistes construisant l'analyse et les règles de classification de questions. Les résultats détaillés par module permettent de diagnostiquer les points forts et points faibles afin d'orienter les travaux de recherche futurs. Enfin l'étude de l'équilibre vitesse-performance permet de choisir un point de fonctionnement optimal pour le système étant donné les contraintes matérielles imposées par le contexte expérimental.

Ces évaluations ont ainsi permis de mettre en évidence des forces et des faiblesses de l'ensemble. Une grande force est la vitesse élevée de tous les composants du système liée à la flexibilité de ce que l'on peut lui demander. Il est très facile de modifier l'analyse, les règles de classification de questions ou encore les transformations et un nouveau résultat est rapidement obtenu. Prenons le cas des anciennes évaluations QA@Clef par exemple. Avec ses 400M de documents et 200 questions on peut la considérer de taille moyenne. L'analyse et l'indexation de l'ensemble des documents prend moins de deux heures, un tuning complet (57024 runs factorisés au mieux) environ une demi-heure, et la recherche des réponses pour l'ensemble des données de développement vingt à trente secondes. Et il n'est évidemment pas nécessaire de refaire la totalité des étapes à chaque fois. Il est donc possible d'avoir une démarche expérimentale, où l'on teste très souvent l'impact des modifications effectuées, permettant d'avoir une bonne idée de ce qui améliore le résultat ou le dégrade. De plus les *descripteurs de recherche* sont une représentation intermédiaire synthétique donnant un point de contrôle intéressant sur le comportement du système.

Il est intéressant de noter aussi que nous avons atteint nos objectifs fonctionnels. Le système est capable d'intégrer des éléments complémentaires venant du gestionnaire de dialogue via son algorithme de génération des Descripteurs de Recherche. Sa vitesse est suffisante pour un cadre interactif. De plus le typage de la réponse ainsi que la présence des éléments pertinents dans le DDR facilite la construction de la réponse à donner à l'utilisateur. Le système est ainsi réellement utilisé dans un cadre de dialogue en domaine ouvert avec des résultats intéressants, mais sortant du cadre de cette thèse. Le lecteur intéressé peut consulter [Toney et al. 2008] par exemple.

Les approches proposées ont cependant des faiblesses. La première tourne autour de l'évaluation des réponses. La méthode de scoring est construite sur une combinaison d'indices pertinents, et les résultats sont acceptables. Cependant l'évaluation modulaire montre qu'elle gagnerait à être plus robuste et à moins dépendre de paramètres arbitraires. Nous pensons qu'une approche probabiliste, comme évoquée dans la conclusion de la partie précédente, pourrait être plus robuste.

Le seconde faiblesse, que nous avons vue dans les résultats de l'évaluation officielle Quaero, est la sensibilité des scores de document et de passage à la pollution voulue qu'est le spam. Il semble donc indispensable dans les cas où ce genre de problème se pose d'étendre le score pour tenir compte de la qualité informative intrinsèque des documents.

Mais la principale faiblesse est la limite sur ce que l'analyse peut représenter. Comme nous l'avons vu dans la première partie la représentation ne permet pas d'annoter les relations à longue distance. Or nous pensons que des relations sémantiques de qualité liant les éléments trouvés par l'analyse permettraient d'obtenir une bien meilleure qualité au niveau du score de réponse. Cependant, représentation mise à part, poser de telles relations de façon fiable semble très difficile sans de grandes ressources linguistiques. Concevoir un moteur de règles capable de travailler sur de telles relations, écrire des règles pour les établir d'une manière fiable et les exploiter ensuite dans un système Question-Réponse restent des problèmes ouverts.

Quatrième partie

Conclusions et perspectives

Chapitre 15

Conclusions

Le travail que nous avons présenté s'inscrit dans le cadre d'un project de recherche d'informations interactive. Le projet Ritel [Ritel 2007 ; Rosset et al. 2006], qui a vu le jour au LIMSI en 2004, a pour objectif de construire une plateforme de dialogue permettant d'assister un utilisateur dans diverses tâches dont, en premier lieu, la recherche d'informations en domaine ouvert.

Passer en domaine ouvert pose de nombreuses difficultés. Une des questions à se poser est *quels types d'information est-on capable de rechercher en dehors d'une tâche précise*. Depuis la fin des années 90 une sous-partie du domaine de la Recherche d'Informations se développe, les *Systèmes de Réponse à des Questions*, ou *Système Question-Réponse* pour faire court. Le but est de répondre au mieux à des questions posées en langue à partir d'une base de documents. Cependant il existe bien des types de questions, définitions, pourquoi, comment... Un type spécifique qui nous intéresse pour plusieurs raisons est les questions précises qui demandent une réponse précise tenant en peu de mots, souvent qualifiées de *questions factuelles*. Par exemple la question *Qui a été élu président des États-Unis en 2008 ?* attend la réponse *Barak Obama*, pas plus, pas moins.

Ces questions ont trois avantages : le premier est qu'elles sont raisonnablement faciles à évaluer. Décider si une réponse donnée par un système est correcte ou non pose rarement problème. *Le 44e président des États-Unis* ou encore *le candidat démocrate*, pourraient certes poser problème en théorie, mais la réaction humaine immédiate *oui mais c'était qui ?* tend à diminuer la discussion pour les classer insuffisantes, et donc incorrectes. En comparaison une question de définition telle que *qu'est-ce qu'une OPA hostile ?* pose un bien plus gros problème d'évaluation : une réponse doit-elle contenir la définition d'une OPA ou les particularités spécifique des hostiles sont elles suffisantes ? Un deuxième avantage, lié au premier, est qu'elles ont été bien plus étudiées et les systèmes tentant d'y répondre peuvent participer à des évaluations internationales reconnues qui permettent d'avoir une idée de où on se trouve par rapport à l'état de l'art. Enfin le dernier point est spécifique à l'interaction, et en particulier à l'interaction orale : des réponses attendues courtes permettent au système de répondre de façon efficace et même, dans le cas de documents audio, de rejouer les bouts de documents associés.

Et ce sans ennuyer l'utilisateur ou avoir besoin de construire un résumé synthétique des informations trouvées, ce qui est un problème de recherche à part entière.

Nous nous sommes donc intéressés aux systèmes *Question-Réponse* (QR) pour pouvoir les utiliser dans un cadre interactif. Cependant ce cadre apporte des contraintes propres. La première, évidente, est un besoin de contrôler le temps de réponse. En effet un utilisateur est impatient par définition, il ne faut donc pas le faire attendre, particulièrement quand l'interaction se fait via le téléphone. Or le téléphone, par sa facilité de mise en œuvre, est un des vecteurs de communication orale privilégiés par le projet Ritel. Et le problème de la vitesse a été très peu étudié dans le cadre des systèmes QR. La seconde contrainte est plus subtile : la question ne se limite pas à une phrase bien construite. En effet une interaction inclut un contexte, et une demande de l'utilisateur se doit d'être interprétée en fonction de ce contexte. En pratique cette interprétation prend la forme d'entités complétant la demande, entités sélectionnées par le gestionnaire de dialogue, partie de système chargée de la gestion de l'interaction. Nous avons donc besoin d'une certaine flexibilité de l'entrée.

Nous avons vu qu'en dehors d'une structure générale similaire il n'y a pas vraiment d'approche standard pour la conception de tels systèmes. Ils peuvent aller du tout statistique sans connaissance de la langue au très linguistique avec analyse profonde basée sur de grandes bases de connaissances et incluant du raisonnement logique sur les concepts extraits. Notre expérience dans le domaine du dialogue nous a poussé vers une organisation un peu intermédiaire : une *analyse de la langue*, que l'on pourrait qualifier de *compréhension* est appliquée aux documents et aux questions et leur impose une *structure*. L'ensemble des algorithmes de recherche travaille alors uniquement sur le résultat de cette structuration. Cette approche a plusieurs avantages : le premier est une séparation conceptuelle entre les parties dépendantes de la langue, l'analyse, et celles indépendantes de la langue, la recherche. Cette séparation n'est pas parfaite, certaines parties de l'analyse sont presque indépendantes de la langue, et certaines parties de la recherche sont très dépendantes de l'analyse et donc de la langue. Cette décomposition reste cependant très utile pour contrôler la complexité de l'ensemble et donc développer efficacement le système. Un autre avantage est lié à la vitesse. Les documents sont analysés avant toute recherche, et aucune analyse supplémentaire de ces documents n'est nécessaire au moment où la demande de l'utilisateur est disponible. Cette concentration de l'analyse dans les prétraitements permet de concevoir des algorithmes très performants qui le seraient beaucoup moins si il fallait redescendre aux mots. Enfin dans notre cadre interactif cette approche nous permet d'unifier compréhension pour le dialogue et analyse pour la recherche d'informations, permettant au gestionnaire de dialogue et au système Question-Réponse de parler une langue commune que sont les entités structurées construites par l'analyse.

Notre système se divise donc en deux parties, une analyse de la langue et un système de recherche de réponses. Construire une telle analyse de la langue, qui unifie les besoins du dialogue et de Question-Réponse, est un problème ouvert. De plus il ne relève pas vraiment de notre compétence mais plutôt de celle de linguistes. Notre but est donc devenu de fournir à des linguistes le meilleur outil possible pour leur permettre de construire au mieux une telle analyse. Nous avons donc dû étudier le problème des *moteurs d'analyse*. Le caractère très expérimental du problème nous à conduit à nous intéresser en particulier aux moteurs permettant d'écrire des *systèmes à base de règles*. En effet avec un système

basé sur des règles rien n'est figé. Il est toujours possible de modifier le schéma d'annotation choisi, et il est plus facile d'agir directement sur les points semblant poser problème qu'avec un système statistique.

Les moteurs disponibles existants nous ayant semblé insuffisants, nous avons proposé le nôtre qui essaie d'équilibrer au mieux les aspects un peu conflictuels que sont l'*expressivité* et l'*ergonomie*. Une décision fondamentale a été de le structurer comme un *moteur de transformations* qui agissent sur une *représentation commune*. Cette représentation est très structurée, elle consiste en une *forêt d'alternatives*, où chaque nœud contient un ou plusieurs labels qui peuvent être des mots ou des tags. Une manière alternative utile de la voir est un vecteur de nœuds, chaque nœud pouvant optionnellement contenir un vecteur de nœuds dérivés. Une telle représentation a du bon et du mauvais. Le principal avantage d'une telle structuration est la facilité d'interprétation de son contenu. La syntaxe des règles peut en particulier s'appuyer sur cette structure pour fonctionner de manière claire et non-ambigüe. L'inconvénient principal est la limitation sur ce qui peut être représenté. En l'occurrence il n'est pas possible de représenter des relations à longue distance entre nœuds. En termes linguistiques, il n'est pas possible de représenter dans le cas général des dépendances syntaxiques ou des relations sémantiques entre chunks. Tous les groupements sont connexes. Une telle structure reste suffisante pour les parties du discours, les entités nommées, le chunking et l'analyse syntaxique en composants, entre autres, donc sa couverture est loin d'être minimale.

Les transformations peuvent être de trois types : algorithmiques, statistiques ou à base de règles. Les transformations dites algorithmiques couvrent toutes celles qui sont simples et s'écrivent bien sous la forme d'un court programme. Dans un analyseur traditionnel ce rôle est bien souvent couvert par des scripts PERL judicieusement placés. Nous avons préféré les intégrer dans le moteur de façon à ce que leur utilisation soit explicite. Les transformations statistiques sont encore très peu développées et n'incluent pour l'instant qu'une analyse en parties du discours s'appuyant sur les modèles de TreeTagger.

La transformation à base de règles est la plus aboutie. Elle s'appuie fondamentalement sur la notion d'expression régulières mais l'étend sur de nombreux axes. Le point fondamental est ce sur quoi elles s'appliquent. En effet, plutôt que de traiter des caractères, nos expressions travaillent sur des mots. Cela correspond bien évidemment mieux à notre représentation, mais cela permet en plus d'améliorer grandement la lisibilité des règles. En effet, l'espace n'étant plus qu'un séparateur, l'utilisateur est bien plus libre au niveau du formatage des règles. Cela permet aussi, et c'est fondamental, de définir une syntaxe pour des classes et macros nommées, permettant une structuration naturelle des règles. Les opérateurs étendus habituels sont présents, tels que le contrôle des répétitions (bornes inférieures et supérieures, répétitions au plus court ou au plus long) ou encore les *lookaheads*, qui permettent de tenir compte des contextes dans l'application des règles. De plus des opérateurs spécifiques à notre moteur permettent de se déplacer dans la représentation, et en particulier de descendre dans les dérivations, ce qui rend possible d'observer la totalité des annotations présentes à un moment donné et même de redescendre jusqu'aux mots d'origines. Enfin un dernier point important est la possibilité de transformer la représentation de façon assez libre via les règles, permettant de revenir sur des annotations ou de les préciser.

Globalement, un mini-langage basé sur *Lua* permet d'organiser les différentes passes de transformation utilisées dans l'analyse. Cela permet de regrouper en un fichier visible, mais qui en utilise d'autres, l'ensemble d'une analyse. Nous avons aussi donné la possibilité de construire un fichier binaire regroupant la totalité d'un système d'analyse, créant ainsi une version *packagée* très pratique pour les applications voulant ensuite utiliser cette analyse.

Nous avons présenté quelques cas d'utilisation de ce moteur et aussi quelques chiffres de performance montrant qu'il est tout à fait adapté à ce pour quoi nous l'avons conçu. Nos collègues linguistes ont en effet pu développer un système d'analyse multi-niveaux complet couvrant à la fois les besoins du dialogue et ceux de Question-Réponse. Ils ont également pu avec un effort minimal traduire ce système pour lui faire traiter l'espagnol et l'anglais.

Ayant ainsi une analyse de la langue performante, nous avons pu nous intéresser à la seconde partie du problème, le système de recherche d'informations lui-même. Nous avons deux contraintes à ce niveau-là : nous devons être capables d'accepter des entités supplémentaires venant du gestionnaire de dialogue, et nous devons pouvoir contrôler la vitesse globale du système pour assurer de bons temps de réaction du système de dialogue.

Le système Question-Réponse que nous proposons a plusieurs originalités. La première est de ne travailler que sur les structures produites par l'analyse, que nous nommons *entités*. En comparaison, beaucoup de systèmes font d'abord une extraction de texte brut des documents à partir des mots de la question, puis analysent ensuite ce texte pour rechercher la réponse. À contrario, nous préanalysons tous les documents et toutes les extractions se font ensuite en utilisant les éléments de l'analyse comme clés de recherche. Ce principe de fonctionnement a un impact primordial sur la vitesse globale du système.

Une seconde originalité est la création explicite d'un *Descripteur De Recherche* (DDR) structuré. Ce descripteur contient la totalité des informations nécessaires pour représenter la recherche à effectuer pour obtenir la réponse attendue par l'utilisateur. Cela inclut les éléments pertinents de la question et les éléments complémentaires, leurs variantes possibles, ainsi que les types attendus pour la réponse et leurs poids associés. L'existence d'un tel descripteur a deux avantages. Le premier est qu'il est suffisamment compréhensible par un humain pour avoir une idée de sa qualité. En cas d'échec de la recherche il est ainsi facile de savoir si le problème tient à l'interprétation de la question ou à l'extraction effective de la réponse. Le deuxième avantage est qu'il suffit de pouvoir en générer un pour pouvoir effectuer une recherche. L'algorithme que nous proposons gère les requêtes en langue optionnellement accompagnées d'éléments complémentaires. Mais nous pouvons très bien imaginer générer de tels descripteurs à partir d'autres sources, tels que des résultats de raisonnements logiques. De plus, il est aussi possible de reconnaître certains types de DDR pour rediriger la recherche vers d'autres méthodes ou sources d'informations comme des bases de données.

Enfin la dernière originalité est une utilisation forte de la redondance. Cela transparaît dans les approches générales qui sont des approches de *filtrage*. Même si, en pratique, nos algorithmes ne sont pas si différents des approches traditionnelles, leur but fondamental est de supprimer le texte non-

pertinent. En comparaison le but des approches standard est de trouver *la* phrase qui contient *la* réponse, et de diminuer la quantité de texte à traiter avant de la rencontrer. Nous cherchons plus à augmenter le *taux de présence* de la bonne réponse par rapport à la quantité de texte à traiter. Les paramètres de contrôle de vitesse se contentent alors de limiter la quantité de texte examiné pour assurer un temps de réponse maximal prédictible.

Les évaluations nous ont montré que les résultats du système étaient tout à fait honorables, incluant une bonne résistance aux erreurs induites par un système de transcription automatique de la parole. Nous avons cependant noté, dans le cas de la recherche dans une base constituée de pages Web, une sensibilité aux pages de *spam*, cherchant à attirer l'utilisateur par un florilège de mots-clés, mais n'offrant pas les informations recherchées. De plus, globalement, le score servant à évaluer les réponses candidates gagnerait à être plus robuste. La présence de plusieurs valeurs de tuning dont l'impact sur le résultat est difficilement modélisable oblige à une optimisation sur des données de développement, avec un grand risque de sur-apprentissage.

Il est cependant important de noter que notre but a été atteint. Le système global de Question-Réponse a les capacités nécessaires pour s'intégrer dans un système interactif. Il est utilisé dans le cadre du projet Ritel et a permis des premières expériences dont le but était d'étudier le comportement des humains face à un tel système et l'interaction homme-machine en domaine ouvert en général.

Chapitre 16

Perspectives

Les buts ont beau avoir été atteints, il est toujours possible de faire mieux. Un certain nombre de points en particulier peuvent être améliorés.

En premier lieu le moteur d'analyse pourrait tirer un meilleur parti des approches statistiques. La difficulté est double : avoir des données disponibles pour l'entraînement ou des modèles adaptés préexistants, et pouvoir intégrer l'approche en tant que transformation élémentaire sur la représentation de l'état de l'analyse. Nous avons deux pistes intéressantes à l'heure actuelle. Une première est d'utiliser des modèles probabilistes pour la désambiguïsation entre différents types. Il est par exemple difficile dans bien des cas de déterminer via des règles si une instance du nom *France* est utilisée comme lieu ou comme organisation. Des expériences préliminaires effectuées il y a quelques années ont montré qu'un gain relatif de 10% pouvait être obtenu avec un simple classifieur basé sur les exemples (*Memory Based Learning*). Le langage d'organisation des passes peut très facilement être étendu pour autoriser des branches multiples suivies de merges résolus statistiquement. Reste le problème des données d'entraînement. Nous espérons que le renouveau d'interêt actuel pour la détection et classification des entités nommées en français nous permettra d'obtenir des données d'entraînement pertinentes.

Une autre expérience que nous sommes actuellement en train de mener (thèse de G. Bernard) est du chunking et typage de chunks à partir de champs conditionnels aléatoires (CRF). Nous parlons ici de chunks syntaxiques (groupe sujet, verbal, nominal...). Les modèles sont construits sur le résultat de l'analyse actuelle et donnent des résultats prometteurs. S'ils se confirment, intégrer l'approche dans le moteur paraît pertinent.

Mais la principale évolution du moteur d'analyse serait de modifier la représentation pour permettre l'ajout de liens typés entre les nœuds. Ces liens permettraient ainsi d'établir des relations syntaxiques ou sémantiques entre éléments. Cependant cela nécessite de développer une syntaxe de règles capable de manipuler ces liens efficacement, et cela semble a priori très difficile.

Les algorithmes proposés pour la recherche de réponses peuvent eux aussi être améliorés. Les scores en particulier manquent un peu de robustesse. Influence du spam, surapprentissage des variables de tuning, ils mériteraient d'être plus élaborés. Il est toujours possible de rajouter des composants dans ces scores, tel un score de qualité informative par document qui pourrait être estimé a priori. Mais nous pensons qu'une approche probabiliste, un peu dans l'esprit de [Gillard et al. 2007], pourrait permettre d'obtenir une bien meilleure robustesse.

Ultimement, nous voulons aller plus loin que la simple analyse de documents. Un corpus de documents contient un ensemble d'informations. Notre but, fondamentalement, est d'extraire ces informations. Si nous devenons un jour capable de le faire avec une précision suffisante indépendamment de toute question, il deviendra envisageable de construire des graphes de connaissances dans lesquels des recherches directes seront possibles, bien plus rapides et potentiellement bien plus riches.

Bibliographie

S. Abney (1991). ‘Parsing by chunks’. In *Principle-Based Parsing*, pp. 257–278. Kluwer Academic Publishers.

S. Abney (1995). ‘Chunks and Dependencies : Bringing Processing Evidence to Bear on Syntax’. In *Computational Linguistics and the Foundations of Linguistic Theory*.

S. Abney (1996). ‘Partial parsing via finite-state cascades’. In *Workshop on Robust Parsing, 8th European Summer School in Logic, Language and Information*, pp. 8–15.

S. Abney (1997). ‘The SCOL Manual - Version 0.1b’. http ://www.vinartus.net/spa/.

ACE (2000). ‘Entity Detection and Tracking, Phase 1, ACE Pilot Study. Task Definition’. http ://www.nist.gov/speech/tests/ace/phase1/doc/summary-v01.htm.

G. Adda, M. Adda-Decker, J. Gauvain et L. Lamel (1997). ‘Text Normalization and Speech Recognition in French’. In *Proceedings of Eurospeech’97*, vol. 5, pp. 2711–2714, Rhodes, Greece.

G. Adda, J. Mariani, P. Paroubek, M. Rajman et J. Lecomte (1999). ‘L’action GRACE d’évaluation de l’assignation des parties du discours pour le français’. *revue Langues* **2**(2) :119–129.

M. Agatonovic, N. Aswani, K. Bontcheva, H. Cunningham, T. Heitz, Y. Li, I. Roberts et V. Tablan (2008). ‘Large-scale, Parallel Automatic Patent Annotation’. In *Proceedings of 1st International CIKM Workshop on Patent Information Retrieval - PaIR’08*, Napa Valley, California, USA.

K. Ahn et B. Webber (2007). ‘Nexus : a real time QA system’. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Amsterdam, The Netherlands.

AMI (2005). ‘The AMI meeting corpus’. http ://www.amiproject.org.

Apache (2007). ‘Apache Lucene, An overview’. http ://lucene.apache.org/java/docs/.

C. Ayache, B. Grau et A. Vilnat (2006). ‘EQueR : the French Evaluation campaign of Question-Answering Systems’. In *LREC 2006*, Genoa, Italy.

C. Baker, C. Fillmore et J. Lowe (1998). ‘The Berkeley FrameNet project’. In *Proceedings of the COLING-ACL*, pp. 86–99, Montreal, Canada.

A. Barrón-Cedeño, G. Sierra, P. Drouin et S. Ananiadou (2009). ‘An Improved Automatic Term Recognition Method for Spanish’. In *Computational Linguistics and Intelligent Text Processing*, vol. 5449/2009 of *Lecture Notes in Computer Science*, pp. 125–136.

T. C. Belle, A. Moffat, I. Witten et J. Zobel (1994). http ://www.ncsi.iisc.ernet.in/raja/netlis/wise/mg/mainmg.html.

R. E. Bellman (1957). *Dynamic Programming*. Princeton, NJ.

A. Berger, R. Caruana, D. Cohn, D. Freitag et V. Mittal (2000). 'Bridging the lexical chasm : statistical approaches to answer-finding'. In *Proceedings of the 23rd annual internation ACM SIGIR conference on research and develomment in information retrieval*, Athens, Greece.

J.-B. Berthelin, G. de Chalendar, F. Elkateb-Gara, O. Ferret, B. Grau, M. Hurault-Plantet, G. Illouz, L. Monceaux et I. Robba (2003). 'Getting reliable answers by exploiting results from several sources of information'. In *2nd CoLogNET-ElsNET Symposium : Questions and Answers : Theoretical and Applied Perspectives*, Amsterdam.

S. Bird, E. Klein et E. Loper (2009). *Natural Language Processing with Python. Analyzing Text with the Natural Language Toolkit*. O'Reilly.

P. Blunsom (2004). 'Maximum Entropy Markov Models for Semantic Role Labelling'. In *Proceedings of ALTA-2004*, Sydney.

H. Bonneau-Maynard, A. Denis, F. Béchet, L. Devillers, M. Quignard, S. Rosset et J. Villaneau (2008). *L'évaluation technologique dans le domaine du traitement automatique de la langue : l'expérience du programme Technolangue*, chap. Evaluation de la compréhension de la parole : le projet MEDIA. Hermès Éditions, Paris.

H. Bonneau-Maynard, S. Rosset, C. Ayache, A. Kuhn et D. Mostefa (2005). 'Semantic Annotation of the French Media Dialog Corpus'. In *Proceedings of Interspeech 2005*, Lisbon.

G. Bouma, J. Mur, G. van Noord, L. van der Plas et J. Tiedemann (2005). 'Question Answering for Dutch using Dependency Relations'. In *Working Notes for the CLEF 2005 Workshop*, Vienna, Austria.

D. Bourigault (2007). 'Un analyseur syntaxique opérationnel : SYNTEX'. Mémoire d'Habilitation à Diriger les Recherches.

J. Boye, J. Gustafson et M. Wirén (2006). 'Robust spoken language understanding in a computer game'. *Speech Communication* **48**(3-4) :335–353.

A. Branco, A. Mendes et R. Ribeiro (2003). 'Tagging and Shallow parsing of Portuguese : workshop notes of TASHA'2003'. Tech. Rep. TR-03-28, Department of Informatics, Faculty of Sciences, University of Lisbon.

X. Carreras et L. Màrquez (2005). 'Introduction to the CoNLL-2005 Shared Task : Semantic Role Labeling'. In *Proceedings of CoNLL-2004*.

S. F. Chen et J. Goodman (1998). 'An Empirical Study of Smoothing Techniques for Language Modeling'. Tech. Rep. TR-10-98, Computer Science Group, Harvard University, Cambridge, Massachusetts.

CHIL (2007). 'The European project CHIL'. http ://chil.server.de.

N. Chomsky (1956). 'Three models for the description of language'. *IEEE Transactions on Information Theory* **2**(3) :113–124.

O. Christ (1994a). 'The IMS Corpus Workbench Technical Manual'. Institut fur maschinelle Sprachverarbeitung.

O. Christ (1994b). ‘A Modular and Flexible Architecture for an Integrated Corpus Query System’. In *Proceedings of COMPLEX ’94 : 3rd Conference on Computational Lexicography and Text Research, Budapest)*.

O. Christ, B. M. Schulze et A. Hofmann (1999). ‘The IMS Corpus Workbench : Corpus Query Processor (CQP) User’s Manual’.

K. W. Church (1988). ‘A stochastic parts program and noun phrase parser for unrestricted text’. In *Proceedings of the second conference on Applied natural language processing*, pp. 136–143, Morristown, NJ, USA. Association for Computational Linguistics.

P. Cimiano, M. Hartung et E. Ratsch (2006). ‘Finding the Appropriate Generalization Level for Binary Relations Extracted from the Genia Corpus’. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pp. 161–169.

C. W. Cleverdon (1967). ‘The Cranfield tests on index language devices’. In *Aslib Proceedings*.

R. Cole, T. Carmell, P. Connors, M. Macon et J. Wouters (1998). ‘Intelligent animated agents for interactive language training’. In *ACM SIGCAPH Computers and the Physically Handicapped*.

P. Comas et J. Turmo (2008). ‘Robust Question Answering for Speech Transcripts : UPC Experience in QAst 2008’. In *Working Notes of CLEF 2008 Workshop*, Aarhus, Denmark.

P. Comas, J. Turmo et M. Surdeanu (2007). ‘Robust Question Answering for Speech Transcripts Using Minimal Syntactic Analysis’. In *Working Notes for the CLEF 2007 Workshop*, Budapest, Hungary.

B. Courtois (1990). ‘Un système de dictionnaires électroniques pour les mots simples du français in Dictionnaires électroniques du français.’. *Langue française* **87** :11–22.

H. Cunningham, D. Maynard, K. Bontcheva et V. Tablan (2002). ‘GATE : A framework and graphical development environment for robust NLP tools and applications’. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, pp. 168–175.

H. T. Dang, J. Lin et D. Kelly (2006). ‘Overview of the TREC 2006 Question Answering Track’. In *Text Retrieval Conference TREC-15*, pp. 99–116, Gaithersburg, MD, USA.

H. T. Dang, J. Lin et D. Kelly (2007). ‘Overview of the TREC 2007 Question Answering Track’. In *Text Retrieval Conference TREC-15*, Gaithersburg, MD, USA.

E. V. de la Clergerie, O. Hamon, D. Mostefa, C. Ayache, P. Paroubek et A. Vilnat (2008). ‘PASSAGE : from French Parser Evaluation to Large Sized Treebank’. In E. L. R. A. (ELRA) (ed.), *Proceedings of the Sixth International Language Resources and Evaluation (LREC’08)*, Marrakech, Morocco.

R. De Mori, F. Bechet, D. Hakkani-Tur, M. McTear, G. Riccardi et G. Tur (2008). ‘Spoken language understanding’. *IEEE Signal Processing Magazine* **25**(3) :50–58.

C. de Pablo-Sanchez, A. Gonzales-Ledesma, A. Moreno-Sandoval et M. T. Vicente-Diez (2006). ‘Miracle experiments in QA@CLEF 2006 in Spanish : Main Task, Real-Time QA and Exploratory QA Using Wikipedia (WiQA)’. In *Lecture Notes in Computer Science*, vol. 4730.

S. J. Derose (1989). *Stochastic methods for resolution of grammatical category ambiguity in inflected and uninflected languages*. Ph.D. thesis, Providence, RI, USA.

L. Devillers, H. Maynard, S. Rosset, P. Paroubek, K. McTait, D. Mostefa, K. Choukri, L. Charnay, C. Bousquet, N. V. . 4), F. Béchet, L. Romary, J. Antoine, J. Villaneau, M. Vergnes et J. Goulian (2004). 'The French MEDIA/EVALDA project : the evaluation of the understanding capability of Spoken Language Dialogue Systems'. In *Proceedings of the fourth International Conference on Language Resource and Evaluation*.

Diderot et d'Alembert (1751–1772). 'Encyclopédie ou Dictionnaire raisonné des sciences, des arts et des métiers'.

J. Eckle-Kohler (1998). 'Methods for quality assurance in semi-automatic lexicon acquisition from corpora'. In *Proceedings of EURALEX'98*, Liège, Belgique.

B. Favre, F. Bechet et P. Nocéra (2005). 'Robust Named Entity Extraction from Large Spoken Archives'. In *Proceedings of HLT-EMNLP'05*, pp. 491–498, Vancouver, Canada.

C. Fellbaum (1998). *WordNet – An Electronic Lexical Database*.

D. Ferrucci et A. Lally (2004). 'UIMA : An architectural approach to unstructured information processing in the corporate research environment'. *Natural Language Engineering* **10(3-4)** :327–348.

P. Forner, A. Peñas, I. Alegria, C. Forascu, N. Moreau, P. Osenova, P. Prokopidis, P. Rocha, B. Sacaleanu, R. Sutcliffe et E. T. K. Sang (2008). 'Overview of the CLEF 2008 Multilingual Question Answering Track'. In *Working Notes for the CLEF 2008 Workshop*, Aarhus, Denmark.

J. Fukumoto, T. Kato et F. Masui (2002). 'Question Answering Challenge (QAC-1) : Question answering evaluation at NTCIR Workshop 3'. In *Proceedings of the Third NTCIR Workshop on Research in Information Retrieval, Automatic Text Summarization and Question Answering*, Tokyo, Japan.

J. Fukumoto, T. Kato et F. Masui (2004). 'Question Answering Challenge for Five Ranked Answers and List Answers - Overview of NTCIR4 QAC2 Subtask 1 and 2'. In *Proceedings of the Fourth NTCIR Workshop on Research in Information Access Technologies Information Retrieval, Question Answering and Summarization*, Tokyo, Japan.

J. Fukumoto, T. Kato, F. Masui et T. Mori (2007). 'An Overview of the 4th Question Answering Challenge (QAC-4) at NTCIR Workshop 6'. In *Proceedings of the Sixth NTCIR Workshop Meeting on Evaluation of Information Access Technologies : Information Retrieval, Question Answering, and Cross-Lingual Information Access*, Tokyo, Japan.

R. Gaizauskas, P. Rodgers, H. Cunningham et K. Humphreys (1996). 'GATE User Guide'. http ://gate.ac.uk/.

O. Galibert, G. Illouz et S. Rosset (2005). 'Ritel+ : dialogue homme-machine à domaine ouvert'. In *Proceedings of TALN*, pp. 439–444, Dourdan.

S. Galliano, E. Geoffrois, G. Gravier, J. Bonastre, D. Mostefa et K. Choukri (2006). 'Corpus description of the ESTER Evaluation Campaign for the Rich Transcription of French Broadcast News'. In *Proceedings of LREC'06*, Genoa.

D. Giampiccolo, P. Forner, A. Peñas, C. Ayache, D. Cristea, V. Jijkoun, P. Osenova, P. Rocha, B. Sacaleanu et R. Sutcliffe (2007). 'Overview of the CLEF 2007 Multilingual Question Answering Track'. In *Working Notes for the CLEF 2007 Workshop*, Budapest, Hungary.

L. Gillard, P. Bellot et M. El-Bèze (2006a). ‘Question answering evaluation survey’. In *LREC 2006*, pp. 1640–1643, Genoa, Italy.

L. Gillard, P. Bellot et M. El-Bèze (2007). ‘D’une compacité positionnelle à une compacité probabiliste pour un système de Questions / Réponses’. In *Proceedings of CORIA’07*, pp. 271–286.

L. Gillard, L. Sitbon, P. Bellot et M. El-Beze (2006b). ‘Dernières évolutions de SQuaLIA, le système de Questions/Réponses du LIA’. *Traitement Automatique des Langues* **46**(3/2005).

L. Gillard, L. Sitbon, E. Blaudez, P. Bellot et M. El-Bèze (2006c). ‘The LIA at QA@CLEF-2006’. In *Working Notes for the CLEF 2006 Workshop*, Alicante, Spain.

B. Grau, G. Illouz, L. Monceaux, P. Paroubek, O. Pons, I. Robba et A. Vilnat (2005a). ‘FRASQUES, le système du groupe LIR, LIMSI’. In *Atelier EQueR de TALN 05*.

B. Grau, A.-L. Ligozat, I. Robba, M. Sialeu et A. Vilnat (2005b). ‘Term Translation Validation by Retrieving Bi-terms’. In *Working Notes for the CLEF 2005 Workshop*, Vienna, Austria.

G. Gravier, J. Bonastre, E. Geoffrois, S. Galliano, K. McTait et K. Choukri (2004). ‘ESTER, une campagne d’évaluation des systèmes d’indexation automatique d’émissions radiophoniques en français’. In *Proceedings of JEP’04*, Fèz, Maroc.

R. Grishman (1995). ‘Where’s the syntax ? The NYU MUX-6 system’. In *Proceedings of MUC-6*, San Francisco.

T. Hain, L. Burget, J. Dines, G. Garau, M. Karafiat, M. Lincoln, J. Vepa et V. Wan (2007). ‘The AMI system for the Transcription of meetings’. In *Proceedings of IEEE ICASSP’07*, Hawaii.

S. M. Harabagiu, G. A. Miller et D. I. Moldovan (1999). ‘Wordnet 2 - a morphologically and semantically enhanced resource’. In *SIGLEX Workshop On Standardizing Lexical Resources*, pp. 1–8.

S. Heiden et P. Lafon (2002). ‘Lectures assistées de l’Encyclopédie électronique : Philologic et Weblex’. *Recherches sur Diderot et sur l’Encyclopédie* **31–32**.

J. Herrera, A. Peñas et F. Verdejo (2004). ‘Question Answering Pilot Task at CLEF 2004’. In *Working Notes for the CLEF 2004 Workshop*, Bath, UK.

A. Hickl, J. Williams, J. Bensley, K. Roberts, Y. Shi et B. Rink (2006). ‘Question Answering with LCC’s CHAUCER at TREC 2006’. In *The 15th TREC Conference (TREC 2006)*.

R. Ierusalimschy, L. Henrique, F. Waldemar et C. Filho (1996). ‘Lua - an extensible extension language’. *Software : Practice and Experience* **26** :635–652.

A. Ittycheriah et S. Roukos (2002). ‘IBM’s statistical Question-Answering system - TREC-11’. In *Proceedings of the TREC 2002 Conference*.

C. Jacquemin (1996). ‘A symbolic and surgical acquisition of terms through variation’. In *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, pp. 425–438. Springer.

T. Kato, J. Fukumoto et F. Masui (2004). ‘Question Answering Challenge for Information Access Dialogue - Overview of NTCIR4 QAC2 Subtask 3’. In *Proceedings of the Fourth NTCIR Workshop on Research in Information Access Technologies Information Retrieval, Question Answering and Summarization*, Tokyo, Japan.

T. Kato, J. Fukumoto et F. Masui (2005). ‘An Overview of NTCIR-5 QAC3’. In *Proceedings of the Fifth NTCIR Workshop Meeting on Evaluation of Information Access Technologies : Information Retrieval, Question Answering and Cross-Lingual Information Access*, Tokyo, Japan.

T. Kato, J. Fukumoto, F. Masui et N. Kando (2006). ‘WoZ Simulation of Interactive Question Answering’. In *NAACL Workshop on Interactive Question Answering*, New York, USA.

M. Kaufman (1998). ‘Proceedings of the Seventh Message Understanding Conference (MUC-7)’. http ://www.itl.nist.gov/iaui/894.02/related_projects/muc/.

H. Kim, K. Kim, G. G. Lee et J. Seo (2001). ‘MAYA : A Fast Question-answering System Based on a Predictive Answer Indexer’. In *Proceedings of the ACL 2001 workshop on Open-Domain Question-Answering*, Toulouse, France.

J. Kupiec (1993). ‘MURAX : A Robust Linguistic Approach for Question Answering Using an On-Line Encyclopedia’. In *SIGIR*, pp. 181–190.

J. Kürsten, H. Kundisch et M. Eibl (2008). ‘QA Extension for Xtrieval : Contribution to the QAst track’. In *Working Notes of CLEF 2008 Workshop*, Aarhus, Denmark.

L. Lamel, G. Adda, E. Bilinski et J.-L. Gauvain (2005). ‘Transcribing Lectures and Seminars’. In *in InterSpeech’05*, Lisbon, Portugal.

L. Lamel, S. Rosset, J. Gauvain, S. Bennacef, M. Garnier-Rizet et B. Prouts (2000). ‘The LIMSI ARISE System’. *Speech Communication* **31**(4) :339–354.

D. Laurent, P. Séguéla et S. Nègre (2006). ‘Cross Lingual Question Answering using QRISTAL for CLEF 2006’. In *Working Notes for the CLEF 2006 Workshop*, Alicante, Spain.

M. Lesk (1978). ‘Lex : a lexical analysis program generator’. In *UNIX Programming Utilities and Libraries*.

A.-L. Ligozat (2006). *Exploitation et fusion de connaissances locales pour la recherche d’informations précises*. Ph.D. thesis, Université Paris-Sud 11, Orsay, France.

K. C. Litkowski (2001). ‘CL research experiments in TREC-10 question answering’. In *Text Retrieval Conference TREC-10*, pp. 122–121, Gaithersburg, MD, USA.

B. Magnini, D. Giampiccolo, P. Forner, C. Ayache, P. Osenova, A. Peñas, V. Jijkoun, B. Sacaleanu, P. Rocha et R. Sutcliffe (2006). ‘Overview of the CLEF 2006 Multilingual Question Answering Track’. In *Working Notes for the CLEF 2006 Workshop*, Alicante, Spain.

B. Magnini, S. Romagnoli, A. Vallin, J. Herrera, A. Peñas, V. Peinado, F. Verdejo et M. de Rijke (2003). ‘The Multiple Language Question Answering Track at CLEF 2003’. In *Working Notes for the CLEF 2003 Workshop*, Trondheim, Norway.

B. Magnini, A. Vallin, C. Ayache, G. Erbach, A. Peñas, M. de Rijke, P. Rocha, K. Simov et R. Sutcliffe (2004). ‘Overview of the CLEF 2004 Multilingual Question Answering Track’. In *Working Notes for the CLEF 2004 Workshop*, Bath, UK.

M. P. Marcus, M. A. Marcinkiewicz et B. Santorini (1993). ‘Building a Large Annotated Corpus of English : The Penn Treebank’. *Computational Linguistics* **19(2)**.

W. McCune (1994). ‘OTTER 3.0 Reference Manual and Guide’. Tech. Rep. ANL-94/6, Argonne National Laboratory, Argonne, IL.

W. S. Means et M. A. Bodie (2002). *The Book of SAX : The Simple Api for Xml*. USA.

T. Mitamura, E. Nyberg, H. Shima, T. Kato, T. Mori, C.-Y. Lin, R. Song, C.-J. Lin, T. Sakai, D. Ji et N. Kando (2008). 'Overview of the NTCIR-7 ACLIA Tasks : Advanced Cross-Lingual Information Access'. In *Proceedings of the Seventh NTCIR Workshop Meeting on Evaluation of Information Access Technologies : Information Retrieval, Question Answering, and Cross-Lingual Information Access*, Tokyo, Japan.

D. Moldovan, A. Harabagiu, R. Girju, P. Morarescu, F. Lacatusu, A. Novischi, A. Badulescu et O. Bolohan (2002a). 'LCC tools for Question Answering'. In *Proceedings of the 2002 Text Retrieval Conference*.

D. Moldovan, M. Pasca, S. Harabagiu et M. Surdeanu (2002b). 'Performance Issues and Error Analysis in an Open-Domain Question Answering System'. In *Proceeding of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 33–40, Philadelphia.

D. Molla, S. Cassidy et M. van Zaanen (2007). 'AnswerFinder at QAst 2007 : Named Entity Recognition for QA on Speech Transcripts'. In *Working Notes for the CLEF 2007 Workshop*, Budapest, Hungary.

D. Molla, M. van Zaanen et L. Pizzato (2006). 'AnswerFinder at TREC 2006'. In *The 15th TREC Conference (TREC 2006) proceedings*.

D. Mollá, M. van Zaanen, et D. Smith (2006). 'Named Entity Recognition for Question Answering'. In *Proceedings of ALTW 2006*, Sydney.

C. N. Mooers (1948). 'Application of Random Codes to the Gathering of Statistical Information,'. Master's thesis, MIT, Boston, MA.

D. Nadeau et S. Sekine (2007). 'A survey of named entity recognition and classification'. *Linguisticae Investigationes* **30(1)**.

G. Neumann et R. Wang (2007). 'DFKI-LT at QAST 2007 : Adapting QA Components to Mine Answers in Speech Transcripts'. In *Working Notes for the CLEF 2007 Workshop*, Budapest, Hungary.

H. Ney, U. Essen et R. Kneser (1994). 'On structuring probabilistic dependences in stochastic language modelling'. *Computer Speech and Language* **8**(1) :1–38.

P. Olgivie et J. Callan (2002). 'Experiments using the Lemur toolkit'. In *Proceedings of the 2001 Text Retrieval Conference*.

D. S. Pallett, J. G. Fiscus, W. M. Fisher, J. S. Garofolo, B. A. Lund et M. A. Przybocki (1994). '1993 benchmark tests for the ARPA spoken language program'. In *HLT '94 : Proceedings of the workshop on Human Language Technology*, pp. 49–74, Morristown, NJ, USA. Association for Computational Linguistics.

M. Pardiño, J. Gómez, H. Llorens, R. Muñoz-Terol, B. Navarro-Colorado, E. Saquete, P. Martínez-Barco, P. Moreda et M. Palomar (2008). 'Adapting IBQAS to work with Text Transcriptions in QAst Task : IBQAst'. In *Working Notes of CLEF 2008 Workshop*, Aarhus, Denmark.

P. Paroubek, I. Robba, A. Vilnat et C. Ayache (2006). 'Data, Annotations and Measures in EASY, the Evaluation Campaign for Parsers of French'. In *LREC 2006*, Genoa, Italy.

M. Pasca et S. H. Harabagiu (2001). 'The Informative Role of WordNet in Open-Domain Question Answering'. In *NAACL 2001 Workshop on WordNet and Other Lexical Resources : Applications, Extensions and Customizations*, Pittsburg, Pennsylvania.

H. J. Peat et P. Willett (1991). ‘The limitations of term co-occurrence data for query expansion in document retrieval systems’. *Journal of the American Society for Information Science* **42** :378–383.

C. Peters et M. Braschler (2001). ‘European Research Letter : cross-language system evaluation : the CLEF campaigns’. *J. Am. Soc. Inf. Sci. Technol.* **52**(12) :1067–1072.

L. Plamondon et L. Kosseim (2002). ‘QUANTUM : A Function-Based Question Answering System’. In R. Cohen & B. Spencer (eds.), *Advances in Artificial Intelligence, 15th Conference of the Canadian Society for Computational Studies of Intelligence, AI 2002*, pp. 281–292, Calgary, Canada.

L. Plamondon et L. Kosseim (2003). ‘Le web et la question-réponse : transformer une question en réponse’. In *Journées francophones de la toile (JFT 2003)*, pp. 225–234, Tours, France.

L. Plamondon, G. Lapalme et F. Pelletier (2004). ‘Anonymisation de décisions de justice’. In B. Bel & I. Martin (eds.), *XIe Conférence sur le Traitement Automatique des Langues Naturelles (TALN 2004)*, pp. 367–376, Fès, Maroc.

T. Poibeau (2005). ‘Sur le statut référentiel des entités nommées’. In *Proceedings TALN’05*, Dourdan, France.

B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff et M. Goranov (2003). ‘KIM – Semantic Annotation Platform’. In *Proceesings of the 2nd International Semantic Web Conference (ISWC2003)*, Florida, USA.

M. F. Porter (1997). *An algorithm for suffix stripping*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

C. A. Prolo (2002). ‘Generating the XTAG english grammar using metarules’. In *Proceedings of the 19th international conference on Computational linguistics*, pp. 1–7, Morristown, NJ, USA. Association for Computational Linguistics.

Quaero (2008). ‘Le programme Quaero’. http ://www.quaero.org/.

L. Quintard (2009). ‘Overview of the QUAERO 2008 monolingual question-answering track’. http ://www.lne.eu/en/r_and_d/quaero.asp.

Ritel (2007). ‘Le projet RITEL’. http ://ritel.limsi.fr.

S. Rosset (2000). *Stratégies et gestionnaire de dialogue pour des systèmes d’interrogation de bases de données à reconnaissance vocale*. Thèse de doctorat, Université Paris Sud, Orsay.

S. Rosset, O. Galibert, G. Adda et E. Bilinski (2007). ‘The LIMSI Qast systems : comparison between human and automatic rules generation f or question-answering on speech transcriptions’. In *IEEE ASRU*.

S. Rosset, O. Galibert, G. Adda et E. Bilinski (2008). ‘The LIMSI participation to the QAst track’. In *Lecture Notes in Computer Science*, vol. 5152, pp. 414–423.

S. Rosset, O. Galibert, G. Illouz et A. Max (2006). ‘Interaction et recherche d’informations : le projet RITEL’. *Traitement Automatique des Langues* **46**(3/2005).

S. Rosset et S. Petel (2006). ‘The Ritel Corpus - An annotated Human-Machine open-domain question answering spoken dialog corpus’. In *LREC 2006*, Genoa, Italy.

Y. Sasaki, H.-H. Chen, K. hua Chen et C.-J. Lin (2005). 'Overview of the NTCIR-5 Cross-Lingual Question Answering Task (CLQA1)'. In *Proceedings of the Fifth NTCIR Workshop Meeting on Evaluation of Information Access Technologies : Information Retrieval, Question Answering and Cross-Lingual Information Access*, Tokyo, Japan.

Y. Sasaki, C.-J. Lin, K. hua Chen et H.-H. Chen (2007). 'Overview of the NTCIR-6 Cross-Lingual Question Answering (CLQA) Task'. In *Proceedings of the Sixth NTCIR Workshop Meeting on Evaluation of Information Access Technologies : Information Retrieval, Question Answering, and Cross-Lingual Information Access*, Tokyo, Japan.

F. Schilder, A. McCulloh, B. T. McInnes et A. Zhou (2005). 'TLR at DUC : Tree Similarity'. In *Document Understanding Conference (DUC)*, University of Minnesota.

H. Schmid (1994). 'Probabilistic Part-of-Speech Tagging Using Decision Trees'. In *International Conference on New Methods in Language Processing*, pp. 44–49, Manchester, UK.

H. Schmid (1995). 'Improvements in Part-of-Speech Tagging with an Application to German'. In *Proceedings of the ACL SIGDAT-Workshop*, pp. 47–50.

K. K. Schuler (2005). *Verbnet : a broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, Philadelphia, PA, USA. Supervisor-Martha S. Palmer.

H. Schütze (1995). 'Distributional part-of-speech tagging'. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics*, pp. 141–148, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Y. Seginer (2007). 'Fast Unsupervised Incremental Parsing'. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 384–391, Prague, Czech Republic. Association for Computational Linguistics.

S. Sekine (2004). 'Definition, dictionaries and tagger of Extended Named Entity hierarchy'. In *LREC'04*, Lisbon, Portugal.

F. Smadja (1993). 'Retrieving Collocations from Text : Xtract'. *Computational linguistics* **19(1)** :143–177.

M. Surdeanu, J. Turmo et E. Comelles (2005). 'Named Entity Recognition from spontaneous Open-Domain Speech'. In *in InterSpeech'05*, Lisbon, Portugal.

R. S. Swier et S. Stevenson (2004). 'Unsupervised Semantic Role Labelling'. In D. Lin & D. Wu (eds.), *Proceedings of EMNLP 2004*, pp. 95–102, Barcelona, Spain. Association for Computational Linguistics.

TC-Star (2004-2008). http ://www.tc-star.org.

D. Toney, S. Rosset, A. Max, O. Galibert et E. Bilinski (2008). 'An Evaluation of Spoken and Textual Interaction in the RITEL Interactive Question Answering System'. In ELRA (ed.), *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.

J. Turmo, P. Comas, C. Ayache, D. Mostefa, S. Rosset et L. Lamel (2007). 'Overview of the QAST 2007'. In *Working Notes for the CLEF 2007 Workshop*, Budapest, Hungary.

J. Turmo, P. Comas, S. Rosset, L. Lamel, N. Moreau et D. Mostefa (2008). 'Overview of QAST 2008'. In *Working Notes for the CLEF 2008 Workshop*, Aarhus, Denmark.

A. Vallin, D. Giampiccolo, L. Aunimo, C. Ayache, P. Osenova, A. Peñas, M. de Rijke, B. Sacaleanu, D. Santos et R. Sutcliffe (2005). 'Overview of the CLEF 2005 Multilingual Question Answering Track'. In *Working Notes for the CLEF 2005 Workshop*, Vienna, Austria.

E. M. Voorhees (2000). 'Overview of the TREC-9 Question Answering Track'. In *Text Retrieval Conference TREC-9*, pp. 71–80, Gaithersburg, MD, USA.

E. M. Voorhees (2002). 'Overview of the TREC 2002 Question Answering Track'. In *Text Retrieval Conference TREC-11*, Gaithersburg, MD, USA.

E. M. Voorhees (2003). 'Overview of the TREC 2003 Question Answering Track'. In *Text Retrieval Conference TREC-12*, pp. 54–68, Gaithersburg, MD, USA.

E. M. Voorhees (2004). 'Overview of the TREC 2004 Question Answering Track'. In *Text Retrieval Conference TREC-13*, Gaithersburg, MD, USA.

E. M. Voorhees et H. T. Dang (2005). 'Overview of the TREC 2005 Question Answering Track'. In *Text Retrieval Conference TREC-14*, Gaithersburg, MD, USA.

E. M. Voorhees et D. K. Harman (2005). *TREC : Experiment and Evaluation in Information Retrieval*. Digital Libraries and Electronic Publishing.

E. M. Voorhees et D. M. Tice (1999). 'The TREC-8 Question Answering Track Report'. In *Text Retrieval Conference TREC-8*, pp. 77–82, Gaithersburg, MD, USA.

E. M. Voorhees et D. M. Tice (2001). 'Overview of the TREC 2001 Question Answering Track'. In *Text Retrieval Conference TREC-10*, pp. 42–51, Gaithersburg, MD, USA.

P. Vossen (1998). 'EuroWordNet A Multilingual Database with Lexical Semantic Networks'.

M. Walker, A. Rudnicky, J. Aberdeen, E. Bratt, J. Garofolo, H. Hastie, A. Le, B. Pellom, A. Potamianos, R. Passonneau, R. Prasad, S. Roukos, G. Sanders, S. Seneff et D. Stallard (2002). 'DARPA Communicator Evaluation : Progress from 2000 to 2001'. In *ICSLP'02*, Denver, EU.

J. Weizenbaum (1966). 'ELIZA–A Computer Program For the Study of Natural Language Communication Between Man and Machine'. *Communications of the ACM* **9(1)**.

E. Whittaker, P. Chatain, S. Furui et D. Klakow (2005a). 'TREC2005 Question Answering Experiments at Tokyo Institute of Technology'. In *Proceedings of the 14th Text Retrieval Conference*.

E. Whittaker, S. Furui et D. Klakow (2005b). 'A Statistical Classification Approach to Question Answering using Web Data'. In *CW '05 : Proceedings of the 2005 International Conference on Cyberworlds*, pp. 421–428, Washington, DC, USA. IEEE Computer Society.

E. Whittaker, J. Novak, M. Heie et S. Furui (2007). 'CLEF2007 Question Answering Experiments at Tokyo Institute of Technology'. In *Working Notes for the CLEF 2007 Workshop*, Budapest, Hungary.

E. Whittaker, J. Novakand, P. Chatain et S. Furui (2006). 'TREC2006 Question Answering Experiments at Tokyo Institute of Technology'. In *Proceedings of the 15th Text Retrieval Conference*.

W. A. Woods (1973). 'Progress in natural language understanding : an application to lunar geology'. In *AFIPS '73 : Proceedings of the June 4-8, 1973, national computer conference and exposition*, pp. 441–450, New York, NY, USA. ACM.

Xapian (2001). 'The Xapian project'. http ://www.xapian.org.