Explorations in Word Embeddings : graph-based word embedding learning and cross-lingual contextual word embedding learning

Zheng Zhang

UNIVERSITÉ PARIS SUD
Comprendre le monde, construire l'avenir

Explorations in word embeddings: graph-based word embedding learning and cross-lingual contextual word embedding learning

Thèse de doctorat de l'Université Paris-Saclay
préparée à Université Paris-Sud

École doctorale n°580
Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Orsay, le 18/10/2019, par

Zheng Zhang

Composition du Jury :

François Yvon Senior Researcher, LIMSI-CNRS	Président
Mathieu Lafourcade Associate Professor, Université de Montpellier	Rapporteur
Emmanuel Morin Professor, Université de Nantes	Rapporteur
Armand Joulin Research Scientist, Facebook Artificial Intelligence Research	Examinateur
Pierre Zweigenbaum Senior Researcher, LIMSI-CNRS	Directeur de thèse
Yue Ma Associate Professor, Université Paris-Sud	Co-encadrant

Acknowledgements

Word embedding learning is a fast-moving domain.
So is my three years' time as a PhD student at Limsi.

I would like to thank the committee. Thank you for evaluating my work. I felt so happy working with so many awesome people at LIMSI. Thank you, Aurelie, Anne-Laure, Sahar, Anne, Patrick, Michael, Amel and all permanents for guiding me and discussing with me.

Most particularly, I would like to thank my supervisors Pierre and Yue. Thank you for everything you have thought me. And thank you so much for your patience. I know guiding a student for 3 years is not easy, especially when that student speaks English with some French "pronouciation" in Chinese grammars.

Word embeddings are not limited in French, Chinese, Indian, Vitenimien, etc.
So are my friends.

I would like to thank Sanjay, Swen, Hicham, Léon-Paul, Christopher, Arnaud, Rashedur, Julien, Yuming, Charlotte, Aina, Ruqing, Duc and all students at Limsi. Thank you for creating such a great environment not only for research but also for afterwork.

I would like to thank my friends and also my classmates from my bachelor and master studies. Heng, Jianan, Enze, Zhenyu, Botao, Ye, Qi, Xiaomeng, Xin, Guillaume, Amin, Gishan and Daro, thank you so much for being great friends with me and having such a "long long term memory" together.

Word embeddings are important for nearly all modern NLP architectures.
So… well, my parents don't care.

They just support me unconditionally to chase my dreams. I would like to thank their support and love.

"A word is characterized by the company it keeps." So am I.

I would like to thank my company, my dear wife Gao Min. I can not achieve all of this without the peace and love you gave to me. And because of you, I want to be better and better. I'm your company, and you are always my only center.

Abstract

Word embeddings are a standard component of modern natural language processing architectures. Every time there is a breakthrough in word embedding learning, the vast majority of natural language processing tasks, such as POS-tagging, named entity recognition (NER), question answering, natural language inference, can benefit from it. This work addresses the question of how to improve the quality of monolingual word embeddings learned by prediction-based models and how to map contextual word embeddings generated by pre-trained language representation models like ELMo or BERT across different languages.

For monolingual word embedding learning, I take into account global, corpus-level information and generate a different noise distribution for negative sampling in word2vec. In this purpose I pre-compute word co-occurrence statistics with corpus2graph, an open-source NLP-application-oriented Python package that I developed: it efficiently generates a word co-occurrence network from a large corpus, and applies to it network algorithms such as random walks. For cross-lingual contextual word embedding mapping, I link contextual word embeddings to word sense embeddings. The improved anchor generation algorithm that I propose also expands the scope of word embedding mapping algorithms from context-independent to contextual word embeddings.

Résumé

Les plongements lexicaux sont un composant standard des architectures modernes de traitement automatique des langues (TAL). Chaque avancée obtenue dans l'apprentissage de plongements lexicaux peut bénéficier à la grande majorité des tâches de traitement automatique des langues, telles que l'étiquetage morphosyntaxique, la reconnaissance d'entités nommées, la recherche de réponses à des questions, ou l'inférence textuelle.

Ce travail étudie les méthodes d'apprentissage de plongements lexicaux monolingues et multilingues. Évoluant avec les progrès rapides de l'apprentissage de plongements lexicaux, mes recherches vont d'un modèle d'apprentissage monolingue indépendant du contexte à des modèles d'apprentissage multilingues récents et contextuels. Ce travail explore ainsi la question de l'amélioration de la qualité de plongements lexicaux monolingues appris par des modèles prédictifs et celle de la mise en correspondance entre langues de plongements lexicaux contextuels créés par des modèles préentraînés de représentation de la langue comme ELMo ou BERT.

Pour l'apprentissage de plongements lexicaux monolingues, je prends en compte des informations globales au corpus et génère une distribution de bruit différente pour l'échantillonnage d'exemples négatifs dans word2vec.

Dans le chapitre 3 du mémoire, je propose corpus2graph, un paquet Python en source ouverte orienté vers les applications en TAL : il génère efficacement un graphe de cooccurrence à partir d'un grand corpus, et lui applique des algorithmes de graphes tels que les marches aléatoires. Il contient non seulement différentes méthodes intégrées pour le prétraitement des mots, l'analyse des phrases, l'extraction des paires de mots et la définition de la pondération des arêtes, mais prend également en charge des fonctions personnalisées. En utilisant des techniques de parallélisation, il peut générer en quelques heures un vaste réseau de cooccurrences contenant l'ensemble des données de Wikipedia anglais. Grâce à sa conception de calcul progressif à trois niveaux poids-arêtes-poids, la reconstruction de réseaux avec des configurations différentes est encore plus rapide car elle n'a pas besoin de recommencer à zéro. Cet outil fonctionne également avec d'autres bibliothèques de graphes telles que igraph, NetworkX et graph-tool en tant que frontal fournissant des données et permettant d'accroître la vitesse de génération du réseau.

Au chapitre 4, j'émets l'hypothèse que la prise en compte d'informations globales au niveau du corpus et la génération d'une distribution de bruit différente dans un échantillonnage négatif pour chaque mot cible dans l'algorithme word2vec skip-gram satisfont mieux aux exigences relatives aux exemples négatifs pour chaque mot d'apprentissage que la distribution d'origine basée sur la fréquence. Je propose une nouvelle méthode basée sur les graphes pour calculer une distribution de bruit pour un échantillonnage négatif. En utilisant un réseau de cooccurrence de mots pré-calculé, ma distribution de bruit peut être ciblée sur les mots d'apprentissage. Je teste cette hypothèse à travers un ensemble d'expériences dont les résultats montrent que mon approche augmente d'environ 5% les performances sur la tâche d'analogie de mots et d'environ 1% les performances des tâches de similarité de mots par rapport à la méthode d'échantillonnage négatif d'origine de l'algorithme skip-gram.

Pour la mise en correspondance translingue de plongements lexicaux, je relie les plongements lexicaux contextuels à des plongements de sens de mots. L'algorithme amélioré de création d'ancres que je propose étend également la portée des algorithmes de mise en correspondance de plongements lexicaux du cas non-contextuel au cas des plongements contextuels.

Au chapitre 5, j'explore ainsi l'incorporation de plongements lexicaux contextuels (ou encore plongements d'occurrences de mots) pour des mots à sens multiples. Je soutiens que la méthode de pointe actuelle pour l'apprentissage de plongements lexicaux contextuels multilingues ne peut pas gérer correctement les mots à sens multiples. Je propose des solutions qui considèrent les mots à sens multiples comme du bruit lors de l'alignement des espaces des deux langues traitées. Les expériences réalisées montrent que mes méthodes peuvent améliorer l'alignement des plongements lexicaux pour les mots à sens multiples dans une perspective microscopique sans nuire aux performances macroscopiques de la tâche d'induction de lexique.

Table of contents

List of figures

List of tables

Introduction

Context

Word embeddings, vector representations of words, are a standard component of modern natural language processing architectures. Every time there is a breakthrough in word embeddings learning, the vast majority of Natural Language Processing (NLP) tasks, such as question answering (Liu et al., 2017), textual entailment (Chen et al., 2016), semantic role labeling (He et al., 2017), coreference resolution (Pradhan et al., 2012), named entity extraction (Sang and De Meulder, 2003), sentiment analysis (Socher et al., 2013), may benefit from it.

With the rapid development of deep learning, research into monolingual word embeddings took off with the word2vec model in 2013 (Mikolov et al., 2013d) and it directly boosted up applications of deep learning in natural language processing.

At the same time, cross-lingual word embeddings, vector representations of words in multiple languages, attract researchers attention as a key ingredient for multilingual applications in NLP. A cross-lingual word embedding space, where words in different languages are comparable according to their syntactic and semantic similarities, enables the comparison of the meaning of words across languages and *cross-lingual model transfer between languages* (Anders et al., 2019).

Research questions

In this fast-moving domain, there is already much meaningful work, but also still many open research questions in monolingual and multilingual word embedding learning, including about training models (Gouws et al., 2015; Mikolov et al., 2013a), evaluation metrics (Rogers et al., 2018), and the theoretical analysis of hyperparameters (Yin and Shen, 2018) or relations between different models (Artetxe et al., 2016; Levy and Goldberg, 2014b).

Within this large range of options, I decided to focus on the following two questions:

1. **How to combine the strengths of both prediction-based and count-based methods to improve the quality of monolingual word embeddings.** Before the advent of pre-trained language representation models, the state of the art in monolingual word embeddings learning was represented by the skip-gram negative sampling architecture (Mikolov et al., 2013d) and GloVe (Pennington et al., 2014). Both prediction-based and count-based methods achieved high quality word embeddings based on their own advantages, but there was little research about combining them.

2. **How to apply cross-lingual context-independent word embedding learning algorithms to contextual word embeddings.** With the most recent progress of monolingual word embedding learning by using pre-trained language representation models such as ELMo (Peters et al., 2018a) and BERT (Devlin et al., 2019), monolingual word embeddings have moved from context-independent representations to contextual representations. However, nearly all cross-lingual word embedding learning research is still based on using context-independent word embeddings. How to transfer algorithms of this research to contextual word embeddings remains unclear.

Contributions

1. (a) For monolingual word embedding learning, I take into account global, count-based corpus-level information and generate a different noise distribution for negative sampling in word2vec. Experiments show that this improves word analogy performance by about 5% and word similarity tasks by about 1%.

   (b) In this purpose I pre-compute word co-occurrence statistics with corpus2graph, an open-source NLP-application-oriented Python package that I developed: it efficiently generates a word co-occurrence network from a large corpus, and applies to it network algorithms such as random walks.

2. For cross-lingual contextual word embedding mapping, I link contextual word embeddings to word sense embeddings. The improved anchor generation algorithm that I propose also expands the scope of word embedding mapping algorithms from context-independent to contextual word embeddings.

Outline of the manuscript

The remainder of this manuscript is organized as follows.

In Chapter 1, I review monolingual word embedding learning models in general. I detail the architecture of models in count-based methods and prediction-based methods and how they are usually trained. Then I outline the hyperparameters setting and evaluations for monolingual word embedding learning.

Chapter 2 describes various models to learn cross-lingual word embeddings. I categorize the models into three groups: corpus preparation, training and post-training, and introduce the respective models. I then introduce evaluations used for cross-lingual word embedding learning.

In Chapter 3, I begin by introducing the concept of word co-occurrence network, as well as its applications in natural language processing. In order to generate a word cooccurrence network efficiently from Wikipedia dumps, I propose my solution, corpus2graph, an open-source NLP-application-oriented Python package that generates a word co-occurrence network from a large corpus.

In Chapter 4, I aim at improving the quality of monolingual word embeddings trained with the skip-gram negative sampling model, by injecting word co-occurrence network information into negative examples selection. I experiment this hypothesis through a set of experiments whose results show that my approach boosts the word analogy task by about 5% and improves the performance on word similarity tasks by about 1% compared to the skip-gram negative sampling baseline.

Finally, in Chapter 5, I explore the characteristics of contextual word embeddings and show the link between contextual word embeddings and word senses. I also propose an improved anchor-based cross-lingual contextual word embeddings mapping algorithm that improves cross-lingual word embeddings mapping especially for multi-sense words.

I then conclude the manuscript.

Chapter 1

Monolingual Word Embedding and State-of-the-art Approaches

1.1 A brief history about the terminology "word embedding"

The roots of word embeddings can be traced back to the 1950s when the distributional hypothesis, of which the underlying idea is that "a word is characterized by the company it keeps" (Firth, 1957), was discussed in linguistics. The concept and the training models about "word embedding" are evolving with the progress in natural language processing. In fact, even the terminology of "word embedding" itself went a long way to what it is today (as shown in Figure 1.1). Below we give a brief history about the evolution of the terminologies used for "word embedding".

As introduced before, word embedding starts with the distributional hypothesis in linguistics. This hypothesis suggests that *words that are used and occur in the same contexts tend to purport similar meanings* (Harris, 1954). The semantic similarities between words become measurable by comparing their companies (context words).

In natural language processing, a similar idea called Vector Space Model (VSM) has been firstly introduced in information retrieval (Rocchio, 1971; Salton, 1962) to determine document similarities. Based on the term-document matrix, each document is represented as a vector in the VSM, where *Points that are close together in this space are semantically similar and points that are far apart are semantically distant* (Turney and Pantel, 2010).

The notion of a distributed semantic representation, another important element of word embedding, comes after the vector representation basis. It aims to reduce the number of

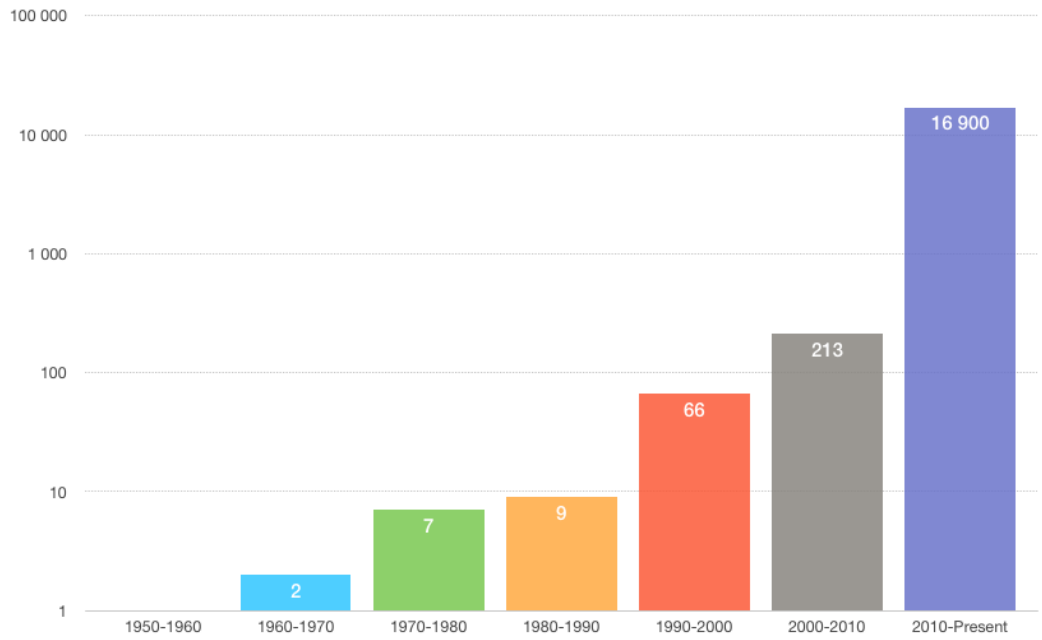


Fig. 1.1 Number of results by searching "word embedding" in Google Scholar

dimensions by using techniques like singular value decomposition (SVD) and latent semantic analysis (LSA) in VSM as the previous term-document matrix used for information retrieval is too sparse to measure the distributional similarity and the matrix is getting larger and larger with the increase of the data.

After the success of LSA in information retrieval (Deerwester et al., 1990), Schütze (1993) introduced Word Space, distributed semantic representations for words derived from lexical co-occurrence statistics. Using distributed semantic representations for words led to many improvements in different NLP tasks in the 2000s such as word sense discovery (Rapp, 2003) and the similarity of semantic relations (Turney, 2006).

In 2003, Bengio et al. (2003a) proposed another way to learn distributed representations for words, where word vectors are generated based on the *linguistic contexts in which the words occur* (prediction-based) instead of global word co-occurrence statistics (count-based).

In the present decade, with the rapid development in deep learning, the research of word embedding took off (as shown in Figure 1.1, around 16900 results by searching "word embedding" in Google Scholar) with the word2vec model (Mikolov et al., 2013d). This directly boosted up applications of deep learning in natural language processing. From the widely used word2vec (Mikolov et al., 2013a) in 2013 to the most recent methods like ELMo (Embeddings from Language Models) (Peters et al., 2018b) and BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018) in 2018, every time there is a breakthrough in word embeddings learning, the vast majority of NLP tasks, such

Table 1.1 Terminology related to "word embedding"

Terminology	Paper	Year
Distributional Hypothesis	Harris (1954)	1954
Vector Space Model (VSM)	Rocchio (1971)	1971
Distributed semantic representation for documents	Deerwester et al. (1990)	1990
Word Space, Distributed semantic representation for words	Schütze (1993)	1993
Distributed representation for words	Bengio et al. (2003b)	2003
Continuous space word representations	Mikolov et al. (2013f)	2013
Continuous vector representations of words	Mikolov et al. (2013a)	2013
Pre-trained word representations / vectors	Peters et al. (2018b)	2018
Deep contextualized word representations	Peters et al. (2018b)	2018
Pre-trained general language representations	Devlin et al. (2019)	2019

as POS-tagging, chunking, named entity recognition (NER), multi-way classification of semantic relations, sentence-level sentiment polarity classification, document-level polarity classification, classification of subjectivity and objectivity and natural language inference task, may benefit from it.

In the present chapter, I focus on word embedding research starting from 2013 (word2vec) along with two classical methods, the neural network based language model (Bengio et al., 2003a; Devlin et al., 2019; Joulin et al., 2017; Mikolov et al., 2013a,d; Peters et al., 2018b) in prediction-based methods and the Pointwise Mutual information (PMI) matrix based model (Levy and Goldberg, 2014b; Pennington et al., 2014; Salle et al., 2016; Xin et al., 2018) in count-based methods.

This chapter is organized as follows. Section 1.2 discusses prediction-based word embedding learning methods. Section 1.3 introduces state-of-the-art approaches of count-based methods for word embedding learning. Section 1.4 talks about the influence of some common parameters used in prediction-based and count-based methods. Section 1.5 introduces evaluations for monolingual word embeddings.

1.2 Prediction-based methods

1.2.1 A neural probabilistic language model

As introduced in Section 1.1, Bengio et al. (2003a) introduced a neural probabilistic language model to solve the *curse of dimensionality*, i.e. the fact that the joint distribution of a large number of discrete variables leads to exponentially large free parameters for probabilistic language modeling. As shown in Figure 1.2, the feedforward neural network contains a linear

projection layer (the bottom layer in the figure) and a non-linear hidden layer (the middle layer in the figure), which learns jointly the word feature vector and a statistical language model. Its training approach is summarized below:

1. *Associate with each word in the vocabulary a distributed word feature vector.*
2. *Express the joint probability function of word sequences in terms of the feature vectors of these words in the sequence.*
3. *Learn simultaneously the word feature vectors and the parameters of that probability function.*

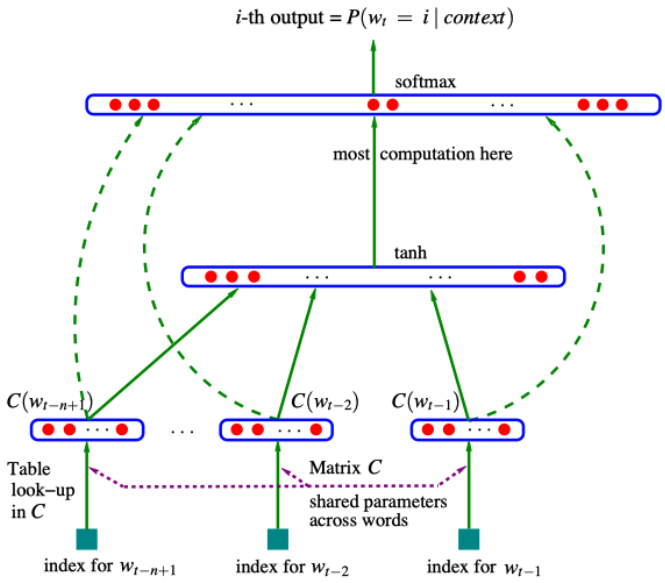


Fig. 1.2 Neural Probabilistic Language Model Architecture. Figure from Bengio et al. (2003a)

1.2.2 word2vec

The word2vec paper Mikolov et al. (2013a) is one of the most famous papers in NLP in the last ten years. It is one of the most successful self-supervised learning models in NLP. This paper has two major contributions.

1.2.2.1 Two new model architectures

To minimize computational complexity in neural network based language models such as feedforward neural net language model (NNLM) and Recurrent neural net language model

(RNNLM), Mikolov et al. (2013a) proposed the Continuous Bag-of-Words Model (CBOW) and the Skip-gram Model (Skip-gram) which are simpler and efficient, and obtain state-of-the-art performance on the word analogy task which I will introduce in the next section (Section 1.2.2.2).

CBOW is similar to the feedforward NNLM. It removes the non-linear hidden layer (the middle layer in Figure 1.2), and the projection layer is shared for all words. As shown in the CBOW model architecture (Figure 1.3), given context words as the input, a log-linear classifier is trained to correctly classify the target middle word. Bag-of-words means that the context words are trained equally regardless of their position.

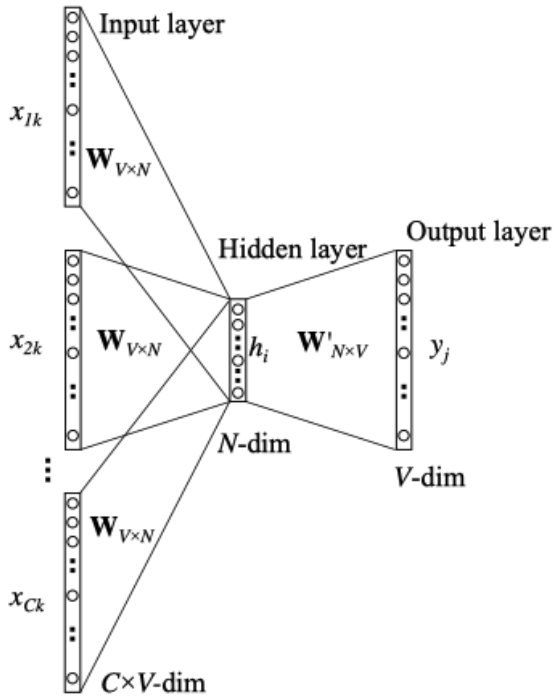


Fig. 1.3 Continuous Bag-of-Words Model. Figure from Rong (2014)

The skip-gram model is similar to CBOW, *but instead of predicting the current word based on the context, it tries to maximize classification of a word based on another word in the same sentence.* In short, given the current middle word as the input, a log-linear classifier is trained to correctly predict context words.

1.2.2.2 A new test set

A good word embedding puts similar words close to each other in the embedding space. As there can be many different types of similarities (syntactic and semantic), an evaluation is

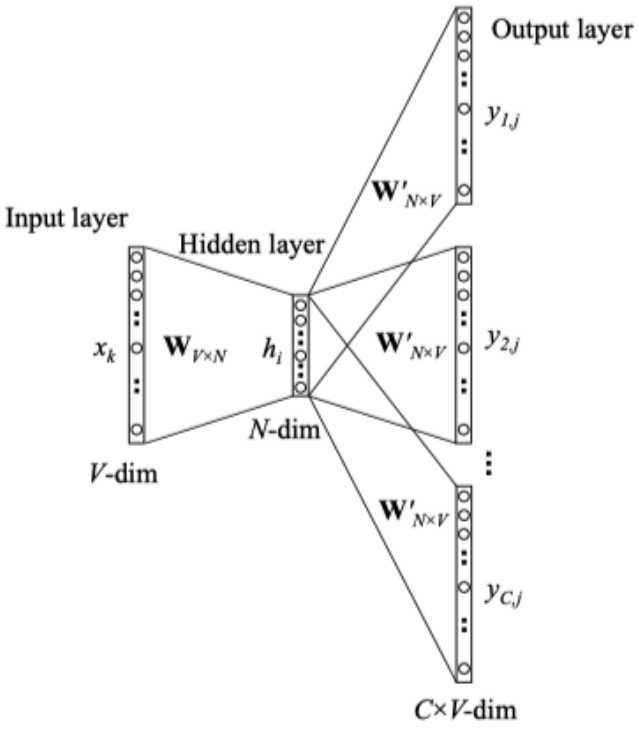


Fig. 1.4 The Skip-gram Model. Figure from Rong (2014)

always necessary to measure the quality of embedding space from a certain point of view. This can be performed as an intrinsic task (e.g. WordSim-353 (Agirre et al., 2009)) where the word embedding is directly used, or as an extrinsic task in which the pre-trained word embedding is used as an input.

Mikolov et al. (2013a) introduced a new data set for measuring these similarities by a word analogy task (a kind of intrinsic task). Detailed explanation and analysis will be presented in Section 1.5.1.2 and Section 1.5.1.3.

While Mikolov et al. (2013a) introduce the major structure of the word2vec model, what made word2vec what we use in all kinds of tools or libraries today is the paper (Mikolov et al., 2013d). (This may also be the reason why (Mikolov et al., 2013d) is more cited than (Mikolov et al., 2013a).) We describe below several important extensions of the Skip-gram model that have been introduced to speedup training and to improve the embedding quality.

1.2.2.3 Negative sampling

The objective of the Skip-gram model is defined as:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p\left(w_{t+j}|w_t\right) \tag{1.1}$$

where T is the number of the training words, c is the context window size, w_t is the center/training word, and $p\left(w_{t+j}|w_t\right)$ is defined by using the softmax function:

$$p\left(w_O|w_I\right) = \frac{\exp\left(v'^{\top}_{w_O} v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left(v'^{\top}_{w} v_{w_I}\right)} \tag{1.2}$$

where v_w and v'_w are the input and output vector representations of w and W is the number of words in the vocabulary.

This function is extremely computationally expensive as it is proportional to W. So negative sampling, which approximately maximizes the log probability of the softmax, has been proposed to replace $logp\left(w_O|w_I\right)$:

$$\log\sigma\left(v'^{\top}_{w_O} v_{w_I}\right) + \sum_{i=1}^{k}\mathbb{E}_{w_i\sim P_n(w)}\left[\log\sigma\left(-v'^{\top}_{w_i} v_{w_I}\right)\right] \tag{1.3}$$

Thus the task is to distinguish the target word w_O from draws from the noise distribution $P_n(w)$ using logistic regressing, where there are k negative samples for each data sample.

The noise distribution is defined as:

$$P_n(w) = U(w)^{3/4}/Z \tag{1.4}$$

where $U(w)$ is the uniform distributions.

Note that the definition of the noise distribution is empirical. Other different definitions will be discussed in Chapter 4.

1.2.2.4 Subsampling of frequent words

During the training time, word2vec model traverses the entire corpus several times. It is clear that the training times of a distinct word is proportional to its frequency in the corpus. To solve this imbalance between frequent and rare words, Mikolov et al. (2013d) suggested using the subsampling approach below:

$$P\left(w_i\right) = 1 - \sqrt{\frac{t}{f\left(w_i\right)}} \tag{1.5}$$

where the frequency of word w_i is $f(w_i)$ and t is a chosen threshold (typically 10^{-5}).

1.2.2.5 Finding phrases in text

During the corpus preprocessing stage, it is better to represent a common phrase as one token instead of several tokens for all the words inside. To achieve this goal, Mikolov et al. (2013d) used the approach below to calculate the score for bigrams:

$$\text{score}\left(w_i, w_j\right) = \frac{\text{count}\left(w_i w_j\right) - \delta}{\text{count}\left(w_i\right) \times \text{count}\left(w_j\right)} \tag{1.6}$$

a variant of the pointwise mutual information (see Formula 1.10, scaled by the size of the corpus), where δ is the discounting coefficient. Typically, this approach will be ran 2-4 times over the entire corpus. Each time the bigrams with scores higher than the chosen threshold will be represented as phrases.

1.2.3 fastText

FastText has been firstly introduced as a text classifier Joulin et al. (2017). As shown in Figure 1.5, fastText classifier's architecture is similar to the CBOW model: Predict the sentence label (instead of the center word) by the sequence of n-grams from that sentence (instead of the bag of context words).

Why does fastText use bag of n-grams instead of bag of words? For the sentence classification task, word order can bring additional information for sentence representations. While bag of words has no word order information, n-grams can bring *some partial information about the local word order*. For instance, a n-gram may be a combination of the ending character sequence of a preceding word and the beginning part of the current word.

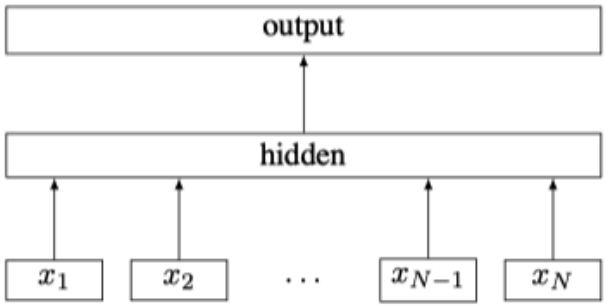


Fig. 1.5 FastText model architecture for a sentence with N ngram features $x_1, ..., x_N$. Figure from Joulin et al. (2017)

Then this bag of n-grams idea has been applied to word embedding learning Bojanowski et al. (2017). Based on the Skip-gram model, the fastText model uses a bag of **character**

n-grams. Models like word2vec ignore the morphology of words, whereas character n-grams can explicitly embed morphology information. It not only improves the word embedding quality but also provides a way for out-of-vocabulary (OOV) word representation learning.

For each target word in the Skip-gram model, instead of calculating the scalar product between its vector and context word vector $v'^{\top}_{w_O} v_{w_I}$ (also the scalar product between its vector and negative example vectors $v'^{\top}_{w_i} v_{w_I}$) in the objective function (Equation 1.3), the target word vector is replaced by *the sum of the vector representations of its n-grams*.

Therefore the scalar product between target word v_{w_I} and context word v'_{w_O} is defined as:

$$\sum_{g \in \mathcal{G}_{w_I}} \mathbf{z}_g^{\top} \mathbf{v}'_{w_O} \tag{1.7}$$

where $\mathcal{G}_{w_I}$ is the set of n-grams appearing in target words w_I and z_g is the vector of an n-gram. Note that the word vector w_I is also included in the set of n-grams.

After training, as fastText also preserves n-gram embeddings, an OOV word embedding can be simply defined as the average of its n-grams embeddings.

1.2.4 Contextual word embedding learning

All word embeddings discussed before are context independent, i.e. each distinct word can have only one vector representation. This has two drawbacks:

- A word always has the same representation, regardless of the context in which its individual tokens occur. Especially for polysemy, one vector representation is not enough for its different senses.
- Even for words that have only one sense, their occurrences still have different aspects including semantics, syntactic behavior and language register/connotations. Different NLP tasks may need different aspects from words.

To solve these problems, the idea of contextual word embeddings came up. Contextual word embeddings are dynamic, based on the context each token is used in, rather than a static context-independent embedding. Below I introduce two representative methods, which also led to breakthrough in many NLP tasks.

1.2.4.1 ELMo: Embeddings from Language Models

ELMo embeddings are *derived from a bidirectional LSTM that is trained with a coupled language model (LM) objective on a large text corpus*.

As shown in Figure 1.7, ELMo uses a two-layer bidirectional language model (biLM), predicting the next word in a sequence of previous words and predicting the previous word in a sequence of following words, for pretraining.

For each token t_k, the word embedding given by ELMo is:

$$\begin{aligned} R_k &= \left\{ \mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} | j = 1, \ldots, L \right\} \\ &= \left\{ \mathbf{h}_{k,j}^{LM} | j = 0, \ldots, L \right\} \end{aligned} \tag{1.8}$$

where L represents the number of layers in biLM ($L = 2$), x_k^{LM} is the context-independent token representation (bottom orange blocks in Figure 1.7), $h_{k,j}^{LM} = [\overrightarrow{\mathbf{h}}_{k,j}^{LM}; \overleftarrow{\mathbf{h}}_{k,j}^{LM}]$ (middle pink and top yellow blocks in Figure 1.7) are the top layer biLSTM output.

There are three steps to use ELMo (Peters et al., 2018b): pretraining of the deep bidirectional language model (biLM), fine tuning biLM on task data, and training the task model with ELMo embeddings (Figure 1.6). To use ELMo on a specific NLP task, one just runs

Fig. 1.6 ELMo in three steps.

the biLM introduced before and records all of the layer outputs for each word. Then a linear combination of these outputs is learned by the end task. So a task-specific weighting of all biLM layers is:

$$\mathbf{ELMo}_k^{\text{task}} = E\left(R_k; \Theta^{\text{task}}\right) = \gamma^{\text{task}} \sum_{j=0}^{L} s_j^{\text{task}} \mathbf{h}_{k,j}^{LM} \tag{1.9}$$

where S^{task} are softmax-normalized weights and the scalar parameter γ^{task} scales the overall usefulness of ELMo to the task.

ELMo is easy to add to existing models and significantly improves, *with relative error reductions ranging form 6 - 20%*, the state of the art across six NLP tasks including question answering (Liu et al., 2017), textual entailment (Chen et al., 2016), semantic role labeling (He et al., 2017), coreference resolution (Pradhan et al., 2012), named entity extraction (Sang and De Meulder, 2003), and sentiment analysis (Socher et al., 2013).

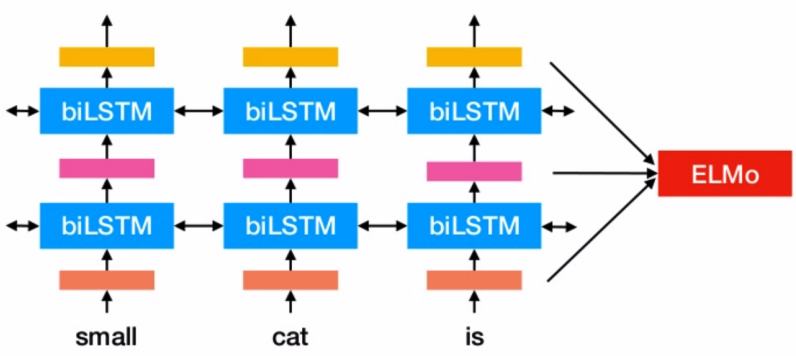


Fig. 1.7 ELMo's two-layer biLMs architecture[1]

1.2.4.2 BERT: Bidirectional Encoder Representations from Transformers (Devlin et al., 2019)

After the breakthrough led by ELMo in six NLP tasks, another pre-trained language representation model BERT came out about one year later and advanced the state of the art for eleven NLP tasks.

BERT contains two steps: pre-training and fine-tuning. In the pre-training stage, the BERT model is trained on unlabeled data over *masked language model (MLM)* (i.e. predicting a word that is randomly selected and masked, based on its context) and *next sentence prediction* tasks. For fine-tuning, BERT starts with pre-trained parameters and fine-tunes all of the parameters based on the downstream tasks. The architectures of these two steps are nearly the same, as shown in Figure 1.8, where the fine-tuning architecture has a different output layer depending on the downstream task.

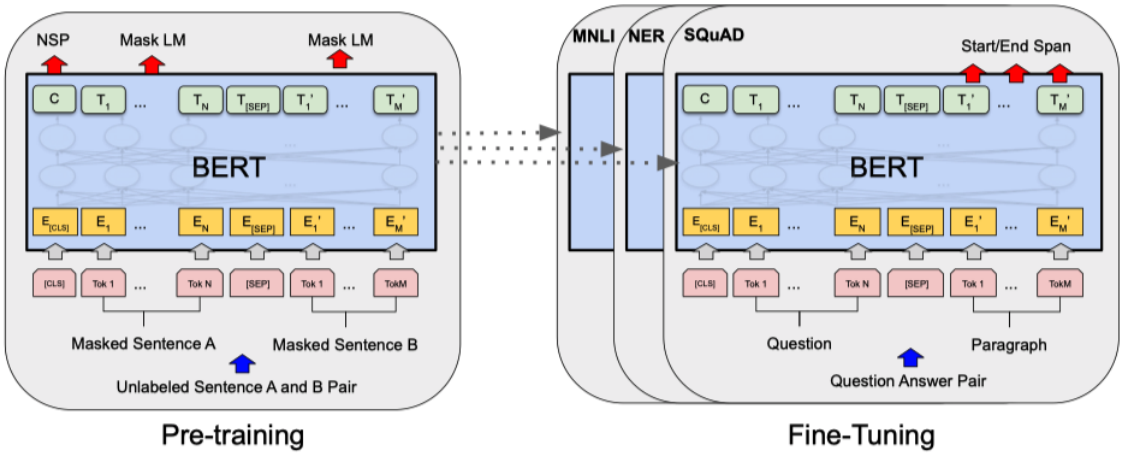


Fig. 1.8 Overall pre-training and fine-tuning procedures for BERT. Figure from Devlin et al. (2019)

There are four major differences between ELMo and BERT:

- How to apply them? As introduced before, representations given by ELMo have been used as additional features in task-specific architectures (feature-based). BERT does not need substantial task-specific architectures. BERT has a *unified architecture across different tasks* (fine-tuning).
- Which language model do they use? ELMo uses a bidirectional language model (LM), which is basically *a shallow concatenation of independently trained left-to-right and right-to-left LMs*. BERT uses a *masked language model (MLM)*. Compared with the simple concatenation of the directional language models, the MLM is more direction-free.
- What is the basic component unit of their language models? ELMo is based on biLSTMs while BERT uses transformers.
- Do these models take into account the relationship between two sentences? ELMo does not, as it is based on language models. BERT does, as one of its pre-training tasks is *next sentence prediction*.

1.3 Count-based methods

Count-based methods still attract people's attention because of their explicitness and interpretability as one common complaint about prediction-based models for word embeddings learning is that they are opaque.

Count-based methods supporters believe that count-based methods can take advantage of the vast amount of repetition in the data where prediction-based methods fail because they need to scan context windows across the entire corpus.

Count-based word embedding learning models share a common framework with 5 steps[2]:

1. Preprocessing: tokenization, annotation, tagging, parsing, feature selection, etc.
2. Matrix design: word $\times$ document, word $\times$ word, word $\times$ search proximity, adj. $\times$ modified noun, word $\times$ dependency relation, etc.
3. Reweighting: probabilities, length norm., TF-IDF, PMI, Positive PMI, etc.
4. Dimensionality reduction: LSA, PLSA, LDA, PCA, NNMF, etc.

[2]Chris Potts' slides of Stanford Linguistics CS224U course: Natural Language Understanding, http://web.stanford.edu/class/cs224u/materials/cs224u-vsm-overview.pdf

5. Vector comparison: Euclidean, Cosine, Dice, Jaccard, KL, etc.

All count-based methods can be seen as a cross-product of the different selection in each step. Below, I follow this segmentation to introduce state-of-the-art methods in this domain.

1.3.1 PMI + SVD: A straightforward and strong baseline method

Applying a singular value decomposition (SVD) to a pointwise mutual information (PMI) word-word matrix is a simple, straightforward method for word embedding learning.

Given a word-word co-occurrence matrix, PMI *measures the association between a word w and a context c by calculating the log of the ratio between their joint probability (the frequency in which they occur together) and their marginal probabilities (the frequency in which they occur independently) (Levy and Goldberg, 2014b).*

$$PMI(w,c) = \log \frac{\#(w,c) \cdot |D|}{\#(w) \cdot \#(c)} \tag{1.10}$$

Word embeddings are then obtained by low-rank SVD on this PMI matrix. *In particular, the PMI matrix is found to be closely approximated by a low rank matrix (Arora et al., 2016).* Note that as explained below, the PMI matrix is often replaced with its positive version (PPMI).

1.3.2 Pull word2vec into count-based methods category

Matrix generation: (shifted-)PMI/PPMI Levy and Goldberg (2014b) showed that skip-gram with negative-sampling (SGNS) is implicitly factorizing a shifted pointwise mutual information (PMI) word-context matrix.

By analyzing the global objective of SGNS:

$$\ell = \sum_{w \in V_W} \sum_{c \in V_C} \#(w,c) \left(\log \sigma(\vec{w} \cdot \vec{c}) + k \cdot \mathbb{E}_{c_N \sim P_D} \left[\log \sigma \left(-\vec{w} \cdot \vec{c}_N \right) \right] \right) \tag{1.11}$$

where $\vec{w}$ and $\vec{c}$ are vectors of the word w and its context c respectively, $\#(w,c)$ is the number of times the pair (w,c) appears, k is the number of "negative" samples and c_N is the sampled context, drawn according to the empirical unigram distribution P_D.

The authors found that it is optimized by setting

$$\vec{w} \cdot \vec{c} = PMI(w,c) - \log k \tag{1.12}$$

for every (w,c) pair.

SGNS returns word vectors matrix W and context word vectors matrix C as its output. If we consider their product $M = W \cdot C^T$, ***SGNS can be described as factorizing an implicit matrix M of dimensions $|V_W| \times |V_C|$ into two smaller matrices.***

So taking the finding from equation 1.12, the matrix M that SGNS is factorizing is:

$$M_{ij}^{\text{SGNS}} = W_i \cdot C_j = \vec{w}_i \cdot \vec{c}_j = PMI\left(w_i, c_j\right) - \log k \tag{1.13}$$

Based on the fact that the number of observations of a word-context pair affects its loss function, *SGNS's objective can now be cast as a weighted matrix factorization problem, seeking the optimal d-dimensional factorization of the matrix $M^{PMI} - logk$ under a metric which pays more for deviations on frequent (w,c) pairs than deviations on infrequent ones.*

The authors introduced the PPMI matrix, an approximation of the PMI matrix, which can solve computational problem (PMI matrix is dense and also ill-defined as log for the 0 cell is $-\infty$) when calculating the factorization of the PMI matrix.

$$PPMI(w,c) = \max(PMI(w,c), 0) \tag{1.14}$$

Matrix factorization: N/A & SVD To get final word embeddings, two algorithms have been applied:

- N/A: directly using rows of (shifted-)PPMI matrix as word embeddings. In this case, word vectors are sparse.
- Spectral algorithms like SVD: to get dense low-dimensional word embeddings, SVD over (shifted-)PPMI has been used. Experiments showed that SVD leads to a better performance on word similarity tasks but not on word analogy tasks.

1.3.3 The GloVe method

Although GloVe and other count-based methods' inputs are based on the word-context co-occurrence matrix, they have different ideas about how to make efficient use of this statistics: GloVe suggests to use *ratios of co-occurrence probabilities rather than the probabilities themselves*:

$$F\left(w_i, w_j, \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}} \tag{1.15}$$

Matrix generation: word co-occurrence matrix This is the standard word-word count matrix.

Matrix factorization: a global log-bilinear regression model Starting from equation 1.15, after several simplifications, the final cost function is defined as:

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2 \tag{1.16}$$

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases} \tag{1.17}$$

To train word embeddings, instead of running a large SVD over the word co-occurrence matrix, Glove goes over all elements in this matrix and optimizes the cost function. Note that GloVe only trains on the nonzero elements.

Compared with SVD or SGNS, Glove trains faster, is scalable to huge corpora and has good performance even with small corpora, and small vectors. And Unlike SGNS, word embeddings trained by Glove merge both word vectors and context vectors:

$$X_{final} = U + V \tag{1.18}$$

1.3.4 LexVec: explicitly factorizes the PPMI matrix using SGD

Levy and Goldberg (2014b) mentioned that stochastic gradient descent (SGD) can be used for matrix factorization, which is *an interesting middle-ground between SGNS and SVD*, but left it to future work.

Salle et al. (2016) explored this direction and proposed the LexVec model for word embeddings learning.

Matrix generation: PPMI PPMI is computed as presented earlier.

Matrix factorization: a reconstruction loss function The LexVec loss function contains two terms:

$$L_{wc}^{LexVec} = \frac{1}{2}\left(W_w \tilde{W}_c^{\top} - PPMI_{wc}^{*}\right)^2 \tag{1.19}$$

$$L_{w}^{LexVec} = \frac{1}{2}\sum_{i=1}^{k} \mathbf{E}_{w_i \sim P_n(w)}\left(W_w W_{w_i}^{\top} - PPMI_{ww_i}^{*}\right)^2 \tag{1.20}$$

To obtain word embeddings, LexVec iterates over the training corpus in exactly the same way as SGNS, unlike GloVe which minimizes its loss function by iterating over all non-zero cells in the word co-occurrence matrix. LexVec penalizes more heavily for errors of frequent co-occurrences as SGNS performs weighted matrix factorization, giving more influence to

frequent pairs, as opposed to SVD which gives the same weight to all matrix cells. Besides, LexVec still treats negative co-occurrences (see Equation 1.20, negative co-occurrences are selected in the same way as negative sampling in SGNS) unlike GloVe.

To evaluate the performance of LexVec, authors experimented it with two different window sizes (2 and 10) and two different loss function minimizing approaches (mini-batch and stochastic) on word similarity (WordSim-353 Similarity and Relatedness (Finkelstein et al., 2001), MEN (Bruni et al., 2012), MTurk (Radinsky et al., 2011), RW (Luong et al., 2013), SimLex-999 (Hill et al., 2014), MC (Miller and Charles, 1991), RG (Rubenstein and Goodenough, 1965), and SCWS (Huang et al., 2012)) and word analogy tasks (Google semantic and syntactic (Mikolov et al., 2013b) and MSR syntactic analogy dataset (Mikolov et al., 2013f)). The results show that LexVec obtained higher scores than PPMI-SVD on in all these tasks, and higher scores than GloVe on word similarity tasks.

1.3.5 AllVec: Alternative to SGD

According to Xin et al. (2018), although most state-of-the-art word embedding learning methods use SGD with negative sampling to learn word representations, SGD suffers from two problems listed below which influence its performance:

- SGD is highly sensitive to the sampling distribution and the number of negative samples. Unfortunately, sampling methods are biased.
- SGD suffers from dramatic fluctuation and overshooting on local minimums.

A direct solution is full batch learning, which does not have any sampling method and does not update parameters in each training step. The drawbacks of this solution is also obvious: a large computational cost.

Xin et al. (2018) proposed AllVec that generate word embeddings from all training samples using batch gradient learning.

Matrix generation: PPMI PPMI is computed as presented earlier.

Matrix factorization: AllVec loss function

$$L = \underbrace{\sum_{(w,c)\in S} \alpha^{+}_{wc} \left(r^{+}_{wc} - U_w \tilde{U}^{T}_{c}\right)^2}_{L_P} + \underbrace{\sum_{(w,c)\in (V\times V)\setminus S} \alpha^{-}_{wc} \left(r^{-} - U_w \tilde{U}^{T}_{c}\right)^2}_{L_N} \tag{1.21}$$

where

$$\alpha_{wc}^{+} = \begin{cases} (M_{wc}/xmax)^{\rho} & M_{wc} < x\max \\ 1 & M_{wc} \geq x\max \end{cases} \tag{1.22}$$

$$\alpha_{wc}^{-} = \alpha_{c}^{-} = \alpha_0 \frac{M_{*c}^{\delta}}{\sum_{c \in V} M_{*c}^{\delta}} \tag{1.23}$$

r_{wc}^{+} is defined as the corresponding cell value in the PPMI matrix, r^{-} is a constant value, V is the vocabulary, S is the set of positive pairs, U_w and $\tilde{U}_c$ denote the k-dimensional embedding vectors for word w and context c, M_{wc} is the number of cooccurrences the word pair (w,c), α_0 can be seen as a global weight to control the overall importance of negative samples and the exponent δ is used for smoothing the weights.

The major contribution of this paper is an efficient batch gradient optimization algorithm for the AllVec loss function based on a partition reformulation for the loss and a decouple operation for the inner product. It achieves a more stable convergence and a better embedding quality with the same complexity as the classical SGD models.

1.4 Hyperparameters setting for monolingual word embedding learning

The choice of a training model can indeed have a huge impact on the quality of word embeddings. The selection of parameters can however be crucial too for the final result. Below I talk about the influence of some common parameters used in both prediction-based and count-based methods and the understanding of these parameters.

1.4.1 Number of dimensions of word vectors

The dimensionality of word embeddings has been studied in (Yin and Shen, 2018). From the point of view of count-based methods , word embedding learning is basically a question about how to "squeeze" a sparse matrix into a smaller dense one. The level of reduction is decided by the dimensionality. For prediction-based methods, dimensionality is set by the hidden layer size. People often just use the default number (e.g. 200). There exists an optimal point for the dimensionality selection. But the key question is how one defines the loss function which can measure the difference between embeddings in two different dimensions (E_1 represents word embeddings in the original dimensionality and E_2 is the word embeddings with the "squeezed" dimension). Yin and Shen (2018) proposed the Pairwise Inner Product loss (PIP loss) (Equation 1.24) to evaluate the word embeddings quality as

vector inner product is the basic operation used to compute word relatedness and analogy with word embeddings.

$$||PIP(E_1) - PIP(E_2)|| = ||E_1E_1^T - E_2E_2^T|| \tag{1.24}$$

The PIP loss can be used for dimensionality selection in matrix factorization methods (SVD, LSA for instance). Word2vec can also benefit from it since it can be seen as implicit matrix factorization (Levy and Goldberg, 2014b).

1.4.2 Contexts selection for training words

Most of the word embedding training methods are based on the Distributional Hypothesis (Harris, 1954). This hypothesis was famously articulated by Firth (1957) as "You shall know a word by the company it keeps". The company of a word is defined as "context" in word embedding learning. In most current work in natural language processing, this context is defined as the words preceding and following the target word within a fixed distance (i.e. window size). But the definition of context can be varied from two views: the context type and the context representation. Besides the normal linear context definition just introduced before, context can be also defined by dependency relations (Levy and Goldberg, 2014a). In Figure 1.9, the context of word "discovers" is "scientist", "star" and "telescope" according to the dependency parse tree. Whether we look at true linguistic dependencies (Harris, 1954) or

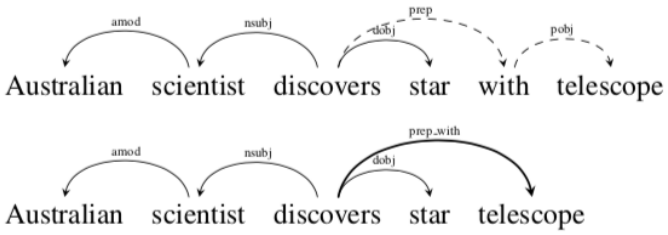


Fig. 1.9 Illustration of dependency parse tree. Figure from (Levy and Goldberg, 2014a)

at graphically neighboring words, we expect that words that are closer in that representation of a sentence (direct predicate-argument relations, or immediate graphical neighbors) provide a stronger context than words that are farther apart (more hops in the dependency tree, or more words in the word sequence). We may quantify the strength of the connection by the distance between context word and target word. In dependency-based context, this strength varies according to the dependency type.

If we map this target-context connection information into a word co-occurrence network, a graph of word interactions representing the co-occurrence of words in a corpus where an undirected edge can be created when two words co-occur within a sentence (Cancho and Solé, 2001), the context type is actually the type of the edge between a target word node and a context word node and the context representation is the definition of the edge weight. I will introduce this mapping in detail in Chapter 3. Experiments in (Levy and Goldberg, 2014a) showed that *most tasks have clear preference for specific context types and representations. Context representation plays a more important role than context type for learning word embeddings*.

1.4.3 Tips for hyperparameters selection using PPMI, SVD, word2vec and GloVe

Levy and Goldberg (2014b) have shown that skip-gram with negative sampling is implicitly factorizing a word-context matrix. In (Levy et al., 2015), they reveal that compared with different word embeddings learning methods, hyperparameter settings are more essential to the word embedding performance. This paper tries to answer several commonly asked (and also important) questions:

- Can we approach the upper bound on word embedding evaluation tasks on the test set by properly tuning hyperparameters on the training set? Yes.
- A larger hyperparameter tuning space or a larger corpus, which is more worthwhile? It depends, for 3/6 word similarity tasks, a larger hyperparameters space is better. For other tasks, a larger corpus is better.
- Are prediction-based methods superior to count-based distributional methods? No.
- Is GloVe superior to SGNS? No. SGNS obtains a better score than GloVe in every task in this paper.
- Is PPMI on-par with SGNS on analogy tasks? No. SGNS has a better performance.
- Is similarity multiplication (3CosMul) always better than addition (3CosAdd)? Yes, for all methods and on every task.
- CBOW or SGNS? SGNS.

This paper also gives several practical recommendations for word embedding training:

- *Always use context distribution smoothing* (power 0.75).

- *SGNS is a robust baseline.*
- *With SGNS, prefer many negative samples.*
- Adding context vectors (w+c) for both SGNS and GloVe.

This paper provides a practical and important analysis of word embeddings learning methods. It also has its own constraint though: the authors only evaluated word embeddings qualities on word similarity and analogy tasks. As there are no extrinsic evaluations, their solutions may only apply to intrinsic evaluations; The paper has been published in 2015, so methods it discussed were only PPMI, SVD, word2vec and GloVe.

1.4.4 Improvements based on pre-trained word embeddings

Some methods do not take a raw corpus but pre-trained word embeddings as input. They aim to fine-tune monolingual word embeddings with additional supervision data.

For instance, Faruqui and Dyer (2014) improve the performance of monolingual word embedding by incorporating multilingual evidence. Their method projects monolingual word embeddings onto a common vector space using bilingual alignment data.

Experiments in Faruqui and Dyer (2014) show that inclusion of multilingual context is helpful for monolingual word embeddings generated by SVD and RNN but not for one generated by the Skip-gram model. Check Section 2.4.3 for more details about this method as it is also used for multilingual word embeddings learning.

1.5 Evaluation Metrics

There is no absolute standard to say whether a word embedding is good or not. To evaluate the quality of a certain word embedding, an evaluation task is always needed. Either an intrinsic task where word embeddings are directly used, or an extrinsic task where pre-trained word embeddings are used as an input (this is the same principle as pre-trained convnets for image classification).

Here, my goal is to study the properties of the embedding space by intrinsic tasks rather than their influence on common NLP tasks by extrinsic tasks.

1.5.1 Intrinsic tasks

Properties of the embedding space are usually assessed through lexical relations (semantic similarity, semantic relatedness, word analogies) of word embeddings.

Humans are asked to provide judgments on the validity of these lexical relations. (Relatively) Simple computational methods are used to assess the validity of these relations based on word embeddings. Scores are then computed to evaluate how comparable the computational assessment of these relations are to the collected human judgments.

When human and computational judgments are provided as scores, rank correlation can be used to evaluate the relevance of the computational method. When human judgments are binary, accuracy, i.e., the average binary agreement of computational and human judgments, can be used to evaluate the relevance of the computational method.

Note that intrinsic tasks discussed below are only for context-independent word embedding evaluation which are based on the *complex characteristics of word use (e.g., syntax and semantics) (Peters et al., 2018b)*. The evaluation about how contextual word embeddings vary across linguistic contexts (i.e., to model polysemy) is out of my scope.

1.5.1.1 Word "similarity" tasks

There are at least two kinds of similarity. Relational similarity is correspondence between relations, in contrast with attributional similarity, which is correspondence between attributes. When two words have a high degree of attributional similarity, we call them synonyms. When two pairs of words have a high degree of relational similarity, we say that their relations are analogous (Turney, 2006).

Therefore, datasets for measuring word similarity usually have two parts, word similarity (attributional similarity) subsets and word relatedness (relational similarity) subsets.

Frequently used datasets are listed below:

- WordSim-353 (Agirre et al., 2009) contains two subsets. First subset involves 153 word pairs whose similarities are measured by 13 people and the mean values are used as the human judgment. Second subset contains 200 word pairs annoted by 16 people. WordSim-353 did not distinguish between similarity and relatedness word pairs.

- SimLex-999 consists of 666 noun-noun pairs, 222 verb-verb pairs and 111 adjective-adjective pairs. Scores are based on the opinions from 500 annotators via Amazon Mechanical Turk. As declared in SimLex-999 paper (Hill et al., 2014), *it provides a way of measuring how well models capture similarity, rather than relatedness or association.* In other words, SimLex-999 is for measuring word similarity (attributional similarity) rather than word relatedness (relational similarity). Table 1.2 gives a concrete example, the word *clothes* is not similar to the word *closet* even they are related to each other.

Normally the dataset contains three columns: a word pair with a score (as shown in Table 1.3).

Table 1.2 Comparison of gold standard scores between SimLex-999 and WordSim-353

Pair	SimLex-999 rating	WordSim-353 rating
coast - shore	9.00	9.10
clothes - closet	1.96	8.00

Table 1.3 First 10 word pairs and their scores in WordSim-353 dataset.

Word 1	Word 2	Human (mean)
love	sex	6.77
tiger	cat	7.35
tiger	tiger	10.00
book	paper	7.46
computer	keyboard	7.62
computer	internet	7.58
plane	car	5.77
train	car	6.31
telephone	communication	7.50
television	radio	6.77

Scores are given by human-assigned similarity judgments. The last column in Table 1.3 is the mean score of 13 subjects' individual assessments as shown in Table 1.4.

Table 1.4 Subjects' individual assessments and the mean score for the first word pairs in the WordSim-353 dataset.

Word 1	Word 2	Mean	1	2	3	4	5	6	7	8	9	10	11	12	13
love	sex	6.77	9	6	8	8	7	8	8	4	7	2	6	7	8
tiger	cat	7.35	9	7	8	7	8	9	8.5	5	6	9	7	5	7
tiger	tiger	10.00	10	10	10	10	10	10	10	10	10	10	10	10	10
book	paper	7.46	8	8	7	7	8	9	7	6	7	8	9	4	9
computer	keyboard	7.62	8	7	9	9	8	8	7	7	6	8	10	3	9
computer	internet	7.58	8	6	9	8	8	8	7.5	7	7	7	9	5	9
plane	car	5.77	6	6	7	5	3	6	7	6	6	6	7	3	7
train	car	6.31	7	7.5	7.5	5	3	6	7	6	6	6	9	4	8
telephone	communication	7.50	7	6.5	8	8	6	8	8	7	5	9	9	8	8
television	radio	6.77	7	7.5	9	7	3	6	7	8	5.5	6	8	6	8

Then the evaluation scenario is: Taking the word pair in each line and calculating a similarity score based on the word embeddings of the corresponding words. After getting all similarity scores based on the given word embeddings, using this score list to calculate the correlation score with the human-assigned score list (the third column).

1.5.1.2 Word analogy tasks

Word similarity tasks focus on the relation between two words. Word analogy tasks take three words to predict the fourth one based on syntactic or semantic relationships, "king is to queen as father is to ?" for instance.

$$\overrightarrow{?} = \overrightarrow{father} - (\overrightarrow{king} - \overrightarrow{queen}) \tag{1.25}$$

The answer is defined as the closest word to $\overrightarrow{?}$. It is considered to be the correct answer only if it is **exactly the same** as the gold standard (not even synonyms) word in the dataset. Then overall and question type based accuracies will be calculated based on all answers.

A frequently used dataset for this task is proposed by Mikolov et al. (2013b) which contains 5 types of semantic questions (8869 questions in total), in which 5.71% of *common capital city*, 51.01% of *all capital cities*, 9.76% of *currency*, 27.82% of *city-in-state*, and 5.71% of *family* and 9 types of syntactic questions (10675 syntactic questions in total) in which 9.29% of *adjective to adverb*, 7.61% of *opposite*, 12.48% of *comparative*, 10.51% of *superlative*, 9.89% of *present participle*, 14.98% of *nationality adjective*, 14.61% of *past tense*, 12.48% of *plural nouns*, and 8.15% of *plural verbs*. Examples of each type can be found in Table 1.5.

Table 1.5 Example of questions in the Semantic-Syntactic Word Relationship test set. Table from (Mikolov et al., 2013b)

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

1.5.1.3 Problems with word analogy tasks

In word analogy task, the predicted word is not allowed to be identical to any of other three words in the same test set (as shown in Equation 1.26). Taken the example in Equation 1.25, the predicated word $\overrightarrow{?}$ should not be the word "father", "king" or "queen".

$$\begin{gathered} A:B::C:D \\ s.t.\overrightarrow{D} \neq \overrightarrow{A}, \overrightarrow{D} \neq \overrightarrow{B}, \overrightarrow{D} \neq \overrightarrow{C} \end{gathered} \tag{1.26}$$

Recent research by Nissim et al. (2019) has found that this constraint on the output of word analogy tasks may largely influence the evaluation result. Table 1.6 shows the evaluation results with/without this constraint.

As we can see, the accuracy of the results drops dramatically when the nonidentical constraint has been removed. Although this comparison may not be totally fair (as the Google word analogy test set has been designed under the nonidentical assumption), the analysis of the output constraint is meaningful: it weakens the "magic feeling" of the word embeddings on word analogy tasks. For instance, *man is to king as woman is to queen* is one of the most frequently cited examples. If we remove the output constraint, we may get the word *king* instead of the *queen* as the result. Besides, this finding can be a bad news for scholars who study biases of word embeddings. Cherry-picked examples such as *man is to doctor as woman is to nurse* will no longer exist when the output constraint is removed.

1.5.2 Understanding of evaluation results

1.5.2.1 Prediction-based or count-based methods, which is better?

Since word2vec has been introduced as one of the prediction-based methods for word embeddings learning, "prediction-based or count-based methods, which is better?" has become a commonly ask question. To answer this question, Baroni et al. (2014) performed *an extensive evaluation, on a wide range of lexical semantics tasks and across may parameter settings* showing that prediction-based methods are the winner.

Because this paper was published in 2014, its extensive evaluation serves as a good summary of the competition between traditional count-based methods (at that time of course) and word2vec. The results (see Table 1.7) show that prediction-based approaches are a good direction for word embeddings training, which has been confirmed by subsequent research after 2014.

Note that in Figure 1.7, count-based methods are PMI-SVD-based models with different parameter settings and prediction-based methods are CBOW with different parameter settings.

Table 1.6 Evaluation results on the Google word analogy data set using the original and the constraint free versions. Table from (Nissim et al., 2019).

Category	orig	fair
Semantic		
capital-common-countries	83.20	44.47
capital-world	79.13	25.82
currency	27.37	18.13
city-in-state	70.90	10.21
Morpho-syntactic		
family	84.58	32.61
gram1-adjective-to-adverb	27.72	2.02
gram2-opposite	42.73	1.72
gram3-comparative	90.84	24.70
gram4-superlative	87.34	11.05
gram5-present-participle	78.22	6.91
gram6-nationality-adjective	89.93	73.80
gram7-past-tense	64.49	8.59
gram8-plural	86.04	4.73
gram9-plural-verbs	67.93	12.18

Table 1.7 Performance of count-based methods, prediction-based methods, Distributional Memory (dm) model (Baroni and Lenci, 2010) and Collobert and Weston (cw) vectors (Collobert et al., 2011). Figure from (Baroni et al., 2014).

	rg	ws	wss	wsr	men	toefl	ap	esslli	battig	up	mcrae	an	ansyn	ansem
best setup on each task														
cnt	74	62	70	59	72	76	66	84	98	41	27	49	43	60
pre	84	75	**80**	**70**	**80**	91	75	86	**99**	41	28	**68**	**71**	**66**
best setup across tasks														
cnt	70	62	70	57	72	76	64	84	98	37	27	43	41	44
pre	83	73	78	68	**80**	86	71	77	98	41	26	67	69	64
worst setup across tasks														
cnt	11	16	23	4	21	49	24	43	38	-6	-10	1	0	1
pre	74	60	73	48	68	71	65	82	88	33	20	27	40	10
best setup on rg														
cnt	(74)	59	66	52	71	64	64	84	98	37	20	35	42	26
pre	(84)	71	76	64	79	85	72	84	98	39	25	66	70	61
other models														
soa	**86**	**81**	77	62	76	**100**	**79**	**91**	96	**60**	**32**	61	64	61
dm	82	35	60	13	42	77	76	84	94	51	29	NA	NA	NA
cw	48	48	61	38	57	56	58	61	70	28	15	11	12	9

And "soa" represents the state-of-the-art results which *were obtained in almost all cases using specialized approaches that rely on external knowledge, manually-crafted rules, parsing, larger corpora and/or task-specific tuning.* Using external knowledge may get better result for specific tasks but my research focuses more on general word embeddings training.

1.5.2.2 Which factors of word embeddings may different tasks rely on? What are the correlations between factors and tasks like?

In most of word embedding learning papers, evaluations just stay in the experiment and results section showing that the approach proposed in the paper achieves state-of-the-art performance on several tasks. While the evaluation tasks can be rich and varied, there is almost no detailed analysis of why certain word embeddings can get better results on certain tasks and what is the relation between different tasks.

That is the reason why Rogers et al. (2018)'s paper is crucial to word embeddings learning and understanding. That paper tries to answer two questions:

- Which aspects of word embeddings may different tasks rely on? (factors of word embeddings)
- *What are the properties of embedding X that could predict its performance on tasks Y and Z?* (correlations between factors and tasks)

To answer the first question, they proposed a Linguistic Diagnostics (LD)[3] approach as shown in Figure 1.10.

For each word in one word embedding, LD first extracts its top *n* neighbors. Then by applying linguistic analysis to the relation of these neighbor-target word pairs, LD can finally get a statistics of different linguistic relations of all neighbor-word pairs extracted from one word embedding. These statistics serve as factors (morphological, lexicographic, psychological and distributional) to represent each word embedding's linguistic characteristics.

By comparing LD factors and word embedding performance on evaluation tasks of different models, a user can not only find which model performs better on a certain task, but also get a hint of possible reasons: the model works better on a certain task because its word embedding is more representative in several linguistic aspects.

To further confirm the possible reason (and also to answer the second question), a data visualization tool (as shown in Figure 1.11) has been used to show the correlations of LD factors and extrinsic/intrinsic tasks between themselves and others based on the data from 60 GloVe and word2vec embeddings.

[3]LD is implemented in LDT (Linguistic Diagnostics Toolkit), http://ldtoolkit.space

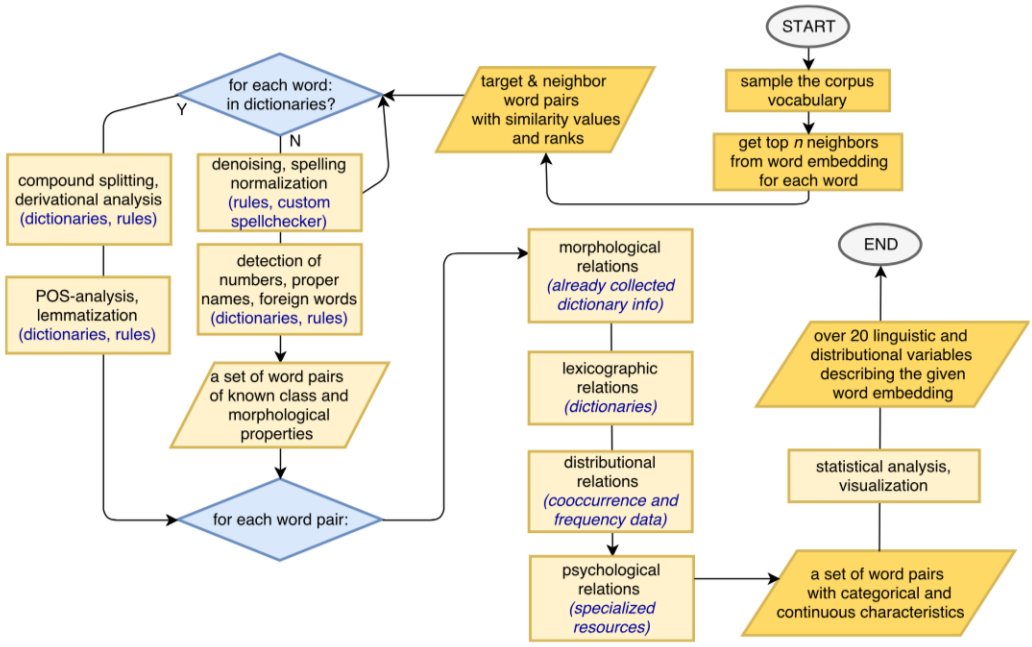


Fig. 1.10 LDT analysis pipeline. Figure from (Rogers et al., 2018)

Note that to calculate correlations, results from one word embedding are not enough. The user can use the LD analysis tool to analyze their word embedding, but there is no need to re-generate this heat map visualization as the factor/task correlation will not be changed by one word embedding.

There are many interesting observations from Figure 1.11, for instance, *the high correlation between almost all intrinsic tasks*. Also, compared with detailed (but boring) analysis of some specific word vectors on some specif tasks in most of word embeddings papers, this paper's analysis is precious because of its practicality and robustness.

1.5.3 Caution when one method "outperforms" the others

When one compares two systems A and B on a dataset and $score(A) > score(B)$, there are several caveats to keep in mind.

- If the difference is not very large or the size of the test set is not very large, the observed difference between the two systems may not be statistically significant: for instance, taking a slightly different subset of the test examples might lead to score(B) > score(A).
- The test set is meant to be a sample of the type of input the systems must process. But it might not be fully representative of that type of input (some dimensions of variance might not be taken into account), and a slightly different test set might lead to different

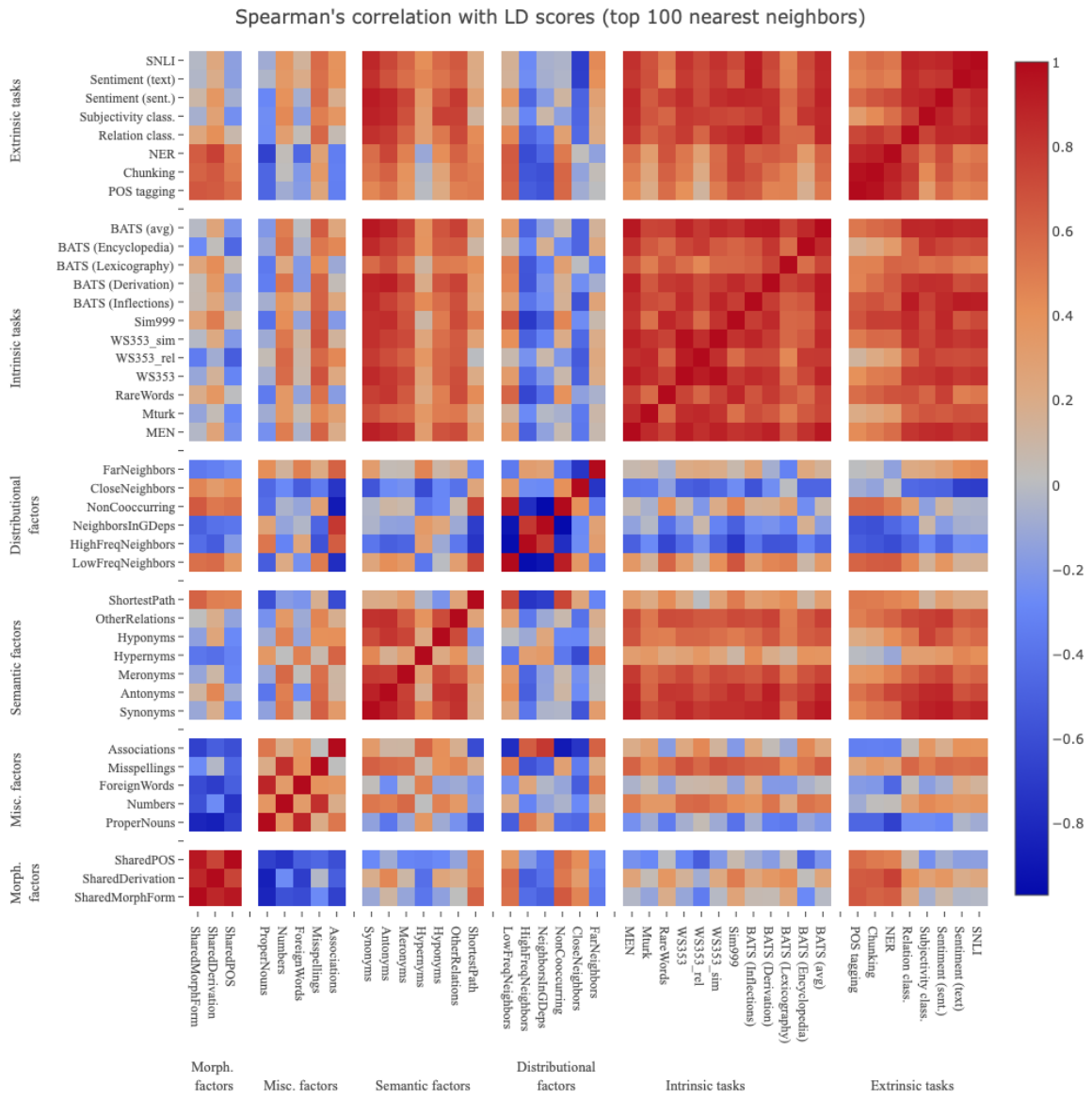


Fig. 1.11 Spearman's correlation with LD scores and evaluations on intrinsic/extrinsic tasks. Figure from (Rogers et al., 2018)

results; i.e., the observed scores may not necessarily generalize to other datasets of the same type.

- More specifically, the test set may have a systematic bias (e.g., Mikolov's analogy dataset (Mikolov et al., 2013b) and the prevalence of a few types of relations) and may therefore not assess what it is advertised to assess (e.g., not semantic relations between words in general, but essentially a couple of very specific semantic relations on a few very specific entity types).

In summary, such a comparison is only one observation point in a much larger space of possible observations, and obtaining a higher score on that observation point does not mean that A will outperform B in all other testing conditions. We should pay attention that the meaning we give to that comparison on that observation point may be overly general given the specificity of that point.

Additionally, if the comparison of system A and system B is intended to compare two different methods, then the only difference between A and B must be that method: they must use exactly the same preprocessing steps, the same training data, the same initialization if relevant, etc.

Chapter 2

Cross-lingual Word Embedding and State-of-the-art Approaches

2.1 Introduction

Cross-lingual word embedding (CLWE) is vector representations of words in multiple languages. In a cross-lingual word embedding space, words are placed according to their similarities in a common embedding space regardless of their languages. It has been widely used for multilingual natural language processing tasks such as document classification, dependency parsing, POS tagging, named entity recognition, super-sense tagging, semantic parsing, discourse parsing, dialog state tracking, entity linking or wikification, sentiment analysis and machine translation (Ruder et al., 2017).

Unlike monolingual word embedding learning which can be done in a totally unsupervised way, cross-lingual word embedding learning nearly always needs alignment data to supervise the training process. But the core of all these methods is the same: a word embeddings training model. In other words, if we improve the monolingual word embeddings training model, a cross-lingual model can also benefit from it as it shares the same basis.

I propose to classify cross-lingual word embeddings learning models according to two properties: the alignment data type and the processing stage when the alignment data is used (Figure 2.1).

Generally speaking, alignment data can be categorized into three types:

Document-level Documents can be aligned through cross-language links (as for instance in Wikipedia) without necessarily being translations of each other.

Sentence-level Sentences that are translations of each other can be aligned.

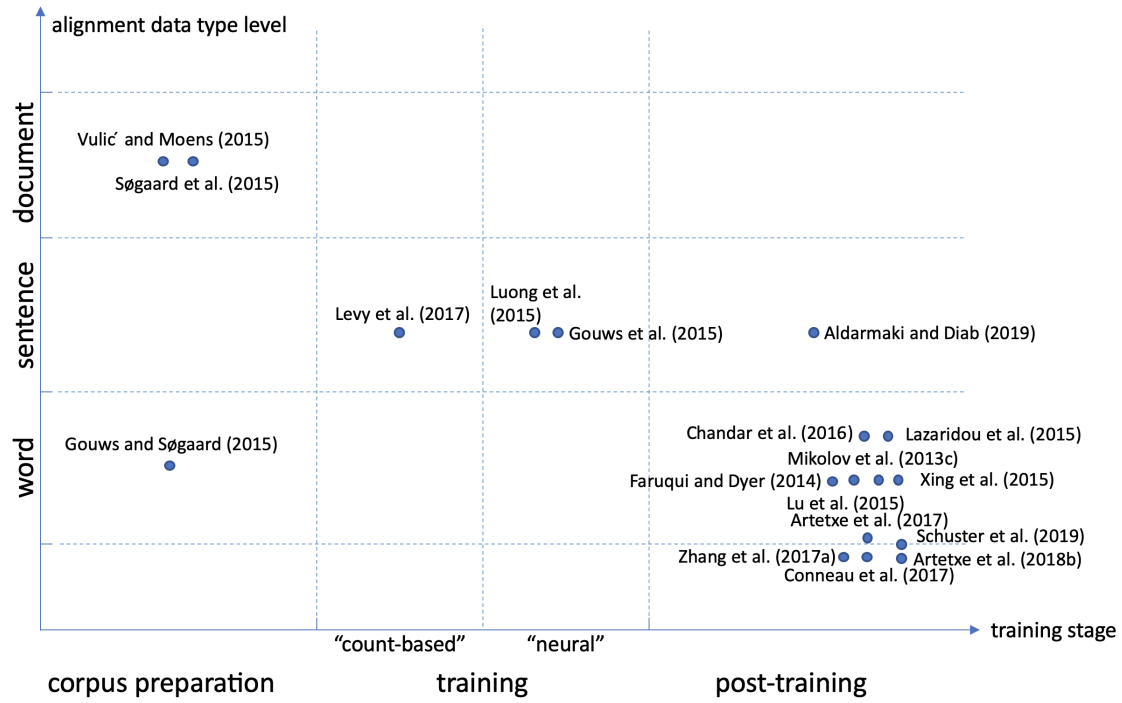


Fig. 2.1 Overview of Cross-lingual word embedding learning models position map

Word-level Word translations can be provided through bilingual dictionaries (or obtained by automatic word alignment in parallel sentences).

"Unsupervised" means eliminating any need for cross-lingual supervision, only using monolingual word embeddings. No alignment data (zero shot) can be seen as an extreme case either in word level or in document level.

Different types of alignment data may have different levels of influence on cross-lingual word embeddings. For instance, Levy et al. (2017) injected the sentence alignment information (sentence IDs) into four cross-lingual word embedding learning models ((AP et al., 2014; Gouws et al., 2015; Søgaard et al., 2015; Vulić and Moens, 2016)) and evaluated the resulting cross-lingual word embeddings on both manually annotated word alignment datasets and bilingual dictionaries. They found that *even a small amount of sentence-aligned data is more powerful than a huge amount of comparable documents*.

Since word-level alignment is easier to use and has been used in earlier studies, I will first present methods that use word-level alignment data before those that use sentence-level or document-level alignment data.

Cross-lingual word embeddings learning models can be categorized into three groups depending on when the alignment data is used: corpus preparation, training and post-training.

2.2 Corpus preparation stage

Cross-lingual word embedding learning methods categorized in the corpus preparation stage focus on the input data to be used for off-the-shelf word embedding learning algorithms. They generate multilingual corpus by using the alignment data.

2.2.1 Word-level alignments based methods

Gouws and Søgaard (2015) introduced a simple but representative corpus preparation stage method. Instead of using a monolingual corpus as in monolingual word embedding training, this method shuffles the non-parallel bilingual corpora and then replaces words with their equivalents, the corresponding translations, with probability $1/2k$ (where k represents the number of equivalents for the corresponding word) by using a small dictionary. Then they apply off-the-shelf word embedding algorithms to this mixed input. Note that the resulting bilingual word embedding is used for an unsupervised cross-language part-of-speech (POS) tagging task and semi-supervised cross-language super sense (SuS) tagging tasks. We do not know whether it is useful for a bilingual lexicon induction task.

2.2.2 Document-level alignments based methods

Until 2015, most of bilingual word embeddings learning methods were based on using parallel sentences or dictionaries. Vulić and Moens (2015) came up with an idea of only using theme-aligned comparable Wikipedia corpora. They merged documents of the same theme in different languages and randomly shuffled the generated documents (see Figure 2.2). Then they applied monolingual word embedding learning methods. Since the original sentence boundaries have been destroyed by shuffling and the "context" for training words is now at the document theme level, they set a larger maximum window size. This method is not convincing for the same reason: the context of words is the key part in word embedding learning. It should be discriminative regarding to different words. A large context range makes more distinct words share the same context, which weakens the discriminative power of the context.

Like in monolingual word embedding learning, besides prediction-based methods, count-based methods is another research direction. Søgaard et al. (2015) proposed to use inverted indexing of Wikipedia for cross-lingual word embedding learning. They generated a word-Wikipedia-articles-cluster co-occurrence matrix, where articles are clustered by their concept. Then matrix factorization methods can be applied to this matrix as in count-based monolingual

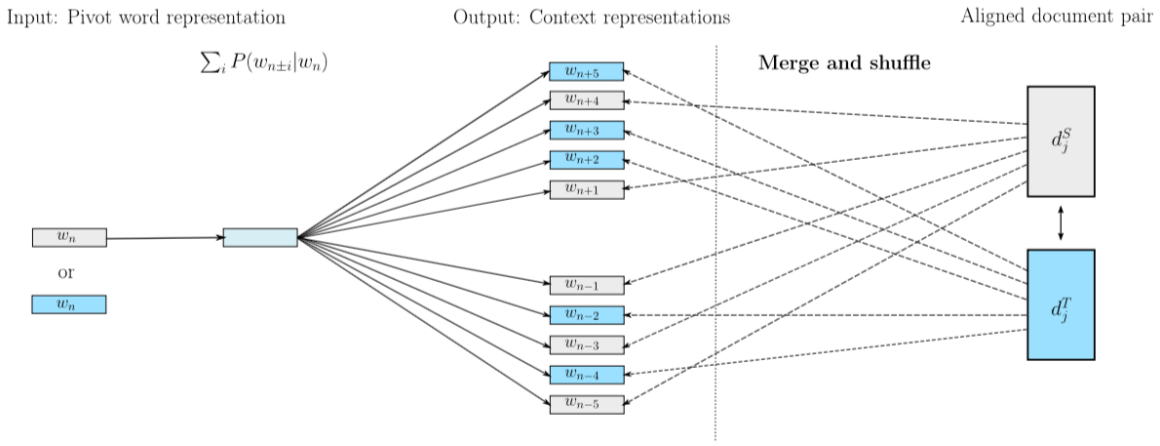


Fig. 2.2 In the right side of the figure, documents of the same theme j in source language S and target language T are merged and shuffled to generate new training document in the middle of the figure. Then the Skip-gram algorithm is applied to it as the left side shows.

word embedding learning. The advantage of this method is that it is not limited to a pair of languages: more languages can be trained at the same time. However, its shortcoming is also obvious: this method is highly dependent on the Wikipedia articles available in different languages.

2.3 Training Stage

In the training stage, like in monolingual word embeddings training, methods are often referred to as “count-based" or “neural” (“prediction-based”) methods. For cross-lingual word embeddings, methods can either improve the “count-based” matrix (Levy et al., 2017) or involve the multilingual constraint in the objective function for cross-lingual word embedding learning (Gouws et al., 2015; Luong et al., 2015).

Luong et al. (2015) adapted the skip-gram negative sampling model to the bilingual context. This method takes **parallel corpora** as input and uses an unsupervised model to align words in parallel sentences. With the obtained word-level alignment data, word-context training pairs are extended to a bilingual version (see Figure 2.3), which lead to the bilingual word embedding directly learned by the skip-gram model. One common point between this work and (Gouws and Søgaard, 2015) is that they both use bilingual word-context training pairs for bilingual word embedding learning.

Gouws et al. (2015) introduced the BilBOWA (Bilingual Bag-of-Words without Word Alignments) model for cross-lingual word embedding learning. This model does not need

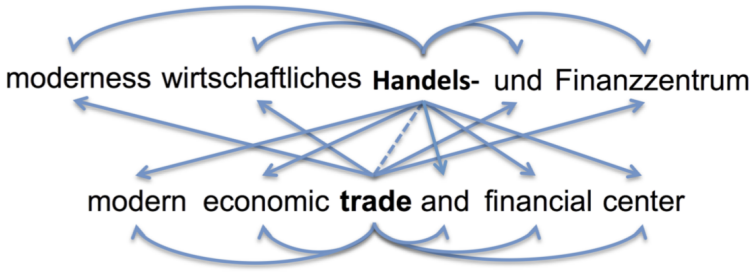


Fig. 2.3 Bilingual skip-gram model predicts both cross-lingually (see links in the middle) based on the word-level alignment information and monolingually (see links above and below). Figure from (Luong et al., 2015)

or use any word-level alignments. Instead, it applies BilBOWA-loss, which *minimizes a sampled L_2 – loss between the mean bag-of-words sentence-vectors of the parallel corpus*, to aligning monolingual word embeddings in the training stage (see Figure 2.4).

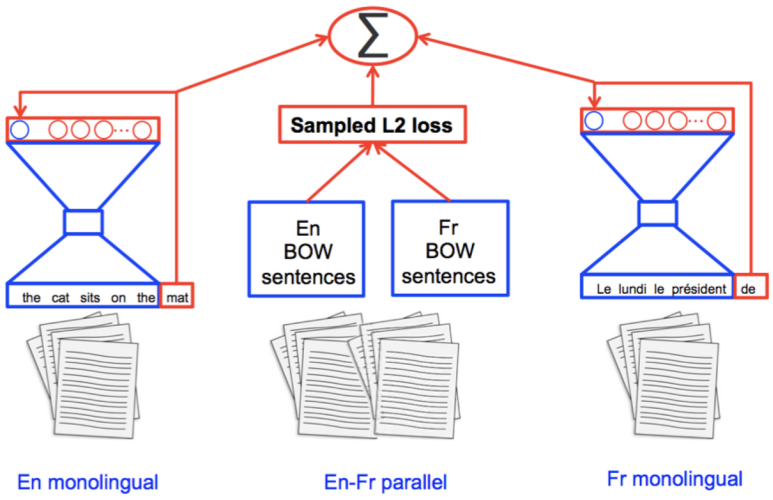


Fig. 2.4 BilBOWA-loss minimizes a sampled L_2 – *loss* between the mean bag-of-words sentence-vectors of the parallel corpus in the training stage. Figure from (Gouws et al., 2015)

2.4 Post-training Stage

In post-training methods, word embeddings are trained on each language independently, then a **linear mapping** is learned from source language to target language or from source

language to a shared embedding space. The cornerstone of this method is the strong similarity of geometric arrangements in word embeddings of different languages (see Figure 2.5).

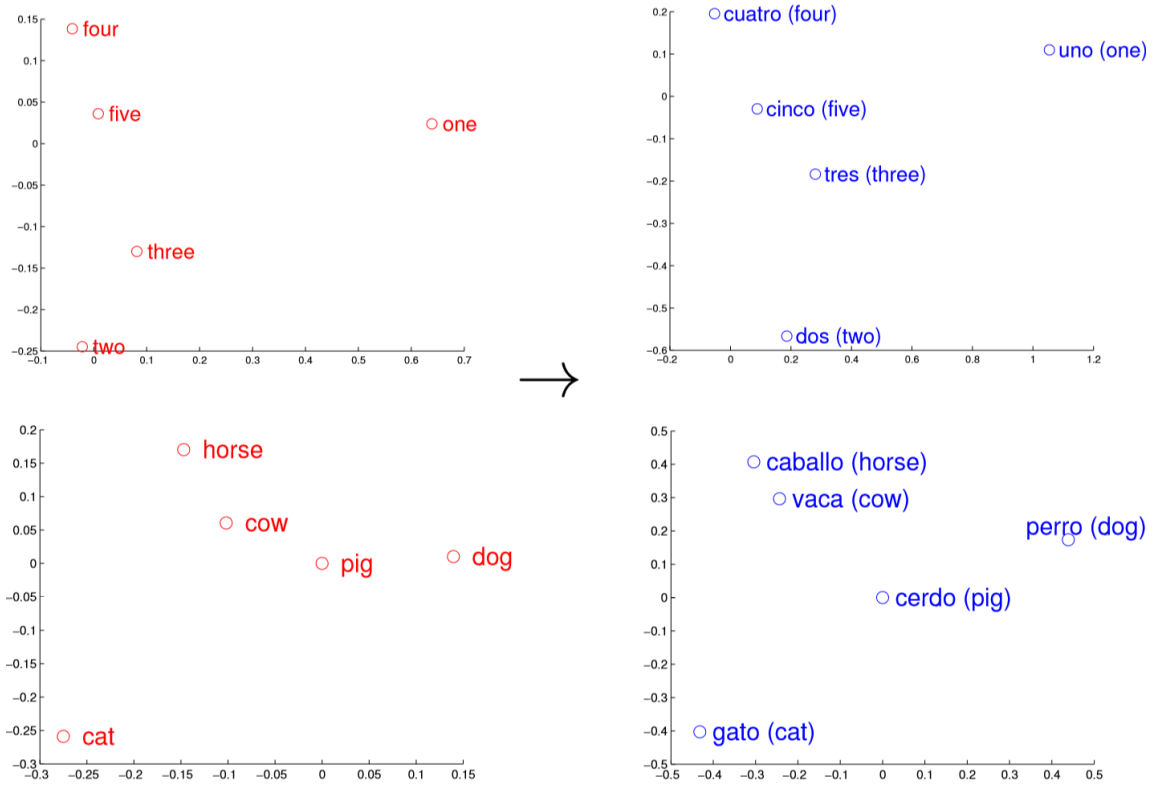


Fig. 2.5 Word embeddings of numbers and animals in English (left) and Spanish (right) projected to two dimensions using PCA. Figure from (Mikolov et al., 2013c)

Following the classification of methods in (Artetxe et al., 2018a), I present the methods below in four groups.

- Regression methods: map the embeddings in one language using a least-squares objective
- Orthogonal methods: map the embeddings in one or both languages under the constraint of the transformation being orthogonal
- Canonical methods: map the embeddings in both languages to a shared space using CCA and extensions of it
- Margin methods: *map the embeddings of the source language to maximize the margin between correct translations and other candidates*

2.4.1 Regression methods

SUPERVISED The same year as the word2vec paper, Mikolov et al. (2013c) introduced a simple but efficient method to automatically generate and extend dictionaries and phrase translation tables through a linear mapping using word-level alignment data. This model learns a linear transformation between word embeddings of different languages by minimizing the sum of squared Euclidean distances for the dictionary entries (word-level alignment data).

In this early paper on learning bilingual word embeddings mapping using word2vec, two simple methods are used for baselines: similarity of the morphological structure of words in different languages measured by their edit distance and similarity of word co-occurrences. The second baseline follows the same procedure as the method proposed in this paper except each word is represented by a vector whose values are the number of co-occurrences with each dictionary word in its language. The poor results of this baseline method compared with the proposed one show the advantage of having a dense word representation (word embeddings generated by word2vec for instance).

2.4.2 Orthogonal methods

SUPERVISED The orthogonal constraint for linear transformation was firstly introduced by Xing et al. (2015). They found two inconsistences in (Mikolov et al., 2013c):

- During the skip-gram model training stage, the distance measurement is the inner product of word vectors according to the objective function while the cosine distance is usually used for word embedding similarity calculation (e.g. for WordSim-353 task (Agirre et al., 2009)).
- The objective function of the linear transformation learning Mikolov et al. (2013c) uses the Euclidean distance. But after mapping, bilingual words' distance is measured by the cosine distance.

To solve these inconsistences, Xing et al. (2015) proposed an **orthogonal transform** based on **normalized word vectors**. Experiments showed that normalized word vectors have a better performance in a monolingual word similarity task (WordSim-353) and the proposed method performs significantly better in the word translation induction task than (Mikolov et al., 2013c).

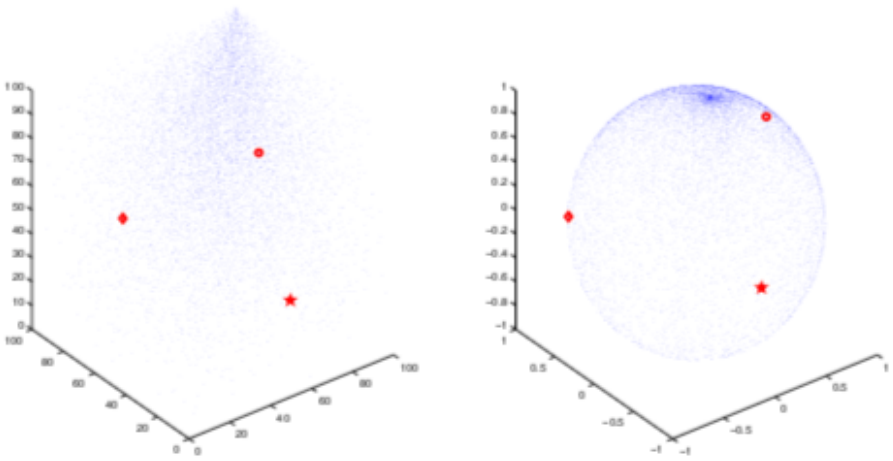


Fig. 2.6 The distribution of unnormalized (left) and normalized (right) word embeddings. Figure from (Xing et al., 2015)

In 2016, Artetxe et al. (2016) reframed the methods of (Faruqui and Dyer, 2014; Mikolov et al., 2013c; Xing et al., 2015) by proposing a unified framework:

- The basis of the proposed framework is transformation matrix calculation following (Mikolov et al., 2013c) except the solution is calculated by using SVD instead of gradient descent.
- The method of Xing et al. (2015) is equivalent to the framework's basis with orthogonality constraint and unit vectors. For pragmatic reasons, Xing et al. (2015) use gradient descent to calculate an approximation of the exact solution. Instead, SVD has been used in this framework for the calculation of the transformation matrix. (Note that Artetxe et al. (2016) is not the first one who talk about using SVD. In fact, in the original orthogonal constraint method paper (Xing et al., 2015), they mentioned that *One can show that this problem can be solved by taking the singular value decomposition (SVD) of W and replacing the singular values with ones.*)
- The method of Faruqui and Dyer (2014) has been linked to this framework by *mean centering for maximum covariance*. The authors showed that this work is close to the framework's basis with orthogonality constraint.

The recommended method proposed in this paper is basically the classic orthogonal constraint method (Xing et al., 2015) but using SVD for transformation matrix calculation and with dimension-wise mean centering in post-training stage. Another contribution of this paper is that they showed that the orthogonality constraint is the key to preserving monolingual performance in the Google word analogy task.

SEMI-SUPERVISED Artetxe et al. (2017) explored another direction for cross-lingual word embedding learning: Their model learns bilingual word embedding with almost no bilingual data. While it may not need any additional bilingual word pairs, it still belongs to the method category which uses word level alignment data as they use common letters or Arabic numerals across languages.

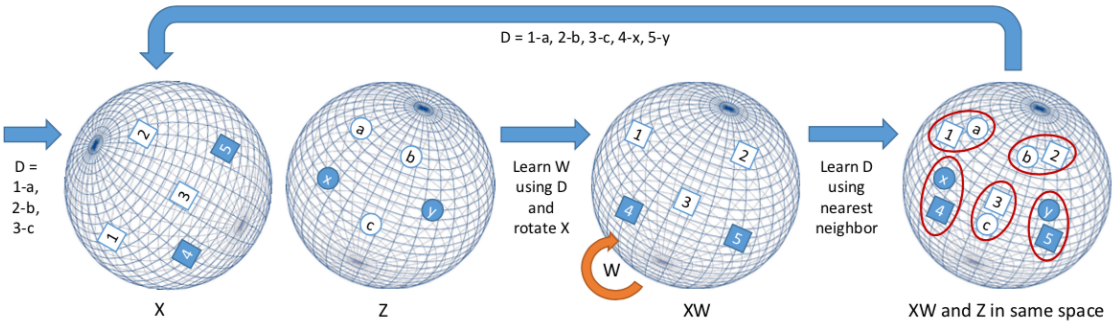


Fig. 2.7 A general schema of the proposed **self-learning** framework.

Figure 2.7 helps us to understand how this method works and its difference with the method just introduced before. If we remove the top feedback link in the schema, it becomes the schema of Mikolov et al. (2013c), where D is the word-level alignment data used to learn a linear mapping W from source language word embedding X to target language word embedding Z. In this method, the seed dictionary D size can be really small (only 25 word pairs). This dictionary extends after bilingual lexicon induction by using the linear mapping W learned by the initial dictionary D. Then the new larger dictionary is returned back as the new seed dictionary D. This self-learning loop continues until *the average dot product for the induced dictionary falls below a given threshold from one iteration to the next*.

UNSUPERVISED In cross-lingual word embedding learning, "unsupervised" means eliminating any need for cross-lingual supervision, only using monolingual word embeddings. Generative Adversarial Net (GAN) has achieved good results in computer vision as an unsupervised model (Goodfellow et al., 2014). Some NLP researchers also shifted their focus to applying GAN to unsupervised cross-lingual word embedding learning.

MUSE (Multilingual Unsupervised and Supervised Embeddings) is a GAN-based open-source tool introduced by Conneau et al. (2017). In their paper, a discriminator is trained to determine whether two word embeddings uniformly sampled from the $50,000$ most frequent words come from the same distribution (WS or T). In the meantime, W is trained to prevent the discriminator from doing so by making elements from two different sources as similar as possible (see Figure 2.8 (B)). Besides, they defined a similarity measure Cross-domain

Similarity Local Scaling (CSLS) that addresses the hubness problem, i.e.points tending to be nearest neighbors of many points in high-dimensional spaces, (see Figure 2.8 (D)) and serves as the validation criterion for early stopping and hyper-parameters tuning.

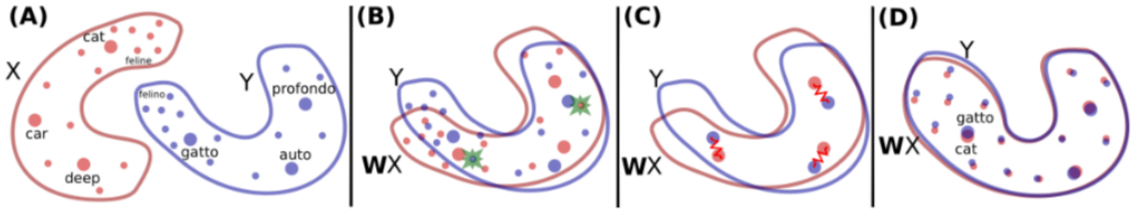


Fig. 2.8 Illustration of the MUSE method. (A) Source word embedding X and target word embedding Y. (B) Learning a linear mapping W between X and Y. (C) Mapping refinement via Procrustes to meet the orthogonality constraint on W. (D) Bilingual lexicon induction with CSLS to alleviate hubness problem.

Hartmann et al. (2018) did experiments based on two word embeddings of the same language (English). By applying MUSE on word embeddings generated by the same or different algorithms on identical or different corpora, they showed that *MUSE cannot align two embedding spaces for English induced by different algorithms*

Another possible way of using a GAN model is proposed by Zhang et al. (2017a): Instead of exploring the word-level invariance of different languages, they view entire word embedding spaces as distributions and define a measure to quantify the "distance" between these spaces. Zhang et al. (2017a) applied the **earth mover's distance (EMD)** to measure a distance between word embedding spaces.

The bilingual word embeddings learning problem becomes finding a transformation matrix G of dimensions $d \times d$ given source language word embedding S of dimensions $d \times n$ and target language word embedding T of dimensions $d \times n$, which can minimize the EMD between the transformed source word embedding GS and the target word embedding T. Two approaches have been proposed for this problem:

- Wasserstein GAN (WGAN): Following the structure of **Generative adversarial net (GAN)**. Here, the generator tries to find a transformation matrix G that can minimize the EMD and the discriminator attempts to accurately estimate the EMD given transformed source word embedding GS and target word embedding T. Note that this approach does not apply the orthogonal constraint on G. According to the experimental results, WGAN *seems better at exploring the parameter space and finally landing in a neighborhood of a good optimum.*

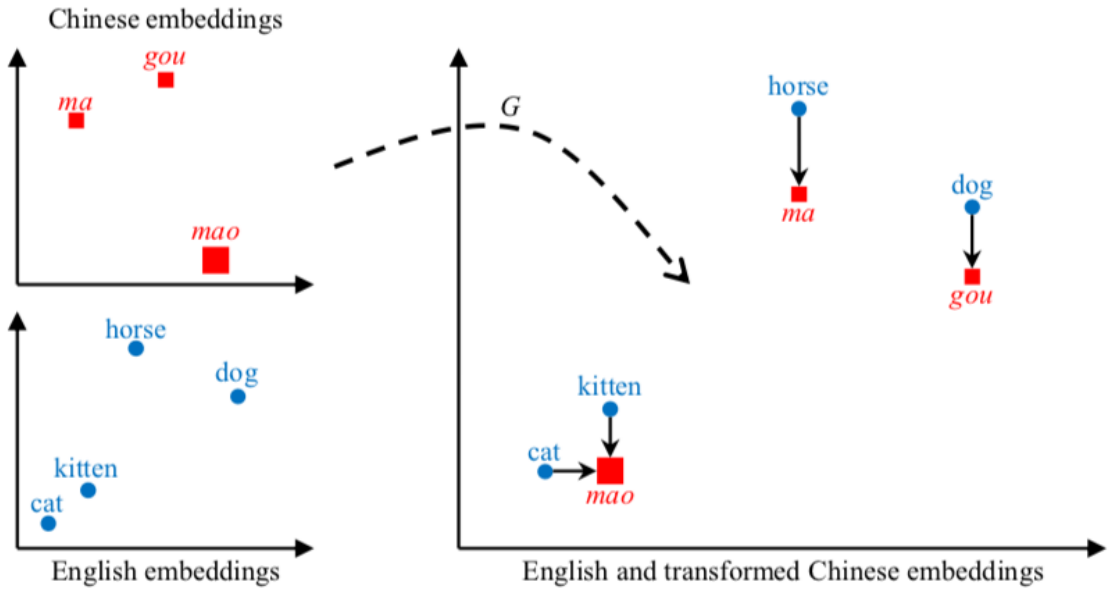


Fig. 2.9 Earth mover's distance measures the distance between two sets of weighted points. Figure from (Zhang et al., 2017a)

- EMD Minimization Under Orthogonal Transformation (EMDOT): It is a NP-hard problem to find the global minimum with the orthogonal constraint, so the SVD based method can find a local minimum of the solution.

Thus, the final proposed approach is starting with WGAN to find the neighborhood of a good optimum and then applying EMDOT to approach it. Note that the authors did not do a complete comparison with the state-of-the-art methods, but with only two methods (Mikolov et al., 2013c; Zhang et al., 2017b). Artetxe et al. (2018b) perform a complete state-of-the-art methods comparison that shows that the performance of this method is poor.

To eliminate the need for cross-lingual supervision, GAN is one research direction. Generating the initial dictionary in a fully unsupervised way (e.g. based on word similarity distribution as the method just introduced) in self-learning is another one.

Continuing applying the **self-learning** idea, Artetxe et al. (2018b) introduced a fully unsupervised method based on orthogonal mapping. Similar to the self-learning structure in (Artetxe et al., 2017), starting with the initial dictionary D, this method improves D to the optimal one by using transformation matrices learned by computing the optimal orthogonal mapping based on D in each training iteration until convergence.

It is hard to generate the **initial dictionary** D in a fully unsupervised way as authors have proved that a completely random seed dictionary D does not work. The authors proposed to

generate the initial dictionary by using the monolingual similarity matrix which is defined as the result of multiplying the word embedding matrix by its transposed matrix. The monolingual similarity matrix characterizes each word by its similarity to all other words in the corpus.

The intuition is that two equivalent words in two languages should have a similar distributions (i.e., rows) in the monolingual similarity matrices of each these two languages. As shown in Figure 2.10, the distributions of word "two" and its equivalent word in Italian "due" are similar in their corresponding monolingual similarity matrices, while the distribution of Italian word "cane" (dog) is totally different.

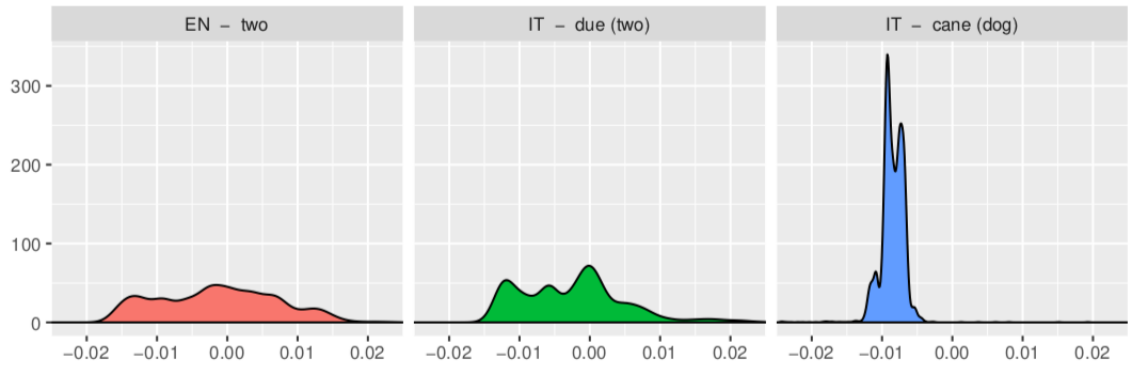


Fig. 2.10 Similarity distribution of three words. Figure from (Artetxe et al., 2018b)

The problem of using word embedding matrix directly is that given two word embedding matrices in two languages, their dimensions are not the same. One word's distribution is not comparable in this dimension system to its equivalent word in another language. Dimensions are monolingual words in similarity matrix, which is comparable in another language.

Then the problem of finding the initial dictionary D is equivalent to finding row mappings in two languages' similarity matrices where the corresponding rows share a similar distribution. Before that, sorting the values in each row of similarity matrices in descending order (or in ascending order) is necessary.

2.4.3 Canonical methods

CCA is a popular method for multi-view representation learning, *Haghighi et al. (2008) were the first to use this method for learning translation lexicons for the words of different languages* (Ruder et al., 2017).

For cross-lingual word embedding learning, two linear transformation matrices ($\mathbf{W}^{s\rightarrow}$ and $\mathbf{W}^{t\rightarrow}$) are learned to project the word embeddings of two languages onto a common

space where the CCA correlation (see Equation 2.1) between projected word pairs in the given dictionary is maximized(see Equation 2.2).

$$\rho\left(\mathbf{W}^{s\rightarrow}\mathbf{x}_i^s, \mathbf{W}^{t\rightarrow}\mathbf{x}_i^t\right) = \frac{\operatorname{cov}\left(\mathbf{W}^{s\rightarrow}\mathbf{x}_i^s, \mathbf{W}^{t\rightarrow}\mathbf{x}_i^t\right)}{\sqrt{\operatorname{var}\left(\mathbf{W}^{s\rightarrow}\mathbf{x}_i^s\right)\operatorname{var}\left(\mathbf{W}^{t\rightarrow}\mathbf{x}_i^t\right)}} \tag{2.1}$$

where $cov(\cdot,\cdot)$ is the covariance and $var(\cdot)$ is the variance.

$$\Omega_{\text{CCA}} = -\sum_{i=1}^{n}\rho\left(\mathbf{W}^{s\rightarrow}\mathbf{x}_i^s, \mathbf{W}^{t\rightarrow}\mathbf{x}_i^t\right) \tag{2.2}$$

where n is the number of word pairs in the dictionary.

Canonical methods benefit from the development of CCA based algorithms, which are usually **neural networks** models such as DCCA (Andrew et al., 2013) and CorrNet (Chandar et al., 2016).

Faruqui and Dyer (2014) use CCA to project monolingual word embeddings onto their maximally correlated subspaces (see Figure 2.11). While they focused on improving monolingual word embeddings quality by preserving bilingual invariance based on a dictionary, their approach obtains cross-lingual word embeddings as a by-product.

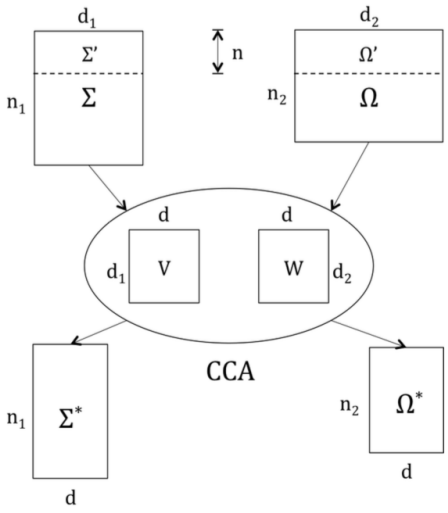


Fig. 2.11 Cross-lingual word vector projection using CCA. Figure from (Faruqui and Dyer, 2014)

A linear feature mapping is often not sufficiently powerful to faithfully capture the hidden, non-linear relationships within the data. Based on the general idea of (Faruqui

and Dyer, 2014), Lu et al. (2015) replaced CCA with deep canonical correlation analysis (DCCA) (Andrew et al., 2013) for both monolingual and cross-lingual word embeddings learning. In DCCA (see Figure 2.12), two neural networks are learned as non-linear transformations that can maximize the CCA correlation. Experiments show an overall improvement on word similarity tasks (WordSim-353 (Agirre et al., 2009) and SimLex-999 (Hill et al., 2014)) and on bigram similarity task (Mitchell and Lapata, 2010) compared with linear CCA.

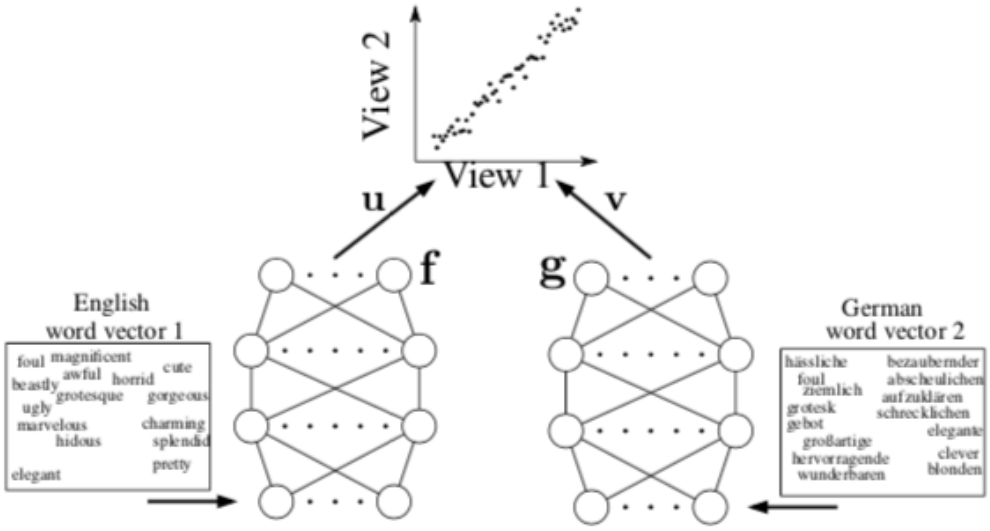


Fig. 2.12 Illustration of deep CCA. Figure from (Lu et al., 2015)

Some word clusters are consistently distributed across languages such as neighbor words in the monolingual word embedding space (e.g. *the neighboring words of "China" in English ("Japan", "India" and "Taiwan") should be close to the neighboring words of "Cina" in Italian ("Beijing", "Korea" and "Japan") in the common semantic space*), similar compositional characters or patterns of words, linguistic properties like word classes (colors, weekdays, months, etc.) and morphological information, based on which Huang et al. (2018) introduced a new cluster-consistent CorrNet to generate cross-lingual word embeddings by aligning words as well as clusters. A correlational neural network (CorrNet), which combines the advantages of CCA and autoencoder (AE) (Chandar et al., 2016), is used to learn common representations for different views of data.

Although this method achieves good performance, it has several limits of using cluster-level alignment data: First, not all languages share similar compositional characters or patterns of words (English and Chinese for instance). Second, linguistic properties require additional data which is not always easy to obtain. So from the perspective of universality

across languages, neighbor based clustering and alignment is the practical contribution of this paper.

2.4.4 Margin methods

As briefly introduced before, margin methods *map the embeddings of the source language to maximize the margin between correct translations and other candidates*. Lazaridou et al. (2015) introduced a max-margin based ranking loss (see Equation 2.3) to solve the hubness problem, i.e. points tending to be nearest neighbors of many points in high-dimensional spaces, in cross-lingual word embedding learning.

For a given translation pair (x_i, y_i), the loss between the gold standard translation y_i and the predicted one $\hat{\mathbf{y}}_i = Wx_i$ is defined as

$$\sum_{j \neq i}^{k} \max \left\{ 0, \gamma + \operatorname{dist}\left(\hat{\mathbf{y}}_i, \mathbf{y}_i\right) - \operatorname{dist}\left(\hat{\mathbf{y}}_i, \mathbf{y}_j\right) \right\} \tag{2.3}$$

where *dist* is the inverse cosine for the distance measurement, k is the number of negative examples and γ is the margin.

2.5 What Has Been Lost in 2019?

All models discussed before are based on static context-independent word embeddings.

One of the major problems of the cross-lingual context-independent word embeddings is that one word may have several meanings depending on different contexts, but word2vec-style methods give it only one vector representation regardless of its context. That may cause problems during the mapping phase when the target word sense does not match the source word sense or the multiple senses of the target word are not consistent with its source word.

In 2019, contextualized word embeddings trained by ELMo (Peters et al., 2018a) or BERT (Devlin et al., 2019) yield better performance in a range of NLP tasks including word sense disambiguation. The contextualized word embeddings are dynamic as the vector representation of a word is determined by its context. This "natural advantage" of the contextualized word embeddings can be a solution to the cross-lingual word sense disambiguation task (Lefever and Hoste, 2009). So it is expected to see research about cross-lingual mapping between contextualized word embeddings.

2.5.1 Supervised

Given a simple lexical dictionary as used for context-independent word embeddings learning, Schuster et al. (2019) proposed a embedding anchor based solution to assign vectors to dictionary words based on contextualized word embeddings.

$$\overline{e}_i = \mathbb{E}_c \left[e_{i,c} \right] \tag{2.4}$$

As shown in equation 2.4, the anchor of token i's context (c) dependent embeddings is defined as *the average over a subset of the available unlabeled data*.

Based on the finding of the contextualized word embeddings that point clouds for each token are well separated, authors suggested to assign embedding anchors to dictionary words.

Aldarmaki and Diab (2019) proposed two approaches using parallel sentences for cross-lingual mapping instead of a simple lexical dictionary:

- Extracting *a dynamic dictionary of aligned contextualized word embeddings*: Making an analogy, "word-level dictionary" to "static context-independent word embeddings" is like what to "contextualized word embeddings"? The answer is "dynamic context-dependent (or contextualized) word-level dictionary". In other words, given aligned sentences and contextualized word embeddings trained on that, the ideal "dictionary" should be the aligned words extracted from the aligned sentences.

- *Using a dictionary of aligned sentence embeddings*: the authors calculate the average of word vectors in each aligned sentence pair and map word embeddings based on these sentence-level embeddings. The intuition it that *a sentence is less ambiguous than stand-alone words since the words are interpreted within a specific context.* Note that this approach works for both context-independent embeddings and contextualized embeddings.

2.5.2 Unsupervised

Unsupervised contextualized word embeddings mapping is harder than unsupervised context independent word embeddings mapping because of the fact that each word has many different vector representations corresponding to different contexts.

As mentioned in (Schuster et al., 2019), MUSE Conneau et al. (2017) can be directly used on cross-lingual contextualized word embedding learning, it can be a solution for homonyms problem in cross-lingual context-independent word embeddings but unfortunately this method is not stable according to their experiments.

Schuster et al. (2019) also proposed another approach on top of the MUSE Conneau et al. (2017) model by using contextualized word embedding anchor (equation 2.4): same as what they did on the supervised case, the word embeddings anchor has been assigned as the word vector representation. Then apply them to the MUSE model.

2.6 Evaluation Metrics

As in the evaluation metrics section for monolingual word embeddings, my goal here is to study the properties of the cross-lingual word embedding space through intrinsic tasks rather than its influence on common multi-lingual NLP tasks by extrinsic tasks.

I introduce two commonly used intrinsic tasks for cross-lingual word embeddings evaluation: word similarity and multiQVEC.

2.6.1 Word similarity

The SemEval 2017 Task 2(Camacho-Collados et al., 2017) provided datasets composed of nominal pairs that are manually scored between English, Farsi (Persian), German, Italian and Spanish (10 cross-lingual word similarity datasets in total as shown in Figure 2.13)[1]. Its 3

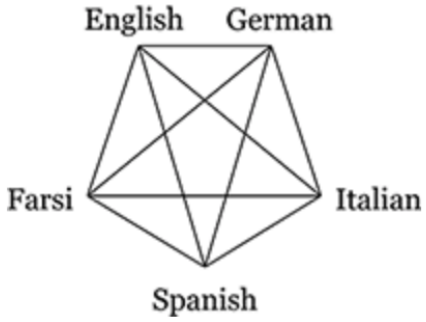


Fig. 2.13 Ten cross-lingual word similarity datasets between English, Farsi, Spanish, Italian and German.

columns data structure and correlation-based evaluations are same as in monolingual word similarity tasks discussed in Section 1.5.1.1.

2.6.2 multiQVEC and multiQVEC-CCA

QVEC (Tsvetkov et al., 2015) is an intrinsic evaluation for monolingual word embeddings based on alignment of the word embedding matrix to a matrix of features extracted from

[1] http://alt.qcri.org/semeval2017/task2/

manually crafted lexical resources (linguistic matrix), where *the correlation is measured using cumulative dimension-wise correlation*.

Ammar et al. (2016) proposed multiQVEC and multiQVEC-CCA, two multilingual extensions of QVEC, by using supersense tag annotations in English, Danish and Italian for the linguistic matrix generation. Unlike multiQVEC, multiQVEC-CCA replaces the cumulative dimension-wise correlation with the CCA correlation.

2.6.3 Summary of experiment settings for cross-lingual word embedding learning models

I summarize the monolingual data, alignment data, evaluations and languages used in each cross-lingual word embedding learning methods introduced in this Section in Table 2.1 and Table 2.2.

Table 2.1 Summary of experiment settings for cross-lingual word embedding learning models (1/2)

Methods	Monolingual data	Alignment data	Evaluation	Languages
mikolov2013exploiting	WMT11[1]	dictionary from GT[2]	lexicon induction	EN⇔SP, EN⇔CZ
	Google News datasets	dictionary from GT[2]	lexicon induction	EN→SP, EN⇔VN
xing2015normalized	WMT11[1]	dictionary from GT[2]	lexicon induction	EN-SP
zhang2017earth	Wikipedia comparable corpora[3]	dictionary from GT[2]	lexicon induction	EN-ZH, EN-ES, EN-IT, ZH-JA, EN-TR
D18-1023	Leipzig Corpora Collection[4] +Europarl[5]	bilingual dictionaries[6]	QVEC, QVEC-CCA	EN,IT,DA; BU,CZ,DA,GE,GR, EN,SP,FI,FR,HU, IT,SW
	LDC+Wikipedia	word-pairs[7]	Name tagging	Amh+Tig Eng+Uig+Tur
artetxe2016learning	word embeddings[8]	bilingual dictionary[8]	lexicon induction	EN-IT
artetxe2017learning	word embeddings[8]	bilingual dictionary[8]	lexicon induction; WordSim353 corsslingual	EN-IT
	SdeWaC[9]	dictionary from OPUS	lexicon induction; RG-65; WordSim353 corsslingual	EN-GE
	WMT16[10]	dictionary from OPUS	lexicon induction	EN-FI
artetxe2018generalizing	same as artetxe2017learning			
	WMT News Crawl 07-12[11]	binlingual dictionary[12]	lexicon induction	EN-ES

[1] http://www.statmt.org/wmt11/training-monolingual.tgz
[2] on-line Google Translate
[3] http://linguatools.org/tools/corpora/wikipedia- comparable-corpora
[4] http://wortschatz.uni-leipzig.de/en/download/
[5] http://www.statmt.org/europarl/index.html
[6] same as those used in Ammar et al. (2016)
[7] combination of the bilingual aligned words extracted from Wiktionary and monolingual dictionaries based on identical strings.
[8] dataset from Dinu et al. (2014), http://clic.cimec.unitn.it/ eorgiana. dinu/down/
[9] part of the WaCky collection (Baroni et al., 2009)
[10] http://www.statmt.org/wmt16/ translation- task.html
[11] http://www.statmt.org/wmt13/translation-task.html
[12] source not mentioned

Table 2.2 Summary of experiment settings for cross-lingual word embedding learning models (2/2)

Methods	Monolingual data	Alignment data	Evaluation	Languages
P18-1073	same as artetxe2018generalizing			
luong2015bilingual	Europarl[13]	N/A	CLDC[14]	EN-DE
hartmann2018why	Wikipedia[15]	N/A	lexicon induction	EN-EN
conneau2017word	word embeddings (Wiki)	N/A	lexicon induction	EN⇔ES,EN⇔FR,EN⇔DE, EN⇔RU,EN⇔ZH,EN⇔EO
	word embeddings (Wiki;WaCky)	N/A	lexicon induction	EN⇔IT
	word embeddings (Smith et al. (2017))	N/A	sentence translation retrieval	EN⇔IT
	word embeddings	N/A	cross-lingual word similarity	EN-ES,EN-DE,EN-IT
	word embeddings	N/A	word-by-word transaltion	EN-Esperanto
sogaard2015inverted	Wikipedia dumps[16]	article concept	Document classification +POS tagging +Dependency parsing +Word alignment	GE, EN, FR, Sp, SW
vulic2015bilingual	theme-aligned Wikipedia data	document theme	BLI	SP-EN,IT-EN,DU-EN

[13] parallel Europarl v7 corpus
[14] Cross-lingual Document Classification
[15] Wikipedia dump from March 2018. https://dumps.wikimedia.org/enwiki/
[16] https://sites.google.com/site/rmyeid/ projects/polyglot

Chapter 3

Generation and Processing of Word Co-occurrence Networks Using corpus2graph

For monolingual word embedding learning, I aim to combine strengths from both prediction-based methods and count-based methods. Precisely, I want to inject information extracted and calculated based on the statistics of word-word occurrences (count-based) on the training corpus into the skip-gram negative sampling architecture of the word2vec model (prediction-based). In this chapter, I focus on the count-based side, i.e. generation and processing of the information extracted from the word-word co-occurrence statistics. This chapter builds on (Zhang et al., 2018).

3.1 Word co-occurrence network and corpus2graph

3.1.1 Word-word co-occurrence matrix and word co-occurrence network

As introduced in Section 1.3, many count-based methods start with the word-word co-occurrence matrix (Levy and Goldberg, 2014b; Pennington et al., 2014; Salle et al., 2016; Xin et al., 2018). A typical way to use this word-word co-occurrence statistics is to reweight the matrix (e.g. by using PPMI) and then reduce its dimensions (e.g. by applying SVD) to get vector representations of words (i.e. rows in the dimension reduced matrix).

Because matrix and network are interchangeable, besides the word-word co-occurrence matrix, the same count-based statistical information can be represented in the network format,

in the so-called *word co-occurrence network*. Based on this network, graph algorithms such as random walks can be applied to it to process the count-based statistics in another way.

A word co-occurrence network is a graph of word interactions representing the co-occurrence of words in a corpus. An edge can be created when two words co-occur within a sentence; these words are possibly non-adjacent, with a maximum distance (in number of words, see Section 3.2.2) defined by a parameter d_{max} (Cancho and Solé, 2001). In an alternate definition, an edge can be created when two words co-occur in a fixed-sized sliding window moving along the entire document or sentences (Rousseau and Vazirgiannis, 2013). Despite different methods of forming edges, the structure of the network for sentences will be the same for the two above definitions if the maximum distance of the former is equal to the sliding window size of the latter. Edges can be weighted or not. An edge's weight indicates the strength of the connection between two words, which is often related to their number of co-occurrences and/or their distance in the text. Edges can be directed or undirected (Mihalcea and Radev, 2011).

Word co-occurrence networks are widely used in graph-based natural language processing methods and applications, such as keyword extraction (Mihalcea and Tarau, 2004) and word sense discrimination (Ferret, 2004).

3.1.2 corpus2graph

While there already exist network analysis packages such as NetworkX (Hagberg et al., 2008), igraph (Csardi and Nepusz, 2006) and graph-tool (Peixoto, 2014), they do not include components to make them applicable to texts directly: users have to provide their own word preprocessor, sentence analyzer, weight function. Moreover, for certain graph-based NLP applications, it is not straightforward to find the best network configurations, e.g. the maximum distance between words. A huge number of experiments with different network configurations is inevitable, typically rebuilding the network from scratch for each new configuration. It is easy to build a word co-occurrence network from texts by using tools like textexture[1] or GoWvis[2]. But they mainly focus on network visualization and cannot handle large corpora such as the English Wikipedia.

Our contributions: To address these inconveniences of generating a word co-occurrence network from a large corpus for NLP applications, I propose corpus2graph, an open-source[3] NLP-application-oriented Python package that generates a word co-occurrence network from a large corpus. It not only contains different built-in methods to preprocess words,

[1] http://textexture.com
[2] https://safetyapp.shinyapps.io/GoWvis/
[3] available at https://github.com/zzcoolj/corpus2graph

analyze sentences, extract word pairs and define edge weights, but also supports user-customized functions. By using parallelization techniques, it can generate a large word co-occurrence network of the whole English Wikipedia data within hours. And thanks to its nodes-edges-weight three-level progressive calculation design, rebuilding networks with different configurations is even faster as it does not need to start all over again. This tool also works with other graph libraries such as igraph, NetworkX and graph-tool as a front end providing data to boost network generation speed. This work was done with the help of Ruiqing Yin, another PhD student at LIMSI. I designed the principles of the algorithms and the prototype architecture then implemented the system. Ruiqing contributed to the implementation and code optimization.

3.2 Efficient NLP-oriented graph generation

Our tool builds a word co-occurrence network given a source corpus and a maximal distance d_{max}. It contains three major parts: word processing, sentence analysis and word pair analysis from an NLP point of view. They correspond to three different stages in network construction.

3.2.1 Node level: word preprocessing

The contents of large corpora such as the whole English Wikipedia are often stored in thousands of files, where each file may contain several Wikipedia articles. To process a corpus, I consider a file as the minimal processing unit. I go through all the files in a multiprocessing way: Files are equally distributed to all processors and each processor handles one file at a time.

To reduce space requirements, I encode each file by replacing words with numeric ids. Besides, to enable independent, parallel processing of each file, these numeric ids are local to each process, hence to each file. A local-id-encoded file and its corresponding local dictionary (word $\rightarrow$ local id) are created after this process. As this process focuses on words, our tool provides several word preprocessing options such as tokenizer selection, stemmer selection, removing numbers and removing punctuation marks. It also supports user-provided word preprocessing functions.

Given sentence “The history of natural language processing generally started in the 1950s.” as an example. After tokenization, stemming, replacing number to zero and removing punctuation marks and stop words, we got the pre-processed sentence shown in Figure 3.1. Then each unique token has been set to a local-id.

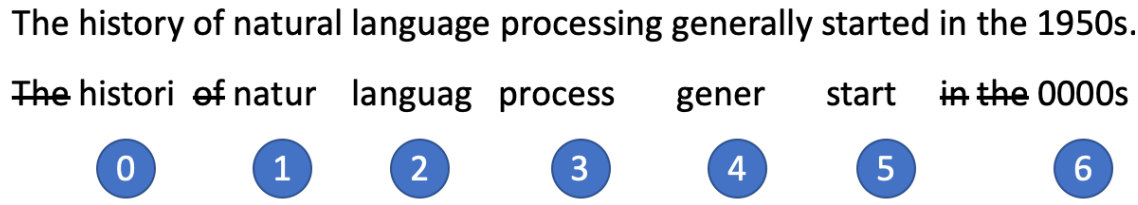


Fig. 3.1 Illustration of word preprocessing.

All these local-id-encoded files and corresponding local dictionaries are stored in a specific folder (`dicts_and_encoded_texts`). Once all source files are processed, a global dictionary (word → global id) is created by merging all local dictionaries. Note that at this point, files are still encoded with local word ids.

3.2.2 Node co-occurrences: sentence analysis

To prepare the construction of network edges, this step aims to enumerate word co-occurrences, taking into account word distance. Given two words w_1^i and w_2^j that co-occur within a sentence at positions i and j ($i, j \in \{1 \dots l\}$ where l is the number of words in the sentence), I define their distance $d(w_1^i, w_2^j) = j - i$. For each input file, d_{max} output files will be created to enumerate co-occurrences: one for each distance $\delta \in \{1, \dots d_{max}\}$. They are stored in the `cooc` folder.

To prepare the aggregation of individual statistics into global statistics (see Section 3.2.3), each process converts local word ids into global word ids through the combination of its local dictionary and of the global dictionary. Note that at this point the global dictionary must be loaded into RAM.

Following the previous example, sentence "The history of natural language processing generally started in the 1950s." has been preprocessed and encoded using local-id as "0 1 2 3 4 5 6". Before sentence analysis, it is further encoded using global-id as "5 0 9 2 4 3 7" as shown in Figure 3.2.

Then, a sentence analyzer goes through this file sentence by sentence to extract all word co-occurrences with distances $\delta \leq d_{max}$. The sentence analyzer will extract word pairs from distance 1 to d_{max}. The results for $d_{max} = 5$ are shown in Table 3.1.

Note that edges generated by my default sentence analyzer are directed. And the direction is defined from the left word to the right word as yellow and blue edges shown in the illustration figure. It means that I distinguish edge (a, b) and edge (b, a) as two different edges. And even in aggregation stage, they are counted respectively. More details will follow in Section 3.2.3.

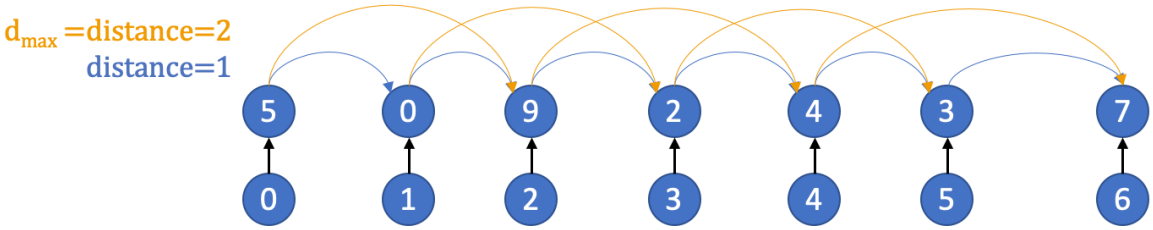


Fig. 3.2 Illustration of sentence analysis.

User-customized sentence analyzer and distance computation are also supported so that more sophisticated definitions of word pair distance can be introduced. For instance, I plan to provide a syntactic distance: the sentence analyzer will build a parse tree for each sentence and compute word pair distance as their distance in the parse tree.

Besides, in this step, I also provide an option to count the number of occurrences of each word $occ(w)$. Given that a large corpus like Wikipedia has a huge number of tokens, a global word count is convenient to enable the user to select words based on a frequency threshold before network generation. I return to this point in the next subsection.

3.2.3 Edge attribute level: word pair analysis

A word pair (w_1, w_2) is represented by an edge linking two nodes in the word co-occurrence network. In this step, I enrich edge information with *direction* and *weight* by word pair analysis.

Let $cooc(\delta, w_1, w_2)$ the number of co-occurrences of w_1 and w_2 with a distance of δ (Eq. 3.1). I define the weight $w(d_{max}, w_1, w_2)$ of an edge (w_1, w_2) as the total number of co-occurrences of w_1 and w_2 with distances $\delta \leq d_{max}$ (Eq. 3.2).

$$cooc(\delta, w_1, w_2) = |\{(w_1^i, w_2^j); d(w_1^i, w_2^j) = \delta\}| \tag{3.1}$$

$$w(d_{max}, w_1, w_2) = \sum_{\delta \leq d_{max}} cooc(\delta, w_1, w_2) \tag{3.2}$$

δ	**Word Pairs**
2	(5, 0), (0, 9), (9, 2), (2, 4), (4, 3), (3, 7)
3	(5, 9), (0, 2), (9, 4), (2, 3), (4, 7)
4	(5, 2), (0, 4), (9, 3), (2, 7)
5	(5, 4), (0, 3), (9, 7)

Table 3.1 Word pairs for different values of distance δ in sentence "5 0 9 2 4 3 7"

For efficiency I use an iterative algorithm (Eq. 3.3):

$$w(d,w_1,w_2) = \begin{cases} cooc(1,w_1,w_2), & \text{if } d = 1 \\ cooc(d,w_1,w_2) + & \\ w(d-1,w_1,w_2), & \text{otherwise} \end{cases} \tag{3.3}$$

I calculate the edge weight of different window sizes in a stepwise fashion by applying Eq. 3.3. For the initial calculation, I start by counting and merging all word pair files of distance 1 in the `edges` folder generated by step 2 to get a co-occurrence count file. This file contains information on all distinct word pair co-occurrence counts for distance 1. I then follow the same principle to obtain a co-occurrence count file for distance 2. I merge this result with the previous one to get word pair co-occurrence counts for window size 2. I continue this way until distance d_{max}.

If I wish to make further experiments with a larger distance, there is no need to recompute counts from the very beginning: I just need to pick up the word pair co-occurrences of the largest distance that I already calculated and start from there. All co-occurrence count files for the different distances are stored in the `graph` folder.

Defining the weight as the sum of co-occurrences of two words with different distances is just one of the most common ways used in graph-based natural language processing applications. I also support other (built-in and user-defined) definitions of the weight. For instance, when calculating the sum of co-occurrences, I can assign different weights to co-occurrences according to the word pair distance, to make the resulting edge weight more sensitive to the word pair distance information.

- *weight* = 1×*number of* + 1×*number of*
- *undirected*

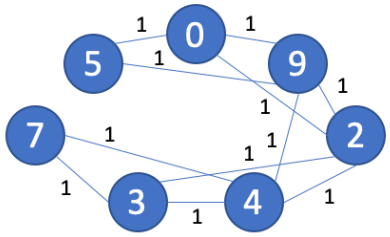


Fig. 3.3 Illustration of word pair.

Figure 3.3 shows the weight based on the example given in Figure 3.2 under the condition that I set the same weight (1) to edges of different distances. I also set the graph to be undirected. It is tricky here as there is no word pair in opposite direction in my example. So the weight does not change either in directed graph or undirected graph. In a more general

case, I will sum the count of edges in both directions with same node pair before generating the network.

For a large corpus, I may not need all edges to generate the final network. Based on the word count information from Section 3.2.2, I may select those nodes whose total frequency is greater than or equal to *min_count*, or the most frequent *vocab_size* number of nodes, or apply both of these constraints, before building edges and computing their weights.

3.3 Efficient graph processing

3.3.1 Matrix-type representations

Although our tool works with graph libraries like igraph, NetworkX and graph-tool as a front end, we also provide our own version of graph processing class for efficiency reasons: Most graph libraries treat graph processing problems in a network way. Their algorithms are mainly based on network concepts such as node, edge, weight, degree. Sometimes, using these concepts directly in network algorithms is intuitive but not computationally efficient. As networks and matrices are interchangeable, our graph processing class uses matrix-type representations and tries to adapt network algorithms in a matrix calculation fashion, which boosts up the calculation speed.

In our matrix representation for graph information, nodes, edges and weights are stored in an adjacency matrix A: a square matrix of dimension $|N| \times |N|$, where N is the number of nodes in the graph. Each row of this matrix stands for a starting node, each column represents one ending node and each cell contains the weight of the edge from that starting node to the ending node.

Note that not all network algorithms are suitable for adapting into a matrix version. For this reason, our graph processing class does not aim to be a replacement of the graph libraries I mentioned before. It is just a supplement, which provides matrix-based calculation versions for some of the algorithms.

To give the reader an intuition about the difference between the common network-type representation and the matrix-type representation, the coming subsection uses the random walk algorithm as an example.

3.3.2 Random walk

Random walks (Aldous and Fill, 2002) are widely used in graph-based natural language processing tasks, for instance word-sense disambiguation (Moro et al., 2014) and text

summarization (Erkan and Radev, 2004; Zhu et al., 2007). The core of the random walk related algorithms calculation is the transition matrix P.

In the random walk scenario, starting from an initial vertex u, we cross an edge attached to u that leads to another vertex, say v (v can be u itself when there exists an edge that leads from u to u, which we call a self-loop). Element P_{uv} of the transition matrix P represents the transition probability $P(u,v)$ of the walk from vertex u to vertex v in one step. For a weighted directed network, $P(u,v)$ can be calculated as the ratio of the weight of the edge (u,v) over the sum of the weights of the edges that start from vertex u.

NetworkX (version 2.0) provides a built-in method *stochastic_graph* to calculate the transition matrix P. For directed graphs, it starts by calculating the sum of the adjacent edge weights of each node in the graph and stores all the results in memory for future usage. Then it traverses every edge (u,v), dividing its weight by the sum of the weights of the edges that start from u.

Based on the adjacency matrix A introduced in Section 3.3.1, the transition probability $P(u,v)$ can be expressed as:

$$P(u,v) = A_{uv} / \sum_{i=1}^{|A_u|} A_{ui}$$

The transition matrix P can be easily calculated in two steps: First, getting sums of all elements along each row and broadcasting the results against the input matrix to preserve the dimensions (*keepdims* is set to True); Second, performing element-wise division to get the ratios of each cell value to the sum of all its row's cell values. By using NumPy (Walt et al., 2011), the calculation is more efficient both in speed and memory. Besides, as the calculations are independent of each row, we can take advantage of multiprocessing to further enhance the computing speed.

3.4 Experiments

3.4.1 Set-up

In the first experiment, I generated a word co-occurrence network for a small corpus of 7416 tokens (one file of the English Wikipedia dump from April 2017) without using multiprocessing on a computer equipped with the Intel Core i7-6700HQ processor. Our tool serves as a front end to provide nodes and edges to the graph libraries NetworkX, igraph and graph-tool. In contrast, the baseline method processes the corpus sentence by sentence, extracting word pairs with a distance $\delta \leq d_{max}$ and adding them to the graph as edges (or

updating the weight of edges) through these libraries. All distinct tokens in this corpus are considered as nodes.

In the second experiment, I used our tool to extract nodes and edges for the generation of a word co-occurrence network on the entire English Wikipedia dump from April 2017 using 50 logical cores on a server with 4 Intel Xeon E5-4620 processors , $d_{max} = 5$, $min\_count = 5$ and $vocab\_size = 10000$.

In the last experiment, I compared the random walk transition matrix calculation speed on the word co-occurrence network built from the previous experiment result between my method and the built-in method of NetworkX (version 2.0) on a computer equipped with Intel Core i7-6700HQ processor.

3.4.2 Results

	NetworkX	igraph	graph-tool
baseline	**4.88**	8727.49	77.70
corpus2graph	15.90	**14.47**	**14.31**

Table 3.2 Word network generation speed (seconds)

Table 3.2 shows that regardless of the library used to receive graph information generated by corpus2graph, it takes around 15 seconds from the small Wikipedia corpora to the final word co-occurrence network. And my method performs much better than the baseline method with igraph and graph-tool even without using multiprocessing. I found that in general loading all edges and nodes information at once is faster than loading edge and node information one by one and it takes approximately the same time for all graph libraries. As for NetworkX, the baseline method is faster. But as the corpora get larger, the baseline model uses more and more memory to store the continuously growing graph, and the processing time increases too.

For the second experiment, our tool took around 236 seconds for node processing (Section 3.2.1), 2501 seconds for node co-occurrence analysis (Section 3.2.2) and 8004 seconds for edge information enriching (Section 3.2.3). In total, it took less than 3 hours to obtain all the nodes and weighted edges for the subsequent network generation.

Generation of:	network	transition matrix
NetworkX	447.71	2533.88
corpus2graph	**116.15**	**1.06**

Table 3.3 Transition matrix calculation speed (seconds)

Table 3.3 shows the results of the third experiment. Loading network information into our graph processing class is faster than loading into the graph class of NetworkX. Moreover, our random walk transition matrix calculation method is 2390 times faster than the built-in method in NetworkX.

3.5 Discussion

3.5.1 Difference between word co-occurrence network and target-context word relation in word embeddings training

While if we set d_{max} to the same value as window size in word embeddings training, the undirected word co-occurrence network generated by corpus2graph can represent the statistics of target-context word relations in word embeddings learning (e.g. in word2vec). It is not exactly the same because of the following three reasons:

- The weight of edges in an undirected word co-occurrence network represents the times of two words co-occurring within a defined distance. While in word2vec, the number of target-context word relations of two particular words should be twice as the weight in the unidirected word co-occurrence network of same word pairs. Because during the word2vec training, the context word A of the target word B will become the target word when word B is the context word. So this word pair relation has been counted twice under the undirected case;

- Self-loop. In word embeddings training stage, the context word of any target word can never be the target word itself. But in word co-occurrence network, this situation exists. For instance, in sentence “context word and target word are two important concept in word2vec training.” “word” has a self loop when maximum distance is bigger than 2.

- Word embedding training methods normally train over the corpora several times. So the times of target-context word relations is proportional to the number of iterations of training.

Therefore, the stochastic matrix is more appropriate to graph-based word embeddings learning. I will talk in more detail in Section 4.2.1.

3.5.2 Three multiprocessing...

As shown in Figure 3.4, three stages (word processing, sentence analysis and word pair analysis) of corpus2graph match three multiprocessing in the data-flow diagram. The key to making this tool efficient is maximizing the use of multiprocessing, which means:

- Use as many processors as you can (just to make sure the memory is safe);
- Avoid using large training corpus files directly, try to split them into smaller files (around 200 sentences per file is a suggested setup) before training.

3.6 Conclusion

I presented an NLP-application-oriented Python package that generates a word co-occurrence network from a large corpus. Experiments show that our tool can boost network generation and graph processing speed compared to baselines.

Possible extensions of this work would be to support more graph processing methods and to connect our tool to more existing graph libraries.

In this chapter, I focused on the usage of the count-based information. Corpus2graph enables us to generate a word co-occurrence from a Wikipedia-size corpus and to apply graph algorithms to it. In the next chapter, I show how to inject the information obtained from corpus2graph into the skip-gram negative sampling architecture to improve monolingual word embedding learning.

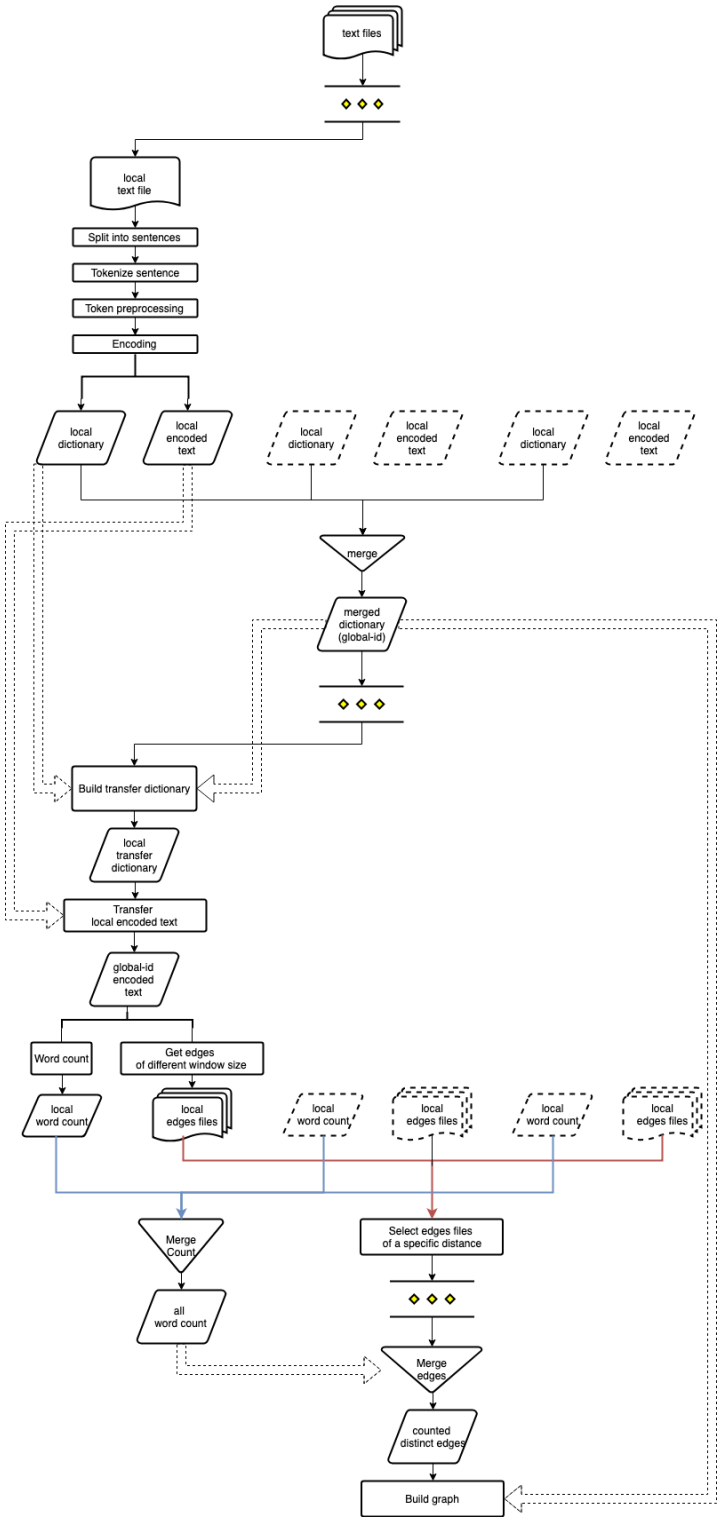


Fig. 3.4 Data-flow diagram of corpus2graph

Chapter 4

GNEG: Graph-Based Negative Sampling for word2vec

As introduced in Chapter 3, I plan to inject the information extracted from a word co-occurrence network into the skip-gram negative sampling architecture of word2vec to improve word embedding learning. Precisely, I hypothesize that taking into account global, corpus-level information and generating a different noise distribution in negative sampling for each training word better satisfies the requirements of negative examples for each training word than the original frequency-based distribution.

As I have a tool, corpus2graph, to pre-compute word co-occurrence statistics from the corpus and apply to it network algorithms such as random walk, in this chapter, I focus on the analysis and the improvement of the word2vec model. This chapter builds on (Zhang and Zweigenbaum, 2018).

4.1 Negative Sampling

Negative sampling, as introduced by Mikolov et al. (2013e), is used as a standard component in both the CBOW and skip-gram models of word2vec. For practical reasons, instead of using a softmax function, earlier work explored different alternatives which approximate the softmax in a computationally efficient way. These alternative methods can be roughly divided into two categories: softmax-based approaches (hierarchical softmax (Morin and Bengio, 2005), differentiated softmax (Chen et al., 2015) and CNN-softmax (Kim et al., 2016)) and sampling-based approaches (importance sampling (Bengio et al., 2003b), target sampling (Jean et al., 2014), noise contrastive estimation (Mnih and Teh, 2012) and negative

sampling (Mikolov et al., 2013e)). Generally speaking, among all these methods, negative sampling is the best choice for distributed word representation learning (Ruder, 2016).

Negative sampling replaces the softmax with binary classifiers (see Section 1.2.2.3). For instance, in the skip-gram model, word representations are learned by predicting a training word's surrounding words (positive target words) given this training word. When training, correct surrounding words provide positive examples in contrast to a set of sampled negative examples (noise). To provide these negative examples, a noise distribution is empirically defined as the unigram distribution of the words to the $3/4^{th}$ power:

$$P_n(w) = U(w)^{\frac{3}{4}} / \sum_{i=1}^{|vocab|} U(w_i)^{\frac{3}{4}} \tag{4.1}$$

Although this noise distribution is widely used and significantly improves the distributed word representation quality, I believe there is still room for improvement in the two following aspects: First, the unigram distribution only takes into account word frequency, and provides the same noise distribution when selecting negative examples for different target words. Labeau and Allauzen (2017) already showed that a context-dependent noise distribution could be a better solution to learn a language model. But they only use information on adjacent words. Second, unlike the positive target words, the meaning of negative examples remain unclear: For a training word, we do not know what a good noise distribution should be, while we do know what a good target word is (one of its surrounding words).

My contributions: To address these two problems, I propose a new graph-based method to calculate noise distribution for negative sampling. Based on a word co-occurrence network, my noise distribution is targeted to the (positive) target words, i.e. the context words of each training word. Besides, through our empirical exploration of the noise distribution, I get a better understanding of the meaning of 'negative' and of the characteristics of good noise distributions.

The rest of the chapter is organized as follows: Section 4.2 defines the word co-occurrence network concepts and introduces my graph-based negative sampling approach. Section 4.3 shows the experimental settings and results, then discusses my understanding of the good noise distributions. Finally, Section 4.5 draws conclusions and mentions future work directions.

4.2 Graph-based Negative Sampling

I begin with the word co-occurrence network generation (Section 4.2.1). By comparing it with the word2vec models, I show the relation between the stochastic matrix of the word

co-occurrence network and the distribution of the training word contexts (positive examples) in word2vec. I introduce three methods to generate noise distributions for negative sampling based on the word co-occurrence network:

- Directly using the word-context distribution extracted from the word co-occurrence network (Section 4.2.2)
- Calculating the difference between the original unigram distribution and the word-context distribution (Section 4.2.3)
- Performing t-step random walks on the word co-occurrence network (Section 4.2.4)

I finally insert my noise distribution into the word2vec negative sampling training (Sec. 4.2.5).

4.2.1 Word Co-occurrence Network and Stochastic Matrix

I have introduced the definition of the word co-occurrence network and discussed the relation between the word-word co-occurrence matrix and the word co-occurrence network in Section 3.1.1. In this section, I recall the word co-occurrence network definition and show its relation with the word-word co-occurrence matrix in a detailed mathematical way.

A word co-occurrence network is a graph of word interactions representing the co-occurrence of words in a corpus. An undirected edge can be created when two words co-occur within a sentence; these words are possibly non-adjacent, with a maximum distance defined by a parameter d_{max} (Cancho and Solé, 2001). Given two words w_u^i and w_v^j that co-occur within a sentence at token positions i and j ($i, j \in \{1 \dots l\}$), I define the distance $d(w_u^i, w_v^j) = |j - i|$ and the co-occurrence of w_u and w_v at a distance δ as $cooc(\delta, w_u, w_v) = \left| \left\{ \left(w_u^i, w_v^j \right) \middle| d\left(w_u^i, w_v^j \right) = \delta \right\} \right|$. I define the weight $w(d_{max}, w_u, w_v)$ of an edge (w_u, w_v) as the total number of co-occurrences of w_u and w_v with distances $\delta \leq d_{max}$:
$w(d_{max}, w_u, w_v) = \sum_{\delta=1}^{d_{max}} cooc(\delta, w_u, w_v)$.

An undirected weighted word co-occurrence network can also be represented as a symmetric adjacency matrix (Mihalcea and Radev, 2011), a square matrix A of dimension $|W| \times |W|$. In my case, W is the set of words used to generate the word co-occurrence network, and the matrix elements A_{uv} and A_{vu} are the edge weight $w(d_{max}, w_u, w_v)$. Then each row of the adjacency matrix A can be normalized (i.e., so that each row sums to 1), turning it into a right stochastic matrix S.

Negative sampling, in the skip-gram model, uniformly draws at random for each training word one of its surrounding words as the (positive) target word. This range is determined by

the size of the training context c. In other words, a surrounding word w_s of the training word w_t must satisfy the following condition: $d(w_t^i, w_s^j) \leq c$.

For the same corpus, let us set d_{max} equal to c in word2vec and generate a word co-occurrence network of the whole corpus. Then element S_{uv} in the adjacency matrix extracted from the network represents the probability that word w_v be selected as the context word for the selected center word w_u ($P_{bigram}(w_u, w_v)$ in Eq. 4.2). Row S_u thus shows the distribution of context words for the center word w_u after training the whole corpus. Note that no matter how many training iterations are done over the corpus, this distribution will not change.

$$P_{bigram}(w_u, w_v) = \frac{\sum_{\delta=1}^{d_{max}} cooc(\delta, w_u, w_v)}{\sum_{i=1}^{|vocab|} \sum_{\delta=1}^{d_{max}} cooc(\delta, w_u, w_i)} = S_{uv} \tag{4.2}$$

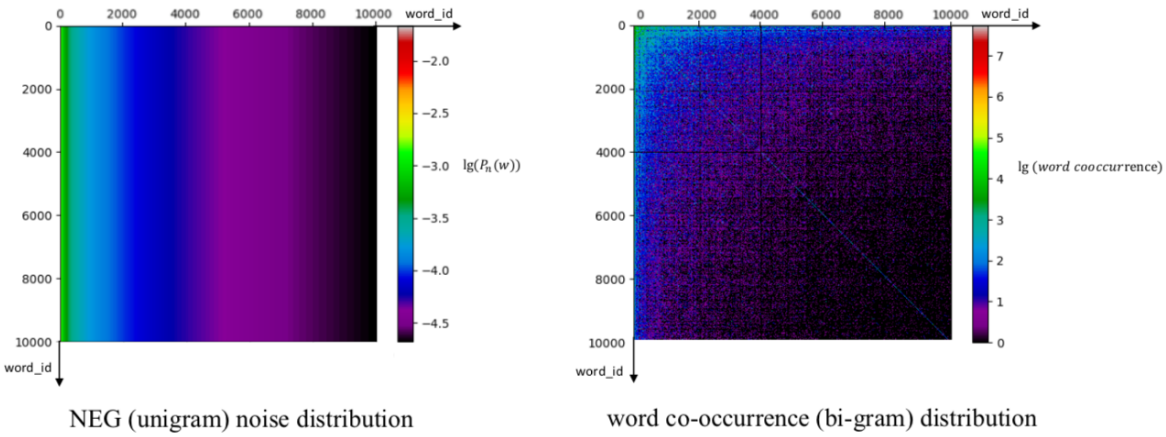
Networks and matrices are interchangeable. The adjacency matrix of the word co-occurrence network can also be seen as the matrix of word-word co-occurrence counts calculated in a statistical way as in GloVe (Pennington et al., 2014). But unlike Glove, where the matrix is used for factorization, I use word co-occurrence statistics to generate a network, then use network algorithms.

4.2.2 (Positive) Target Word Context Distribution

As discussed in Section 4.2.1, the stochastic matrix S of the word co-occurrence network represents the context distribution of the center words. Here, I use the context distribution of the (positive) target word, i.e. the context word of the training word, as one of the three types of bases for noise distribution matrix calculation.

The idea behind this is to perform nonlinear logistic regression to differentiate the observed data from some artificially generated noise. This idea was introduced by Gutmann and Hyvärinen (2010) with the name Noise Contrastive Estimation (NCE). Negative sampling (NEG) can be considered as a simplified version of NCE that follows the same idea and uses the unigram distribution as the basis of the noise distribution. I attempt to improve this by replacing the unigram distribution with a bigram distribution (word co-occurrence, not necessarily contiguous) to make the noise distribution targeted to the positive target word of the training word.

Compared to the word-frequency-based unigram distribution (see left graph in Figure 4.1), the word co-occurrence based bigram distribution (see right graph in Figure 4.1) is sparser. With the unigram distribution, for any training word, all the other vocabulary words can be



NEG (unigram) noise distribution word co-occurrence (bi-gram) distribution

Fig. 4.1 Difference between word-frequency-based unigram distribution and word co-occurrence bi-gram distribution on the same corpus.

selected as noise words because of their non-zero frequency. In contrast, with the bigram distribution, some vocabulary words may never co-occur with a given (positive) target word, which makes them impossible to be selected as the negative examples for the training word. To check the influence of this zero co-occurrence case, I also provide a modified stochastic matrix S' smoothed by replacing all zeros in matrix S with the minimum non-zero value of their corresponding rows.

4.2.3 Difference Between the Unigram Distribution and the (Positive) Target Words Contexts Distribution

Starting from the 'white noise' unigram distribution, for each (positive) target word of the training word w_u, we subtract from it the corresponding context distribution of this target word. Elements in this new basis matrix $S_{difference\,u,v}$ of noise distribution are:

$$P_{difference}(w_u, w_v) = P_n(w_u) - S_{uv} \tag{4.3}$$

where w_v is one of the negative examples when the (positive) target word is w_u, P_n is the unigram distribution defined in Eq. 4.1 and S is the stochastic matrix I used in Section 4.2.2. For zeros and negative values in this matrix, I reset them to the minimum non-zero value of the corresponding row $P_{difference}(w_u)$.

4.2.4 Random Walks on the Word Co-occurrence Network

After generating the word co-occurrence network, I apply random walks (Aldous and Fill, 2002) to it to obtain yet another noise distribution matrix for negative sampling.

Let us define random walks on the co-occurrence network: Starting from an initial vertex w_u , at each step we can cross an edge attached to w_u that leads to another vertex, say w_v. For a weighted word co-occurrence network, we define the transition probability $P(w_u, w_v)$ from vertex w_u to vertex w_v as the ratio of the weight of the edge (w_u, w_v) over the sum of weights on all adjacency edges of vertex w_u. Using the adjacency matrix A and the right stochastic matrix S presented in Section 4.2.1, $P_{u,v}$ can be calculated by: $P(w_u, w_v) = A_{uv} / \sum_{i=1}^{|A_u|} A_{ui} = S_{uv}$.

As I want to learn transition probabilities for all (positive) target words, I apply random walks on all vertices by making each (positive) target word an initial vertex of one t-step random walk at the same time.

The whole set of transition probabilities can be represented as a transition matrix, which is exactly the right stochastic matrix S of the word co-occurrence network in my case. I found that the self-loops (edges that start and end at the same vertex: the main diagonal of an adjacency matrix or a stochastic matrix) in matrix S represent the occurrence of a word in its own context, which may happen in repetitions. I hypothesize they constitute spurious events and therefore test the t-step random walk both on matrix S and its smoothed version R' in which the self-loops are removed. To see the effect of the self-loops, I perform the t-step random walk on both matrices S and R'.

Based on that, the elements of the t-step random walk transition matrix can be calculated by:

$$P_{\textit{random-walk}}(w_u, w_v) = S^t_{uv} \; or \; (R')^t_{uv} \tag{4.4}$$

Cancho and Solé (2001) showed that a word co-occurrence network is highly connected. For such networks, random walks converge to a steady state in just a few steps. Steady state means that no matter which vertex one starts with, the distribution of the destination vertex probabilities remains the same. In other words, all S^t columns will have the same value. So I set the maximum step number t_{max} to 4. I will use these t-step random walk transition matrices as the basis for one of my noise distributions matrices for negative sampling.

4.2.5 Noise Distribution Matrix

Starting from the basic noise distribution matrix, I use the power function to adjust the distribution. Then I normalize all rows of this adjusted matrix to let each row sum to 1.

	WordSim-353		SimLex-999		Word Analogy		
	Pearson	Spearman	Pearson	Spearman	Semantic	Syntactic	Total
baseline word2vec	66.12%	69.60%	37.36%	36.33%	73.00%	70.67%	71.37%
bigram distr. (Eq. 4.2)	66.10%	69.77%	**38.05%**	**37.18%**	77.36%†	75.55%†	76.09%†
difference distr. (Eq. 4.3)	**67.71%**†	**71.51%**†	37.65%	36.58%	77.14%†	**75.98%**†	**76.33%**†
random walk (Eq. 4.4)	66.94%†	70.70%†	37.73%	36.74%	**77.75%**†	74.86%†	75.73%†

Table 4.1 Evaluation results on WordSim-353, SimLex-999 and the word analogy task for the plain word2vec model and my three graph-based noise distributions on the entire English Wikipedia dump. A dagger† marks a statistically significant difference to the baseline word2vec.

distribution	d_{max}	p	others
bigram	3	0.25	*replace_zeros=T*
difference	3	0.01	
random walk	5	0.25	$t = 2$, *no_self_loops=T*

Table 4.2 Best parameters settings for Graph-based negative sampling

Finally, we get:

$$P_n(w_u, w_v) = (B_{uv})^p / \sum_{i=1}^{|B_u|} (B_{ui})^p \tag{4.5}$$

where B is the basic noise distribution calculated according to Eq. 4.2, 4.3 or 4.4 and p is the power.

When performing word2vec training with negative sampling, for each (positive) target word (context word) of the training word, I use the corresponding row in my noise distribution matrix to replace the original unigram noise distribution for the selection of noise candidates.

4.3 Experiments and Results

4.3.1 Set-up and Evaluation Methods

I use the skip-gram negative sampling model with window size 5, vocabulary size 10000, vector dimension size 200, number of iterations 5 and negative examples 5 to compute baseline word embeddings. My three types of graph-based skip-gram negative sampling models share the parameters of the baseline. In addition to these common parameters, they have their own parameters: the maximum distance d_{max} for co-occurrence networks generation, a Boolean *replace_zeros* to control whether or not to replace zeros with the

minimum non-zero values, a Boolean *no_self_loops* to control whether or not to remove the self-loops, the number of random walk steps t (Eq. 4.4) and the power p (Eq. 4.5).

All four models are trained on an English Wikipedia dump from April 2017 of three sizes: about 19M tokens, about 94M tokens (both are for detailed analyses and non-common parameters grid search in each of the three graph-based models) and around 2.19 billion tokens (for four models comparison). During corpus preprocessing, I use CoreNLP (Manning et al., 2014) for sentence segmentation and word tokenization, then convert tokens to lowercase, replace all expressions with numbers by 0 and replace rare tokens with *UNK*s.

I perform a grid search on the ∼19M tokens corpus, with $d_{max} \in \{2,\dots 10\}$, $t \in \{1,\dots 4\}$, $p \in \{-2,-1,0.01,0.25,0.75,1,2\}$ and $True, False$ for the two Boolean parameters. I retain the best parameters obtained by this grid search and perform a tighter grid search around them on the ∼94M tokens corpus. Then based on the two grid search results, I select the final parameters for the entire Wikipedia dump test. I evaluate the resulting word embeddings on word similarity tasks using WordSim-353 (Finkelstein et al., 2001) and SimLex-999 (Hill et al., 2014) (correlation with humans), and on the word analogy task of Mikolov et al. (2013b) (% correct). Therefore, I use the correlation coefficients between model similarity judgments and human similarity judgments for WordSim-353 and SimLex-999 tasks and the accuracy of the model prediction with gold standard for the word analogy task (the metrics in Table 4.1) as objective functions for these parameter tuning processes.

4.3.2 Results

The best grid search parameters are shown in Table 4.2, the final evaluation results on the entire English Wikipedia in Table 4.1. The results show that graph-based negative sampling boosts the word analogy task by about 5% and improves word similarity by about 1%.

As vocabulary size is set to 10000, not all data in evaluation datasets is used. I report here the sizes of the datasets and of the subsets that contained no unknown word, that I used for evaluation: WordSim-353: 353;261; SImLex-999: 999;679; Word Analogy: 19544;6032. I also computed the statistical significance of the differences between my models and the baseline model. Both word similarity tasks use correlation coefficients, so I computed Steiger's Z tests (Steiger, 1980) between the correlation coefficients of each of my models (bigram distribution, difference distribution and random walk distribution) versus the word2vec skip-gram negative sampling baseline. For WordSim-353, differences are significant (*2-tailed* $p < 0.05$) for difference distribution and random walk distribution for both Pearson and Spearman correlation coefficients; differences are not significant for bigram distribution. For SimLex-999, no difference is significant (all *2-tailed* $p > 0.05$). The word analogy task uses accuracy, I tested statistical significance of the differences by

approximate randomization (Yeh, 2000). Based on 10000 shuffles, I confirmed that all differences between the accuracies of my models and the accuracy of word2vec skip-gram are statistically significant ($p < 0.0001$).

The time complexity when using my modified negative sampling distribution is similar to that of the original skip-gram negative sampling except that the distribution from which negative examples are sampled is different for each token. I pre-compute this distribution off-line for each token so that the added complexity is proportional to the size of the vocabulary. Specifically, pre-computing the co-occurrences and graphs using corpus2graph (Zhang et al., 2018) takes about 2.5 hours on top of 8 hours for word2vec alone on the entire Wikipedia corpus using 50 logical cores on a server with 4 Intel Xeon E5-4620 processors : the extra cost is not excessive.

4.3.3 Discussion

Let us take a closer look at each graph-based model. First, the (positive) target word context distribution based model: I find that all else being equal, replacing zero values gives better performance. I believe a reason may be that for a training word, all the other words should have a probability to be selected as negative examples—the job of noise distributions is to assign these probabilities. I note that for SimLex-999, all combinations of parameters in my grid search outperform the baseline. But unfortunately the differences are not significant.

Second, the difference model: the word analogy task results show a strong dependency on power p: the lower the power p, the higher the performance.

Third, the random-walk model: I observe that all top 5 combinations of parameters in the grid search do random walks after removing self-loops.

4.4 The implementation of word2vec

The source code of word2vec was published by Mikolov et al. (2013a) in their first paper introducing this model. It was written in C++ and both CBOW and skip-gram architectures are supported. Word2vec has many versions of implementations in different programming languages (e.g. Python version in gensim (Řehůřek and Sojka, 2010), Java version in deeplearning4j (Eclipse-Deeplearning4j-Development-Team, 2014), Spark version in Spark MLlib (Meng et al., 2015) and TensorFlow version in TensorFlow tutorials (Abadi et al., 2015)). It also has many customized extensions (e.g. customized input in word2vecf (Levy and Goldberg, 2014a) and word2vecPM (Li et al., 2017)).

While Mikolov et al. (2013a,d) have already explained in detail how word2vec works, there are still some differences and unclear parts between the theoretical explanations and the practical implementations which may cause misunderstandings of this model. In the subsections below I will talk about three points that have not been discussed in the word2vec papers but play an important role in word embeddings training.

4.4.1 The skip-gram model: Predict each context word from its target word?

The key difference between CBOW model and the skip-gram model is the "direction": CBOW predicts target words from context words; The skip-gram model predicts context words from the target words (in the opposite direction compared to the CBOW model). But if we check the original word2vec C++ implementation or the gensim version, I notice that during the skip-gram training time, these tools are predicting target words from the context words. Let us consider sentence "The history of NLP started in the 1950s" as an example. Given a window size of 1, we then have (context, target) pairs as ([history], The), ([The, of], history), ([history, NLP], of), ([of, started], NLP), etc. According to the definition of skip-gram, the ("from-word", "to-predict-word") training pairs should be like (The, history), (history, The), (history, of), (of, history), (of, NLP), (NLP, of), (NLP, started), etc. Surprisingly, the actual training pairs used in the skip-gram training are (history, The), (The, history), (of, history), (history, of), (NLP, of), (of, NLP), (started, NLP), etc. Note that after training of each sentence, both the theoretical or actual versions of skip-gram have trained the same (context, target) pairs, but in a different training order.

As shown in Figure 4.2, there is no difference from the "direction" point of view between the CBOW architecture and the actual skip-gram architecture. So what causes different results between them? Or in other words, what is the key difference? The answer lies in how they treat context words: CBOW treats all context words entirely for each target word, while skip-gram treats context words individually. That is also the main reason why CBOW is better for smaller datasets (by smoothing over more of the distributional information compared to the skip-gram architecture).

In fact, I am not the first one who noticed this weirdness in the actual execution of skip-gram. In 2013, just after the word2vec source code was published, this question has already been asked in the google group of word2vec-toolkit. Tomas Mikolov explained that because it makes no difference, he swapped the order for cache efficiency[2].

[1]Figure adapted from Figure 1 in (Mikolov et al., 2013a).

[2]https://groups.google.com/forum/#!searchin/word2vec-toolkit/word$20word2/word2vec-toolkit/-AUPLOHGymI/yD4gl0mSNNEJ

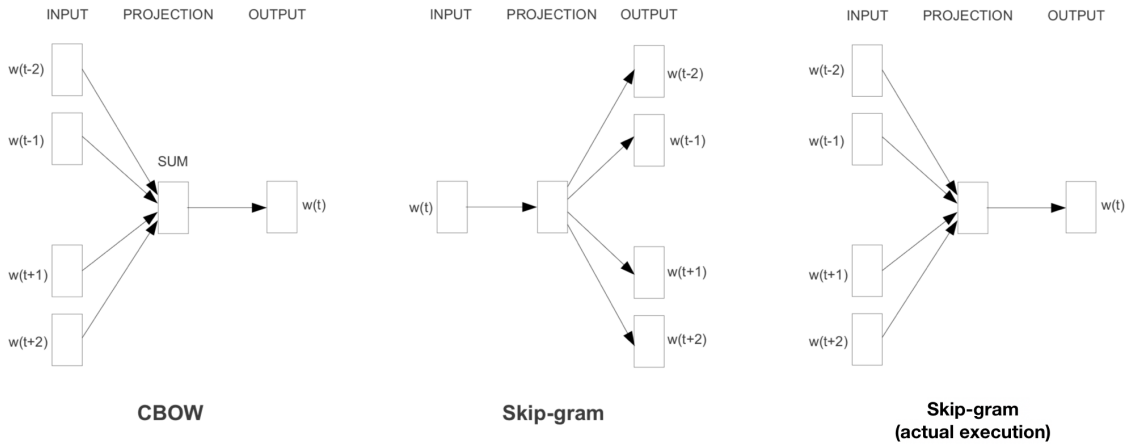


Fig. 4.2 Word2vec model architectures. The CBOW architecture predicts the current word based on the context, the Skip-gram predicts surrounding words given the current word, and the actual execution of the Skip-gram model predicts the current word given each surrounding word.[1]

To verify his explanation, I evaluated the word embeddings generated by using the theoretical order and the actual swapped order of the skip-gram model on word similarity tasks using WordSim-353 (Finkelstein et al., 2001), SimLex-999 (Hill et al., 2014) and Mturk-771 (Halawi et al., 2012) (correlation with humans), and on the word analogy task of Mikolov et al. (2013b) (% correct).

Word embeddings are trained on a part of the English Wikipedia (April 2017) dump of about 1G with window size 5, vocabulary size 50000, vector dimension size 200, start alpha 0.025, end alpha 0.0001, number of iterations 5 and negative examples 5. The Stanford CoreNLP (Manning et al., 2014) is used for dependency parsing. After parsing, tokens are converted to lowercase. I train 5 times for each type of skip=gram implementation and the final results are shown in Figure 4.3.

I examined how other word2vec tools implemented this point. Gensim follows the same swapped order for skip-gram implementation[3] and Levy and Goldberg (2014a) follow the theoretical order by replacing the raw corpus with a collection of training word pairs as input. The results show that the performances of these two implementations of skip-gram are similar. In certain cases however, the swapped order can cause an important difference in performance.

[3]https://github.com/RaRe-Technologies/gensim/issues/300

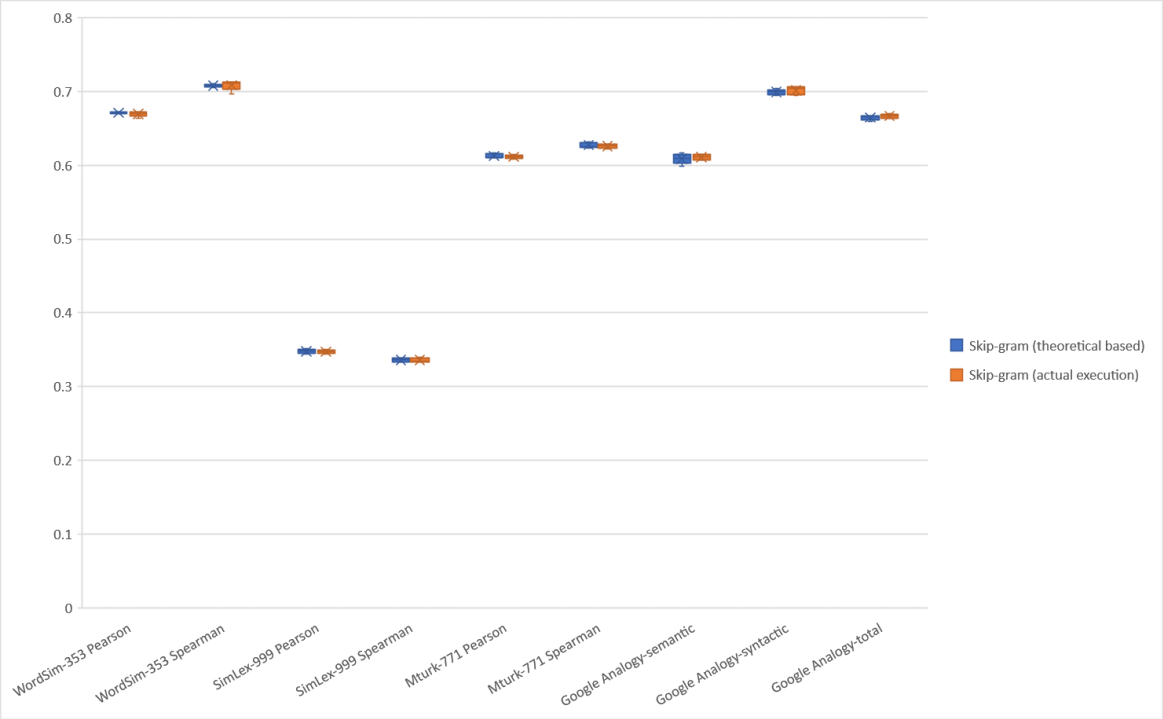


Fig. 4.3 Box plot of evaluation results on WordSim-353, SimLex-999, Mturk-771 and Google word analogy task for two skip=gram implementations (theoretical based and actual execution).

4.4.2 Relation between learning rate and the number of iterations over the corpus

Not all parameters of word2vec model are independent, for instance, the learning rate during training is influenced by the number of iterations.

In most cases, people use the default values of parameters alpha (0.025) and min_alpha (0.0001) which determine the range of the learning rate. While the range has been fixed, the number of iterations over the corpus influences the evolution of the learning rate because of linear learning rate decay, i.e. the learning rate drops linearly from alpha to min_alpha during the whole training.

As shown in Figure 4.4, given default values of alpha and min_alpha, the learning rate drops linearly from 0.025 to 0.0001 over the entire training corpus. Compared to 5 iterations over the corpus (right in Figure 4.4), the learning rate drops more slowly when the number of iterations is set to 1 (left in Figure 4.4) over the same quantity of training corpus. When the iteration number is greater than 1, the learning rate's starting point decreases with each iteration while its range remains the same within each iteration.

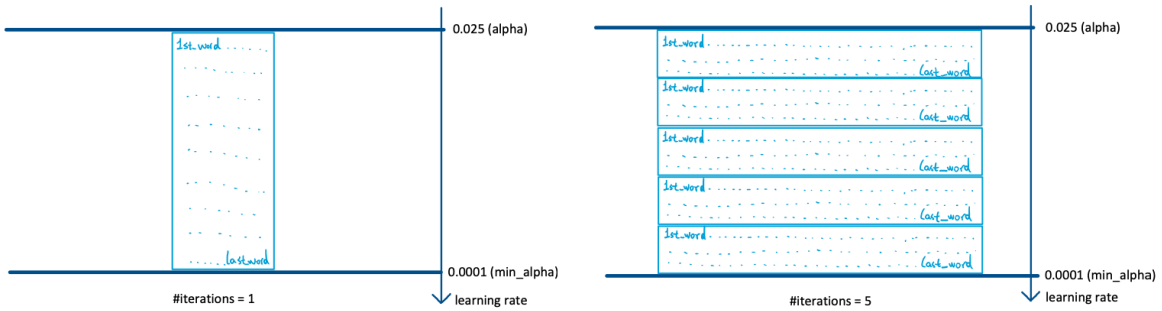


Fig. 4.4 Learning rate drops linearly from 0.025 to 0.0001 over the entire training corpus.

4.4.3 Gensim: Python version of word2vec

Gensim is a popular choice of word2vec implementation because of its simplicity (Pythonic interfaces) and efficiency (using highly optimized C routines and data streaming algorithms), which is also the main reason that my modifications are based on it. Although Gensim is usually considered to be a Python implementation of word2vec, its actual code is executed in the C programming language. To make the code more maintainable and easy to use while keeping the maintaining efficiency, Gensim uses the Cython compiler to generate C code from Cython code.

4.5 Conclusion

I presented in this chapter three graph-based negative sampling models for word2vec. Experiments show that word embeddings trained by using these models can bring an improvement to the word analogy task and to the word similarity tasks.

I found that pre-computing graph information extracted from word co-occurrence networks is useful to learn word representations. Possible extensions would be to test whether using this information to select (positive) target words could improve training quality, and whether using it to reorder training words could improve training efficiency.

In the next chapter, I will move my focus from monolingual context-independent word embedding learning to cross-lingual contextual word embedding learning.

Chapter 5

Explorations in Cross-lingual Contextual Word Embedding Learning

5.1 Introduction

Cross-lingual word embeddings (CLWEs), vector representations of words in multiple languages, are crucial to natural language processing tasks that are applied in multilingual scenarios, such as document classification, dependency parsing, POS tagging, named entity recognition, super-sense tagging, semantic parsing, discourse parsing, dialog state tracking, entity linking or wikification, sentiment analysis and machine translation (Ruder et al., 2017).

As introduced in Section 2.1, cross lingual word embedding learning models can be categorized into three groups based on when the alignment data is used: corpus preparation, training and post-training. For post-training models (see Section 2.4), research about the mapping of state-of-the-art pre-trained monolingual word embeddings across different languages (Devlin et al., 2019; Joulin et al., 2017; Mikolov et al., 2013a; Peters et al., 2018a) keeps evolving with the progress of monolingual word embeddings learning (Conneau et al., 2017; Lefever and Hoste, 2009; Mikolov et al., 2013c; Schuster et al., 2019).

With the most recent progress of word embeddings learning by using pre-trained language representation models such as ELMo (Peters et al., 2018a), BERT (Devlin et al., 2019) and XLNet (Yang et al., 2019), word embeddings move from context-independent to contextual representations. Peters et al. (2018a) have shown that contextual word embeddings have a richer semantic and syntactic representation. For consistency and simplicity, I define these two kinds of representations as word type embedding and token embedding.

Word type embedding Context-independent embedding of each word. A word in the training corpus obtains one embedding.

Token embedding Contextual word embedding of each token. A token is one of the occurrences of a word (type) in a text. Its embedding depends on its context. As a result, a word in the training corpus receives as many embeddings as it has occurrences in that corpus.

Despite many advantages of token embeddings, mapping independently pre-trained token embeddings across languages is challenging: most existing word embeddings, and therefore most existing cross-lingual mapping algorithms, are based on the concept of *word type* embedding, meaning that each word has only one representation, that is hence context independent. How to use multiple token embeddings for one word in order to apply previous cross-lingual word embedding mapping algorithms to token embeddings remains unclear.

Schuster et al. (2019) proposed the current state-of-the-art solution by compressing multiple token embeddings of one word type into one context-independent embedding *anchor*, which enables word-type-based cross-lingual word embedding learning algorithms to apply to token embeddings. In their paper, the compression of token embeddings is simply obtained by averaging them.

Although experiments show that this simple average anchor calculation is effective for cross-lingual token embeddings mapping, i.e. it obtained a better score on dependency parsing tasks than the previous state-of-the-art method, I believe there is still room for improvement especially for multi-sense words. Schuster et al. (2019) found that token embeddings for each word are well separated like clouds, and the token embeddings of a multi-sense word may also be separated according to different word senses inside each token embedding cloud.

Based on these findings, I argue that averaging is not a good choice for multi-sense word anchor calculation, which directly influences the cross-lingual token embeddings learning.

- For the supervised mapping methods (Mikolov et al., 2013c; Xing et al., 2015), the average anchor of a multi-sense word depends on the frequency of the token embeddings of each word sense. Besides, as each translation pair containing multi-sense words in the supervision dictionary may only cover one sense at one time, using only one anchor for each multi-sense word may not correspond to mono-sense based translation pairs.

- For the unsupervised cross-lingual word embedding learning model MUSE (Conneau et al., 2017), because a multi-sense word may not have a translation word that would exactly have all its senses, the average anchor of that word may not find a corresponding average anchor embedding of a word in the target language.

My contributions: To address these problems, I firstly analyze the geometric distribution of token embeddings of multi-sense words, suggesting its relation to sense embeddings (Section 5.3.1). Then I show the existing problems of using average anchor embeddings (Section 5.3.2) for both supervised (Section 5.3.3) and unsupervised (Section 5.3.4) cross-lingual word embedding learning models. Finally, I propose my solutions by treating multi-sense word anchor embeddings as noise (Section 5.4). I also discuss other clustering-based ideas and difficulties in evaluating cross-lingual token embeddings (Section 5.7).

5.2 Related work

In Section 2.5, I introduced two recent papers about cross-lingual contextual word embedding (token embedding) learning. One relies on using parallel sentences (Aldarmaki and Diab, 2019) either to generate a dynamic dictionary of token embeddings as the word-level alignment data or to calculate sentence embeddings as the sentence-level alignment data. Another proposed to compress the token embeddings of each word into one anchor embedding (Schuster et al., 2019) so as to apply previous cross-lingual word embedding learning algorithms. Here I focus on the solution of Schuster et al. (2019) as it does not need the additional alignment data and it aims to connect all previous cross-lingual word embedding learning algorithms to the token embeddings field.

Below I introduce two cross-lingual word embedding learning methods (one supervised method in Section 5.2.1 and another unsupervised method in Section 5.2.2) along with their adaptations for token embeddings proposed by Schuster et al. (2019).

5.2.1 Supervised mapping

Supervised mapping methods aim to learn a linear mapping using the supervision of alignment data. Mikolov et al. (2013c) introduced a model that learns a linear transformation between word embeddings of different languages by minimizing the sum of squared Euclidean distances for the dictionary entries. Based on this work, Xing et al. (2015) proposed an orthogonal transform to map the normalized word vectors in one or both languages under the constraint of the transformation being orthogonal because of two inconsistences in (Mikolov et al., 2013c):

- During the skip-gram model training stage, the distance measurement is the inner product of word vectors according to the objective function while the cosine similarity is usually used for word embedding similarity calculation (e.g. for the WordSim-353 task).

- The objective function of the linear transformation learning (Mikolov et al., 2013c) uses the Euler distance. But after mapping, the closeness of bilingual words is measured by the cosine similarity.

Xing et al. (2015) made experiments that showed that normalized word vectors have a better performance in monolingual the word similarity task WordSim-353 and that the proposed method performs significantly better in the word translation induction task than (Mikolov et al., 2013c).

Adaptation for token embeddings Given a dictionary used for supervised cross-lingual context-independent word (word type) embedding learning, Schuster et al. (2019) proposed to generate average token embeddings anchors and to assign word anchor vectors to dictionay words.

$$\overline{e}_i = \mathbb{E}_c \left[e_{i,c} \right] \tag{5.1}$$

As shown in Equation 5.1, the anchor embedding of word i is defined as *the average of token embeddings over a subset of the available unlabeled data*, where $e_{i,c}$ is the token embedding of word i in the context c.

5.2.2 Unsupervised mapping: MUSE

MUSE (Multilingual Unsupervised and Supervised Embeddings) is a Generative Adversarial Net (GAN)-based method and open-source tool introduced by Conneau et al. (2017). In their paper, a discriminator is trained to determine whether two word embeddings uniformly sampled from the $50,000$ most frequent words either come from the WS (aligned source word embeddings, where S is the source word embeddings and W is the linear transformation matrix) or T (target word embeddings) distributions. In the meantime, W is trained to prevent the discriminator from doing so by making elements from these two different sources as similar as possible. Besides, they defined a similarity measure Cross-domain Similarity Local Scaling (CSLS) that addresses the hubness problem (i.e., some points tend to be nearest neighbors of many points in high-dimensional spaces), and serves as the validation criterion for early stopping and hyper-parameters tuning.

Adaptation for token embeddings Schuster et al. (2019) also proposed another adaptation on top of the MUSE model Conneau et al. (2017) by using anchor embeddings: as they did in the supervised case, anchor embeddings are assigned as the vector representations for words. Then they use them in the unsupervised MUSE model.

5.3 Average anchor embedding for multi-sense words

Averaging for anchor's calculation is based on two findings from Schuster et al. (2019)'s exploration of token embeddings:

1. The clouds of token embeddings of each word are well separated.
2. (a) The clouds of multi-sense words may be separated according to distinct senses.
   (b) Although the distances between token embeddings and the averaged token embedding cloud center are slightly larger than in single-sense words, the token embeddings of multi-sense words *still remain relatively close to their [...] anchor*. Because of this, the authors believe "these anchors can still serve as a good approximation for learning alignments".

It does not sound logical that token embeddings of a multi-sense word are close to their anchor. Take the English word "bank" as an example, which has multiple distinct senses including the meaning of a financial institution and the meaning of the river side. There is no reason why token embeddings related to the financial institution meaning should be close to token embeddings of the river side meaning.

I decided to investigate these claims by analyzing monolingual and aligned cross-lingual token embeddings. My empirical investigation is consistent with the first conclusion and the first point of the second conclusion, but disagrees with the second point of the second conclusion. Additionally, I attempt to explain why this second point is not likely to hold in principle.

5.3.1 Token embeddings

To show the difference of token embedding geometrical distributions between multi-sense words and single-sense words, I need a multi-sense word that is directly related to single-sense words. The English word "lie" could be a good choice: the verb "lie" has two distinct senses, and each sense has a different past tense: *lied* (did not tell the truth) or *lay* (was in a horizontal position, was located)[1]. Besides, the English word "lie" can also be a noun, whose antonym is "truth".

So I project the embeddings of the English word "lie" along with its two past tenses "lied" and "lay" and one of its antonyms, "truth".

As shown in Figure 5.1, I found that the point clouds of the single-sense words "lied" and "truth" are more concentrated than for the multi-sense word "lie". The point cloud of

[1] https://jakubmarian.com/lie-lied-lay-laid-and-layed-in-english/

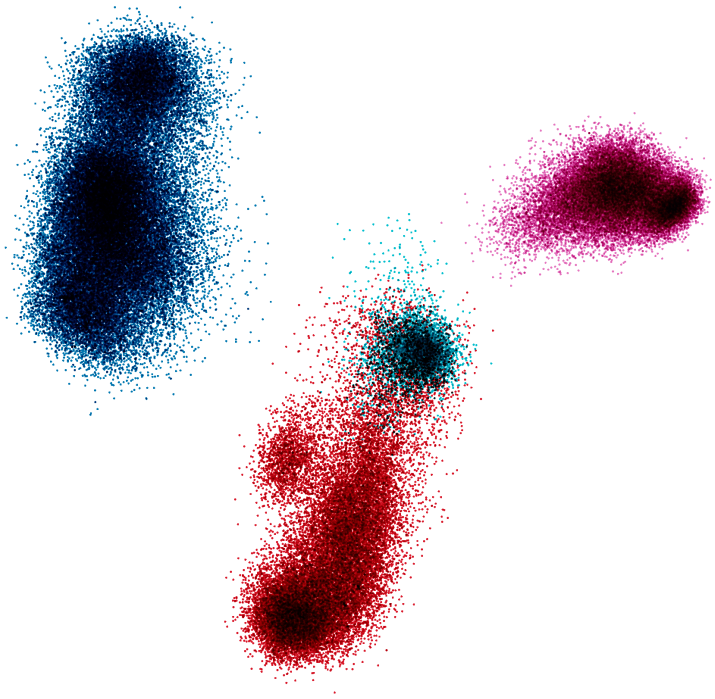


Fig. 5.1 Token embeddings of the English word “lie” (points in red, bottom middle) along with its two past tenses “lied” (points in light blue, top middle) and “lay” (points in dark blue, top left) and one of its antonyms “truth” (points in purple, top right).

the word “lie” can be visually categorized into 3 clusters: one overlaps the cloud of “lied” in light blue, one at the bottom and another one on the left. By randomly selecting points and checking their corresponding sentences (Table 5.1) from each cluster, I found that the point clouds of the word “lie” are separated according to its distinct senses. Surprisingly, I also found that the point cloud of the word “lay” is also visually separated into 2 parts. By checking the corresponding sentences, I found the bottom part is used as the past tense of the word “lie” and the top part is used as an adjective (Three corresponding sentences: “In 1980, Mr. Greg Coffey was appointed the first lay principal of the College.”, “Conwell took up the post at an advanced age, and spent much of his time there feuding with the lay trustees of his parishes, especially those of St. Mary’s Church in Philadelphia.” and “This includes a wide range of relationships, from monastic (monks and nuns), to mendicant (friars and sisters), apostolic, secular and lay institutes.”).

Similar findings can be found in the token embeddings of other words, in different languages and also in aligned cross-lingual embedding spaces (as shown in Figure 5.4). As suggested by Schuster et al. (2019)’s conclusion, point clouds for each word are well separated. Besides, the point clouds of multi-sense words are also separated according to distinct senses.

Cluster position	Sentence	Semantic category
overlapping	*Yutaka and Lady Elizabeth come to the hearing and **lie** to incriminate Oyuki.*	[verb] to deliberately say sth that is not true
overlapping	*s a result of his confession, prosecutors decided not to pursue a prosecution against the remaining 20 charges, and asked that they **lie** on file, in order to spare a jury the horror of having to watch graphic images and videos of child abuse since the 71 charges which Huckle admitted to would be sufficient for a lengthy sentence.*	[verb] to deliberately say sth that is not true
bottom	*The city's prime locations **lie** within a radius of 6 km from Thammanam, making it thus a predominantly residential and small commercial area with basic facilities in and around the region.*	[verb] to be in a particular position
bottom	*As of 2009, the most heavily trafficked segments of NY 31 **lie** in and around the city of Rochester.*	[verb] to be in a particular position
left	*James Murphy later admitted that this was entirely a **lie** on his part, and that he does not actually jog.*	[noun] sth you say that you know is not true
left	*The dater then asks the suitors questions which they must answer while hooked up to a **lie** detector, nicknamed the "Trustbuster".*	[noun] sth you say that you know is not true

Table 5.1 Corresponding sentences selected from each visual clusters of the token embeddings of the word "lie"

5.3.2 Average anchor embeddings for multi-sense words

To analyze multi-sense word token embeddings and their average anchors in detail, I manually selected 4 multi-sense English words from the Wikipedia list of true homonyms[2] from different perspectives:

- Distinct senses of the same part of speech (POS) (noun): **bank**-financial, **bank**-river, etc.; **spring**-season, **spring**-fountain, **spring**-coiled, etc.
- Distinct senses of different POS: **check**/Noun **check**/Verb; **clear**/Adj, **clear**/Verb

Distribution of token embeddings for multi-sense words. I firstly calculate all the token embeddings of the selected words over the whole English Wikipedia. For the embeddings visualization, I use the output of both the first and the second LSTM layers of ELMo (see Figure 5.2).

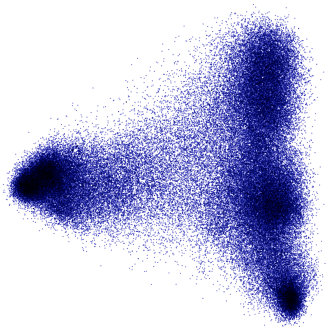
Position of anchor embeddings for multi-sense words. Besides the embeddings projection, I also calculate anchor embeddings for the selected multi-sense words. Then I label the 100 nearest neighbors of each anchor in the token embedding space (see Figure 5.3). Note that all token embeddings are also present in that second visualization, but only the top 100 are labeled with the word.

Context of token embeddings. Also, to verify that token embeddings are geometrically separated according to distinct senses, for each cluster in the point cloud of a multi-sense word, I randomly select two points (token embeddings) inside and show their corresponding sentences (see Table 5.2). Note that I do not apply any clustering algorithms, clusters are just recognized based on human judgment. We return to this limitation below (Section 5.7, Discussion and future work).

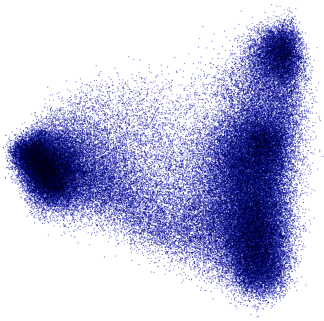
Observation. As shown in Figure 5.3, most of the 100 token embeddings nearest to the anchor embedding are located in only one of the word sense clusters. The anchor is pulled closer to the sense clusters that have more token embeddings because of the averaging, which causes the first problem for cross-lingual token embeddings mapping:

Problem 1 The anchor of a multi-sense word is biased by the frequency of the token embeddings of its senses.

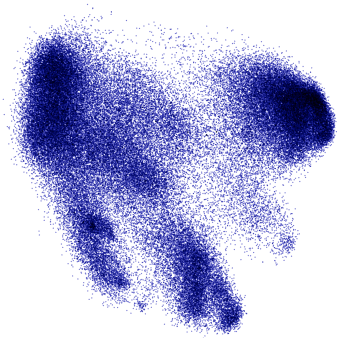
[2]https://en.wikipedia.org/wiki/List_of_true_homonyms, Retrieved 18 June 2019



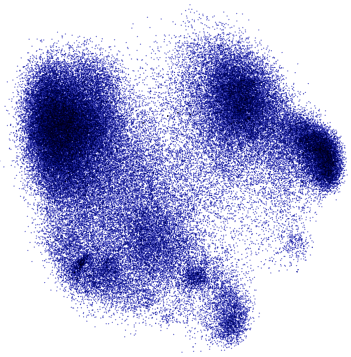
bank @ 1st output layer



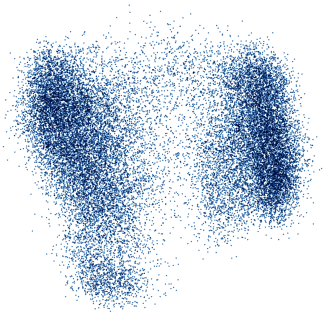
bank @ 2nd output layer



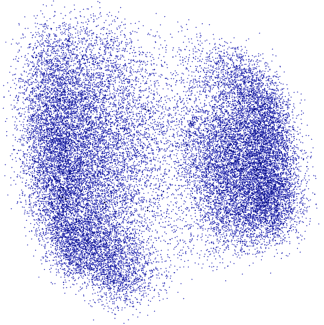
spring @ 1st output layer



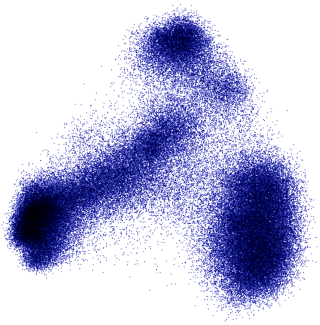
spring @ 2nd output layer



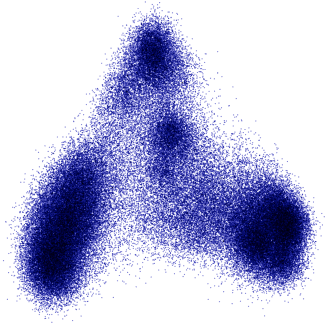
check @ 1st output layer



check @ 2nd output layer

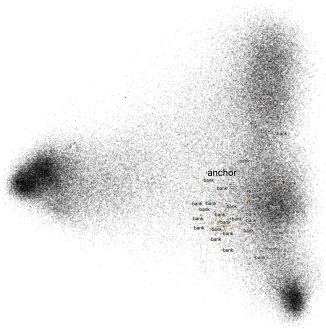


clear @ 1st output layer

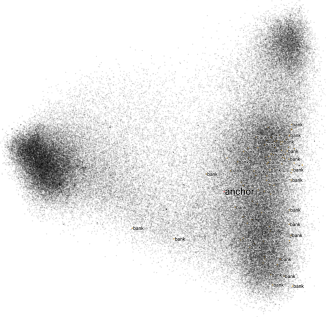


clear @ 2nd output layer

Fig. 5.2 Token embeddings of English words “bank”, “spring”, “check” and “clear” generated from the first and the second LSTM layers of the ELMo model.

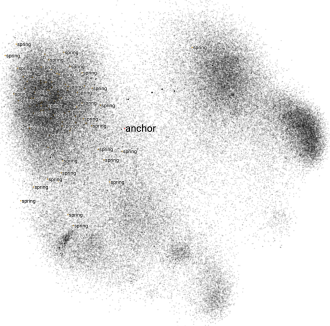


bank @ 1st output layer

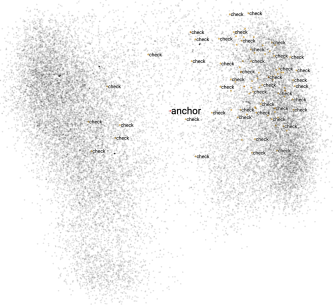


bank @ 2nd output layer

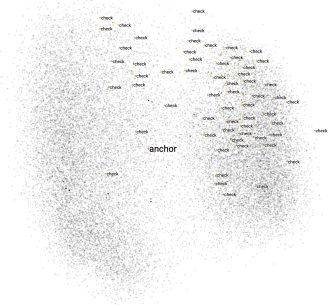
spring @ 1st output layer



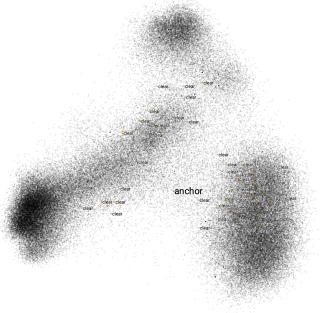
spring @ 2nd output layer



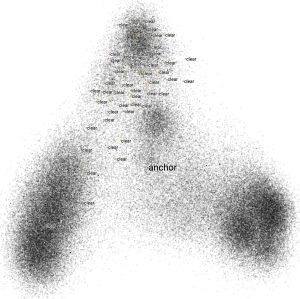
check @ 1st output layer



check @ 2nd output layer



clear @ 1st output layer



clear @ 2nd output layer

Fig. 5.3 Labelling of the anchor embeddings (*anchor*) of English words “bank”, “spring”, “check” and “clear” and of their 100 nearest token embeddings (*bank*, *spring*, etc.). Embeddings are generated from the first and the second LSTM layers of the ELMo model.

Word	Cluster Positions	Sentences @ 1st layer	Sentences @ 2nd layer
bank	left	Small Craft Company USMC assisted in locating the bodies of the slain snipers and were engaged in a large fire fight on the east **bank** of the Euphrates River in the city of Haditha.	At the northern **bank** of the Svir River () the Finnish army had prepared a defence in depth area which was fortified with strong-points with concrete pillboxes, barbed wire, obstacles and trenches.
		The population on the east **bank** of the Weser had not prepared adequate defenses, so the crusading army attacked there first, massacring most of the population; the few survivors were burnt at the stake.	These specimens were collected at the Karagachka locality (locality 34 or PIN 2973), to the opposite **bank** of the Karagatschka River from Karagachka village located in a drainage basin of left bank of the Ural River, Sol'Iletsk district of Orenburg Region, southern European Russia.
	right	If government bonds that have come due are held by the central **bank**, the central bank will return any funds paid to it back to the treasury.	Issue **bank** notes;
		Liz is astonished when the police suddenly arrive at the pub to tell her that Jim has been caught robbing a **bank** and now has a number of hostages.	Although such measures were not effected, the new administration was successful in tackling other issues: both deficit and the cost of living dropped while the **bank** reserves trebled, and some palliatives were introduced in lieu of a land reform (the promised tax cuts, plus the freeing of "mainmorte" property).
spring	top left	However, after reaching Ulster the horse stops and urinates, and a **spring** rises from the spot.	The **spring** had been shut off by a rock 74 meters long and 30 meters wide, which obstructed the construction of a running water system.
		Over running water – Literally "living", that is, **spring** water.	The holy **spring** is known to change its colour with various hues of red, pink, orange, green, blue, white, etc.
	bottom left	A 5'10", 170-pound infielder, Werber was at **spring** training and toured for several weeks in July with the Yankees in 1927.	Joss attended **spring** training with Cleveland before the start of the 1911 season.
		He was invited to **spring** training and sent to minor league camp on March 14.	He pitched in the California Angels minor league system in the early 1990s and participated in "Replacement player" **spring** training games in 1995 for the Toronto Blue Jays.
	bottom middle	In **spring** 912, the Jin attack against Yan got underway, with Zhou commanding the Jin army in a joint operation with the Zhao general Wang Deming (Wang Rong's adoptive son) and the Yiwu Circuit (headquartered in modern Baoding, Hebei) army commanded by Cheng Yan (whose military governor, Wang Chuzhi, was also a Jin ally).	In **spring** 2017, Ponders hit the road supporting Pouya and Fat Nick, opening to sellout crowds across Ontario and Quebec.
		In **spring** 2010 CSX railroad removed the diamonds connecting the southern portion of the Belt Railroad, thus isolating the line from the U.S. rail system.	In **spring** 1944, the Rabstejn sub-camp of Flossenburg was created here, with a capacity of 600 prisoners.
	right	In the **spring** of 1935, the All-Union Organization of Cultural Relations with Foreign Countries agreed to send a delegation to the upcoming First International Festival of the Folk Dance in London.	Hirsig's role as Crowley's initiatrix reached a pinnacle in the **spring** of 1921 when she presided over his attainment of the grade of Ipsissimus, the only witness to the event.
		In the **spring** of 2012 in Pakistan was established Pakistani mission.	Brown wrote, "In the **spring** of 1819 a nightingale had built her nest near my house.
check	left	Because the defined cases are exhaustive, the compiler can **check** that all cases are handled in a pattern match:	It is standardized for use by mud engineers to **check** the quality of drilling mud.
		Most spotters maintained books of different aircraft fleets and would underline or **check** each aircraft seen.	The lowest level, where the sounds are the most fundamental, a machine would **check** for simple and more probabilistic rules of what sound should represent.
	right	Usually, the trial **check** will quickly reject the trial match.	It is important to realize that glucose-glutamic acid is not intended to be an accuracy **check** in the test.
		The donor's hematocrit or hemoglobin level is tested to make sure that the loss of blood will not make them anemic, and this **check** is the most common reason that a donor is ineligible.	U.S. Attorney General John Mitchell, citing an extensive background **check** by the Justice Department, was willing to forgive, stating that it was unfair to criticize Carswell for "political remarks made 22 years ago."
clear	top	From here, she had to fight an uphill battle to **clear** her name and proved her right by finding the authentic painting, while she was also struggling with financial hardship and interference from Min Jung-hak.	On 1 November, Ouagadougou Mayor Simon Compaoré led volunteers on "Operation Mana Mana" (Operation Clean-Clean in Dyula) to **clear** the streets, which earned him praise on social media.
		Jones' shoulder injury came after Botha attempted to **clear** him from a ruck and the Bulls star was subsequently cited and banned for two weeks for the challenge.	Again a gold medal favourite in the 110 metre hurdles at the London Olympics he pulled his Achilles tendon attempting to **clear** the first hurdle in the heats.
	Bottom left	She made it **clear** that she did not intend for Nassar to ever be free again.	Hugenberg for his part regarded "Katastrophenpolitik" as a good idea that was unfortunately abandoned, and made it **clear** that he wanted a return to "Katastrophenpolitik".
		Many Southerners felt that the Compromise of 1850 had been shaped more towards Northern interests; the Georgia Platform made it **clear** that the future of the nation depended on the North strictly adhering to the Compromise.	The political heat was turned up on the issue since Bush mentioned changing Social Security during the 2004 elections, and since he made it **clear** in his nationally televised January 2005 speech that he intended to work to partially privatize the system during his second term.
	Bottom right	However, in "Reference re Secession of Quebec", the Supreme Court of Canada has essentially said that a democratic vote in itself would have no legal effect, since the secession of a province in Canada would only be constitutionally valid after a negotiation between the federal government and the provincial government; whose people would have clearly expressed, by a **clear** majority, that it no longer wished to be part of Canada.	He was the **clear** winner with ten seconds over the runner-up, fellow Kenyan Albert Kiptoo Yator.
		The game sees Kasparov rejecting **clear** drawing opportunities and eventually losing.	He wrote to Irene Tasker in South Africa, in a **clear** hand, telling her how much better he was.

Table 5.2 Corresponding sentences selected from the token embedding clusters of the English words "bank", "spring", "check" and "clear".

5.3.3 Muti-sense words in dictionaries for supervised mapping

The supervised model is trained on a bilingual dictionary of source-target words. Dictionaries are not always generated with attention paid to multi-sense words. When a dictionary contains incomplete translation pairs related to a multi-sense word, it may contribute inaccurate mapping supervision data.

Let us take as an example the English-French dictionary, containing 5,000 source words, used for the supervised baseline model in MUSE. I list in Table 5.3 all translation pairs in that dictionary related to a common multi-sense word: "bank".

bank	banques
bank	banque
banks	banques
banking	banques
banking	banque
banking	bancaire

Table 5.3 All translation pairs related to the multi-sense word "bank" in the English-French dictionary used in MUSE for supervised mapping.

It is obvious that all translation pairs listed above are related to the "financial institution" meaning of the word "bank". The other senses of bank, such as "land at river's side", are ignored. Similar cases can be found for other multi-sense words in the dictionary.

Problem 2 Because the average anchor for a multi-sense word can be considered as a general representation of all its distinct senses, using multi-sense word anchors for semantically incomplete translation pairs in a dictionary may lead to inaccurate mappings.

5.3.4 Muti-sense words for the unsupervised mapping in MUSE

As mentioned in Section 5.2.2, the unsupervised mapping model in MUSE uses a GAN to learn a linear mapping between source and target embeddings without parallel supervision data. Based on the intuition that source and target embedding spaces should share a similar global geometric structure, in the best case, source words should be mapped to their corresponding translation words in target languages.

Problem 3 For multi-sense words, translations that have exactly the same set of senses may not exist, e.g. for the English word "bank", there is no corresponding French word which has both the "financial institution" ("banque") and "land at river's side" ("berge", "bord", "rive",

etc) senses. Therefore a multi-sense word anchor may not have a corresponding point in the target language.

5.4 Cross-lingual token embeddings mapping with multi-sense words in mind

I propose solutions to these problems for both supervised mapping and unsupervised mapping methods below.

5.4.1 Noise in dictionary for supervised mapping

I consider incomplete translation pairs of multi-sense words as noise in the supervision data (dictionary). A simple but effective solution is to remove noise. Here I propose two types of removal:

- Exact removal: remove translation pairs that contain the exact multi-sense words. For instance, given that the source word “bank” is known to have multiple senses, “bank banques” and “bank banque” should be removed in Table 5.3.
- Lemma-based removal: remove translation pairs containing words having the same lemma as multi-sense words. In the “bank” example, all 6 translation pairs in Table 5.3 should be removed as “bank”, “banks”, and “banking” have the same lemma.

Note that I do not supply the part of speech (POS) to the lemmatizer as there is no context to analyze the POS for words in the translation pairs of the dictionary.

5.4.2 Noisy points for unsupervised mapping in MUSE

As discussed in Section 5.3.4, the exact corresponding senses-to-senses translation of a multi-sense word may not exist in target languages, i.e. the average anchor for multi-sense words may not be correctly aligned to target embedding spaces.

In that context, I consider multi-sense word anchors as noise for the unsupervised mapping model in MUSE. So I remove all multi-sense word anchors from the independently pre-trained monolingual word embeddings used for training.

5.5 Experiments

5.5.1 Token embeddings

Pre-trained model I use the same ELMo models[3] as in (Schuster et al., 2019), which are trained on Wikipedia dumps[4] with the default parameters of ELMo (Peters et al., 2018a).

Corpus The Wikipedia dumps I used for specific words analysis are the same as the training data for ELMo models.

Lexicon induction evaluation Following (Schuster et al., 2019), I use average anchors to produce word translations to evaluate alignments. Gold standard dictionaries are taken from the MUSE framework[5] and contain 1,500 distinct source words.

5.5.2 Supervised mapping

Dictionary The baseline supervised linear mapping is calculated based on a dictionary of 5,000 distinct source words downloaded from the MUSE library[6].

Corpus for word occurrence embedding and anchor calculation I compute the average of token embeddings on a fraction (around 500MB, or 80 million words) of English (/French) Wikipedia dumps as anchor vectors for the English (/French) words in dictionaries.

5.5.2.1 Detailed analysis about "bank"

To obtain an intuitive understanding of how multi-sense words behave in supervised mapping methods, I start my supervised mapping experiment focusing on a common English multi-sense word "bank".

2 dictionaries used for supervised linear mapping To analyze the influence of incomplete translation pairs about "bank" in the dictionary, I generate two additional dictionaries by removing translation pairs containing "bank" (exact removal: "bank ⇔ banques and bank

[3]https://github.com/ TalSchuster/CrossLingualELMo.
[4]https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1989
[5]https://github.com/facebookresearch/MUSE/
[6]https://github.com/facebookresearch/MUSE

⇔ banque) and by removing translation pairs having the same lemma as "bank" (lemma-based removal: "bank ⇔ banques, bank ⇔ banque, banks ⇔ banques, banking ⇔ banques, banking ⇔ banque, and banking ⇔ bancaire").

For token embeddings visualization, I compute token embeddings of the English word "bank" and of its French translations (i.e. "banque", "bord", "rive", and "berge", according to the Collins English-French Dictionary[7] and WordReference.com[8]) over around 500*MB* English and French corpora.

5.5.2.2 Removal of English multi-sense words

Based on the English homonyms list from Wikipedia[9], I generate two dictionaries by exact removal and lemma-based removal. The original dictionary has 9496 valid translation pairs, the exact removal dictionary has 9161 valid translation pairs and the lemma-based removal dictionary has 9076.

5.5.3 Unsupervised mapping

I calculate token embeddings for the 50,000 most frequent words in English and in the target language. For frequent words selection, I follow the word order in FastText pre-trained word vectors[10], which are *sorted in descending order of frequency*. The corpus used for anchor calculation is the same as the one used for supervised mapping.

5.5.4 Set-up for embedding visualization

Embedding Projector[11] has been used for data visualization. I generate two 2-D graphs for each selected polysemy (or polysemies) by selecting the "PCA" (Principal Component Analysis) for dimensionality reduction and "Sphereize data" *(The data is normalized by shifting each point by the [coordinates of the] centroid and making it unit [length])* for data normalization.

Note that PCA is approximate in the Embedding Projector, i.e., *for fast results, the data was sampled to 50,000 points and randomly projected down to 200 dimensions.* As token embeddings generated by ELMo have 1024 dimensions, the embeddings used for visualization were randomly projected down to 200 dimensions.

[7] https://www.collinsdictionary.com/dictionary/english-french/bank
[8] https://www.wordreference.com/enfr/bank
[9] https://en.wikipedia.org/wiki/List_of_true_homonyms
[10] https://fasttext.cc/docs/en/crawl-vectors.html
[11] http://projector.tensorflow.org

To prove these two approximations will not largely influence the PCA output, I apply Embedding Projector to the 129,857 token embeddings of the English word “bank” and the French words “banque” and “berge” mapped into the common space 9 times with different random initializations. The result is shown in Figure 5.4 where points in blue are for English

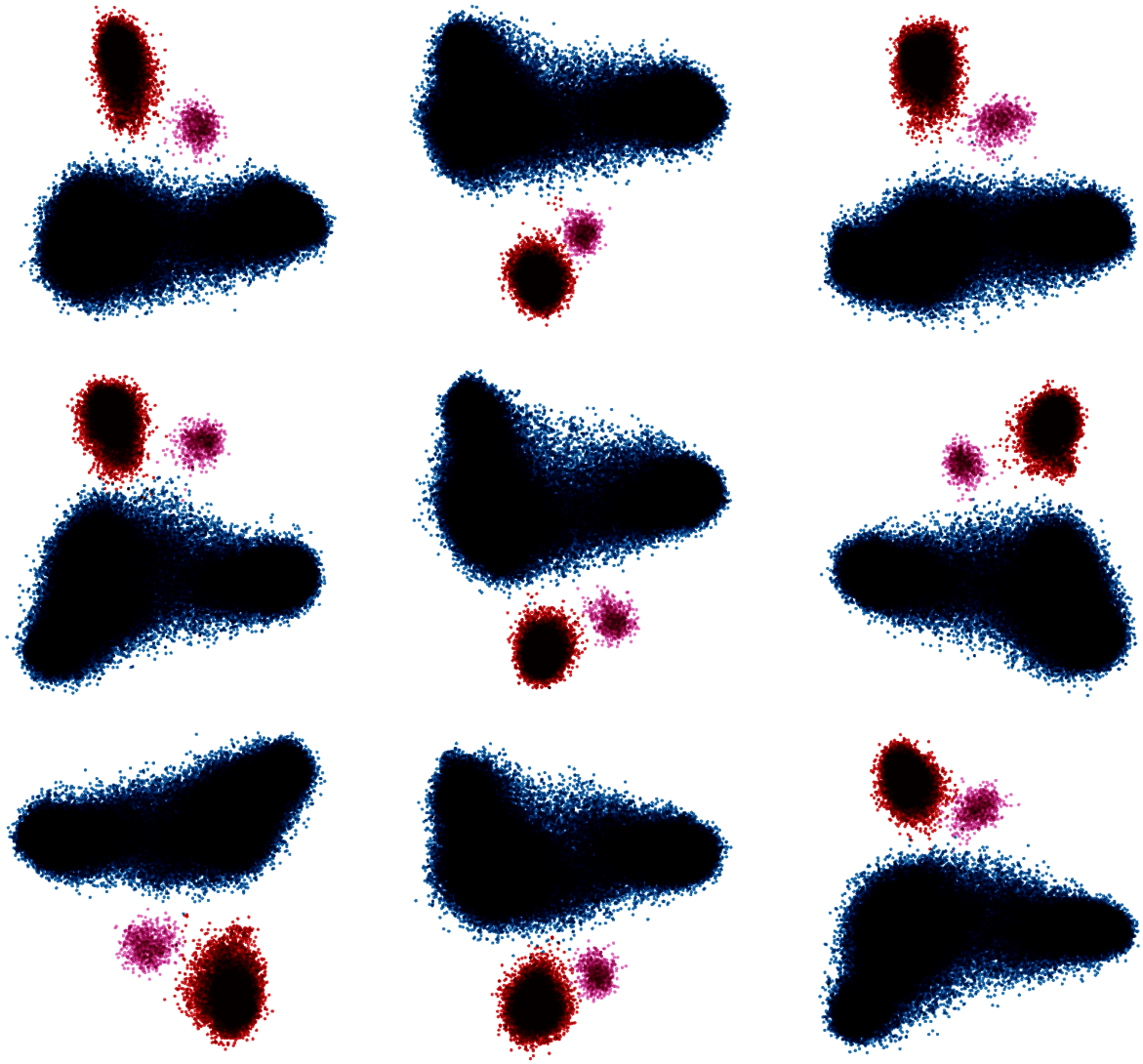


Fig. 5.4 Visualizations of the embeddings of the English word “bank” (blue points) and French words “banque” (red points) and “berge” (pink points) by Embedding Projector.

word “bank” and red and pink points are for French words “banque” and “berge” respectively. The embedding projections keep a similar structure in the 2-D visualization.

The 3-D visualizations (as shown in Figure 5.5) are even more similar compared to the 2-D versions. The 2-D versions are like looking at the 3-D version from slightly different views.

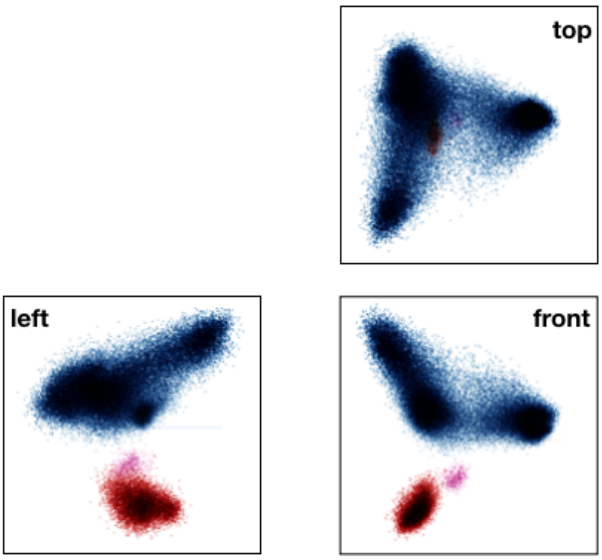


Fig. 5.5 Top, front and left views of the 3-D visualization of the token embeddings of the English word “bank” (blue points) and French words “banque” (red points) and “berge” (pink points) by Embedding Projector.

5.6 Results

5.6.1 Visualization of the token embeddings of “bank”

The results of experiments in Section 5.5.2.1 are shown in three figures presented below, where the dark blue points represent the English word “bank”, the light blue points are token embeddings for the French word “banque”, and the French words “berge”, “bord”, “rive” are in green, red and pink colors respectively.

In the baseline aligned embedding space (Figure 5.6), the point cloud of “banque” is close to the middle part of the point cloud of “bank”. After removing the exact “bank” translation pairs (see Figure 5.7) and the translation pairs containing words having the same lemma as “bank” (see Figure 5.8), the point cloud of “banque” is moving to the top part of the “bank” point cloud, which is the cluster of the “financial institution” meaning of “bank”.

I take this as meaning that after removing incomplete supervision data (translation pairs in the dictionary) for multi-sense words, the alignment for multi-sense words is indirectly improved thanks to better supervision data for general embedding spaces mapping.

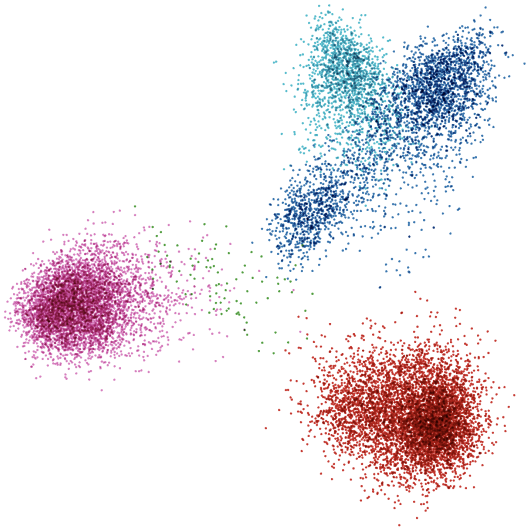


Fig. 5.6 Baseline aligned token embeddings for the English word “bank” (in dark blue) and the French words “banque” (in light blue), “berge” (in green), “bord” (in red) and “rive” (in pink).

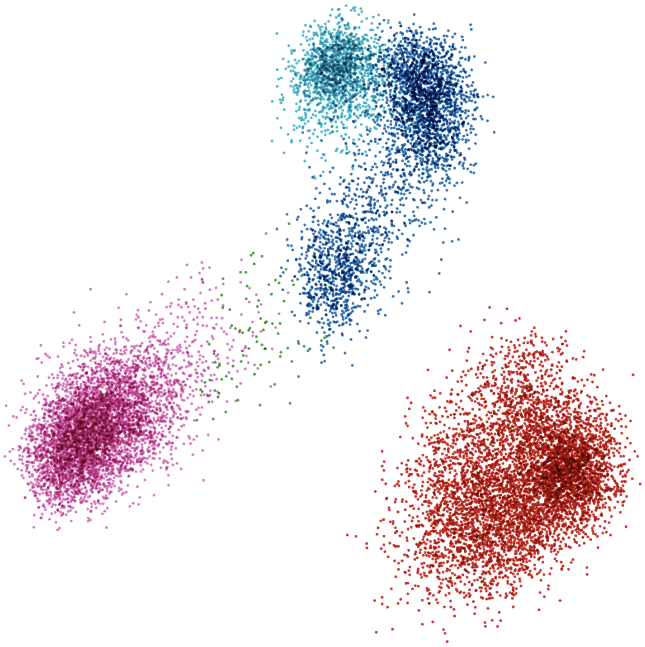


Fig. 5.7 Aligned token embeddings for the English word “bank” (in dark blue) and French words “banque” (in light blue), “berge” (in green), “bord” (in red) and “rive” (in pink) after removing exact “bank” translation pairs.

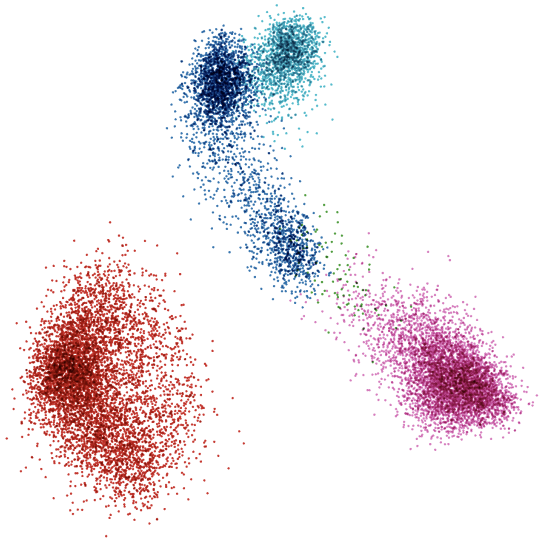


Fig. 5.8 Aligned token embeddings for the English word “bank” (in dark blue) and French words “banque” (in light blue), “berge” (in green), “bord” (in red) and “rive” (in pink) after removing translation pairs having the same lemma as “bank”.

5.6.2 Lexicon induction task

In Tables 5.4 and 5.5, I show the accuracy of the lexicon induction task based on different alignments.

Alignment	P@top1	P@top5	P@top10
Supervised baseline	55.95	73.57	79.49
Exact homonyms removal	55.33	73.43	79.22
Lemma-based homonyms removal	55.33	73.30	79.15

Table 5.4 Supervised method (second LSTM output layer of ELMo). Precision at $k = 1, 5, 10$ of bilingual lexicon induction from the aligned cross-lingual embeddings

Alignment	P@top1	P@top5	P@top10
Unsupervised baseline	42.81	62.70	67.72
English homonymous anchor embedding removal	52.44	67.38	72.06

Table 5.5 Unsupervised MUSE model (first LSTM output layer of ELMo). Precision at $k = 1, 5, 10$ of bilingual lexicon induction from the aligned cross-lingual embeddings

For supervised cross-lingual word embedding alignment (5.4), I found that removing exact homonym-related translation pairs or translation pairs containing words having the

same lemma as homonym words does not largely affect the lexicon induction task results (around 0.6% difference in the precision at $k = 1$).

For unsupervised cross-lingual word embedding alignment (5.5), I found that removing exact homonym-related anchor embeddings improves the P@top1 by 10 points and the P@top5 and P@top10 by 5 points. Removing noisy information about multi-sense words is therefore very beneficial in this case.

5.7 Discussion and future work

Cross-lingual contextual word embedding learning is a new field, research about it just started and is moving fast. Below I present my thoughts about this topic and the related possible future work.

5.7.1 Clustering

While treating multi-sense words as noise is a simple but efficient solution, it has several drawbacks:

- An additional resource, a multi-sense words list, is always needed for both the supervised methods and the unsupervised MUSE model.
- Removing translation pairs from the dictionary, even noisy ones, reduces the scale of the word-level supervision data. It may decrease the cross-lingual word embeddings quality as there is less supervision data.

I believe that token embeddings clustering could help find solutions to these problems:

- The token embeddings of a word are expected to be clustered according to its distinct senses. This could be used to train a multi-sense word detector that would be based on the number of clusters of token embeddings for each word.
- In the current average anchor generation, each word type has only one anchor. We could adapt average anchor generation from the word type level (token embeddings of a word) to the word sense level (clusters in token embeddings of a word) to reduce the average anchor problems we mentioned above.

5.7.2 Evaluations

Although there are many evaluation datasets and tasks for cross-lingual word embeddings as introduced in Section 2.6 and for monolingual contextual word embeddings (token embed-

dings) (see Section 1.2.4), I could not find a suitable evaluation set for cross-lingual word embeddings taking the multi-sense word case into account. It is hard therefore to measure the improvement of the alignment for multi-sense word embeddings macroscopically. So far I only analyzed multi-sense word embedding alignment case by case (like in Section 5.6.1). Therefore, creating a new evaluation task for cross-lingual contextual word embeddings (token embeddings) with attention to multi-sense words could be a meaningful future work.

5.8 Conclusion

In this chapter, I explored the contextual word embeddings (token embeddings) of multi-sense words, argued that the current state-of-the-art method for cross-lingual token embedding learning cannot handle multi-sense words well and proposed my solutions by considering multi-sense word token embeddings as noise. Experiments showed that my methods can improve the token embeddings alignment for multi-sense words in a microscopic perspective without hurting the macroscopic performance on the bilingual lexicon induction task. As the research on cross-lingual token embedding learning is still in its early stage, I also discussed possible future work such as applying clustering algorithms on token embeddings to obtain sense-level multi-sense word representations.

Conclusion

This work investigates monolingual and cross-lingual word embedding learning methods. With the fast progress in the word embedding learning field, my research starts from monolingual context-independent word embedding learning models to the recent cross-lingual contextual word embedding learning models. This work addresses the question of how to improve the quality of monolingual word embeddings by combining the information of count-based methods and prediction-based methods together and how to map contextual word embeddings generated by pre-trained language representation models like ELMo or BERT across different languages, taking multi-sense words into account.

Precisely, for monolingual context-independent word embedding learning, I wanted to inject information extracted and calculated based on the statistics of word-word co-occurrences (count-based) in the training corpus into the skip-gram negative sampling architecture of the word2vec model (prediction-based).

In Chapter 3, I proposed corpus2graph, an open-source NLP-application-oriented Python package that generates a word co-occurrence network from a large corpus. It not only contains different built-in methods to preprocess words, analyze sentences, extract word pairs and define edge weights, but also supports user-customized functions. By using parallelization techniques, it can generate a large word co-occurrence network of the whole English Wikipedia data within hours. Thanks to its nodes-edges-weight three-level progressive calculation design, rebuilding networks with different configurations is even faster as it does not need to start all over again. This tool also works with other graph libraries such as igraph, NetworkX and graph-tool as a front end providing data to boost network generation speed.

In Chapter 4, I hypothesized that taking into account global, corpus-level information and generating a different noise distribution in negative sampling for each target word better satisfies the requirements of negative examples for each training word than the original frequency-based distribution. I proposed a new graph-based method to calculate noise distribution for negative sampling. By using a pre-computed word co-occurrence network, my noise distribution can be targeted to training words. I test this hypothesis through a set of experiments whose results show that my approach boosts the word analogy task by about

5% and improves the performance on word similarity tasks by about 1% compared to the skip-gram negative sampling baseline.

In Chapter 5, I explored the contextual word embeddings (token embeddings) of multi-sense words, argued that the current state-of-the-art method for cross-lingual token embedding learning cannot handle multi-sense words well and proposed my solutions by considering multi-sense word token embeddings as noise. Experiments show that my methods can improve the token embeddings alignment for multi-sense words in a microscopic perspective without hurting the macroscopic performance on the lexicon induction task.

Future work

I have already mentioned future work about corpus2graph and monolingual context-independent word embedding learning in Chapters 3 and 4.

- Possible extensions of corpus2graph would be to support more graph processing methods such as clustering algorithms and to support more NLP pre-processing tools such as spaCy[12].
- For graph-based monolingual word embedding learning, I injected word co-occurrence network information into negative sampling for word2vec, possible extension would be to use the network information for the context word selection for word2vec.

As contextual word embeddings have a richer semantic and syntactic representation than context-independent word embeddings (Peters et al., 2018a), I believe that the most promising direction to follow now is monolingual and cross-lingual contextual word embedding learning, as introduced in Chapter 5. Specifically, clustering algorithms on contextual word embeddings (token embeddings) would be an interesting direction to explore:

- For monolingual contextual word embedding learning, based on the fact that the token embeddings of a word are separated by its distinct senses. By applying clustering algorithms on token embeddings for each word, we could generate a classifier that detects multi-sense words based on the number of clusters in a word's token embeddings. Then these clusters could be used to link word embeddings to word sense embeddings.
- For cross-lingual contextual word embedding learning, we could improve the averaging based word anchor generation method with clustering algorithms. I showed the problems of assigning only one anchor for multi-sense words in Section 5.3. If we can

[12]https://spacy.io

obtain token embedding clusters for each multi-sense word and then generate averaging based anchors for them, there will be no frequency or sense bias on multi-sense word anchors. The new word sense level anchors will be a good replacement to the previous ones for the MUSE model. For supervised methods using word-level alignments, the multi-sense word classifier could also be used to generate a list of multi-sense words in an automatic way.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., and Soroa, A. (2009). A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Aldarmaki, H. and Diab, M. (2019). Context-aware cross-lingual mapping. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3906–3911, Minneapolis, Minnesota. Association for Computational Linguistics.

Aldous, D. and Fill, J. A. (2002). Reversible Markov chains and random walks on graphs. Unfinished monograph, recompiled 2014, available at http://www.stat.berkeley.edu/~aldous/RWG/book.html.

Ammar, W., Mulcaire, G., Tsvetkov, Y., Lample, G., Dyer, C., and Smith, N. A. (2016). Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Anders, S., Ivan, V., Sebastian, R., and Manaal, F. (2019). *Cross-Lingual Word Embeddings*. Morgan |& Claypool Publishers, New York, NY, USA, 1st edition.

Andrew, G., Arora, R., Bilmes, J., and Livescu, K. (2013). Deep canonical correlation analysis. In *International Conference on Machine Learning*, pages 1247–1255.

AP, S. C., Lauly, S., Larochelle, H., Khapra, M., Ravindran, B., Raykar, V. C., and Saha, A. (2014). An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems*, pages 1853–1861.

Arora, S., Li, Y., Liang, Y., Ma, T., and Risteski, A. (2016). A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399.

Artetxe, M., Labaka, G., and Agirre, E. (2016). Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294.

Artetxe, M., Labaka, G., and Agirre, E. (2017). Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 451–462.

Artetxe, M., Labaka, G., and Agirre, E. (2018a). Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Artetxe, M., Labaka, G., and Agirre, E. (2018b). A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798. Association for Computational Linguistics.

Baroni, M., Bernardini, S., Ferraresi, A., and Zanchetta, E. (2009). The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.

Baroni, M., Dinu, G., and Kruszewski, G. (2014). Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247. Association for Computational Linguistics.

Baroni, M. and Lenci, A. (2010). Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36:673–721.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003a). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Bengio, Y., Senécal, J.-S., et al. (2003b). Quick training of probabilistic neural nets by importance sampling. In *AISTATS*, pages 1–9.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Bruni, E., Boleda, G., Baroni, M., and Tran, N.-K. (2012). Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

Camacho-Collados, J., Pilehvar, M. T., Collier, N., and Navigli, R. (2017). Semeval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 15–26.

Cancho, R. F. i. and Solé, R. V. (2001). The small world of human language. *Proceedings of the Royal Society of London B: Biological Sciences*, 268(1482):2261–2265.

Chandar, S., Khapra, M. M., Larochelle, H., and Ravindran, B. (2016). Correlational neural networks. *Neural computation*, 28(2):257–285.

Chen, Q., Zhu, X., Ling, Z., Wei, S., Jiang, H., and Inkpen, D. (2016). Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

Chen, W., Grangier, D., and Auli, M. (2015). Strategies for training large vocabulary neural language models. *arXiv preprint arXiv:1512.04906*.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

Conneau, A., Lample, G., Ranzato, M., Denoyer, L., and Jégou, H. (2017). Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Csardi, G. and Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal*, Complex Systems:1695.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dinu, G., Lazaridou, A., and Baroni, M. (2014). Improving zero-shot learning by mitigating the hubness problem. *arXiv preprint arXiv:1412.6568*.

Eclipse-Deeplearning4j-Development-Team (2014). Deeplearning4j: Open-source distributed deep learning for the jvm, apache software foundation license 2.0. Website. http://deeplearning4j.org.

Erkan, G. and Radev, D. R. (2004). Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Faruqui, M. and Dyer, C. (2014). Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471.

Ferret, O. (2004). Discovering word senses from a network of lexical cooccurrences. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., and Ruppin, E. (2001). Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Firth, J. R. (1957). *A synopsis of linguistic theory 1930-55.*, volume 1952-59. The Philological Society, Oxford.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Gouws, S., Bengio, Y., and Corrado, G. (2015). Bilbowa: Fast bilingual distributed representations without word alignments. In *International Conference on Machine Learning*, pages 748–756.

Gouws, S. and Søgaard, A. (2015). Simple task-specific bilingual word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1386–1390.

Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304.

Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA.

Haghighi, A., Liang, P., Berg-Kirkpatrick, T., and Klein, D. (2008). Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL-08: Hlt*, pages 771–779.

Halawi, G., Dror, G., Gabrilovich, E., and Koren, Y. (2012). Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 1406–1414, New York, NY, USA. ACM.

Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

Hartmann, M., Kementchedjhieva, Y., and Søgaard, A. (2018). Why is unsupervised alignment of english embeddings from different algorithms so hard? In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 582–586. Association for Computational Linguistics.

He, L., Lee, K., Lewis, M., and Zettlemoyer, L. (2017). Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483.

Hill, F., Reichart, R., and Korhonen, A. (2014). Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *CoRR*, abs/1408.3456.

Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. (2012). Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Huang, L., Cho, K., Zhang, B., Ji, H., and Knight, K. (2018). Multi-lingual common semantic space construction via cluster-consistent word embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 250–260. Association for Computational Linguistics.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2014). On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2016). Character-aware neural language models. In *AAAI*, pages 2741–2749.

Labeau, M. and Allauzen, A. (2017). An experimental analysis of noise-contrastive estimation: the noise distribution matters. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 15–20.

Lazaridou, A., Dinu, G., and Baroni, M. (2015). Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 270–280.

Lefever, E. and Hoste, V. (2009). Semeval-2010 task 3: Cross-lingual word sense disambiguation. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 82–87. Association for Computational Linguistics.

Levy, O. and Goldberg, Y. (2014a). Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 302–308.

Levy, O. and Goldberg, Y. (2014b). Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Levy, O., Goldberg, Y., and Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Levy, O., Søgaard, A., and Goldberg, Y. (2017). A strong baseline for learning cross-lingual word embeddings from sentence alignments. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 765–774.

Li, B., Liu, T., Zhao, Z., Tang, B., Drozd, A., Rogers, A., and Du, X. (2017). Investigating different syntactic context types and context representations for learning word embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2421–2431.

Liu, X., Shen, Y., Duh, K., and Gao, J. (2017). Stochastic answer networks for machine reading comprehension. *arXiv preprint arXiv:1712.03556*.

Lu, A., Wang, W., Bansal, M., Gimpel, K., and Livescu, K. (2015). Deep multilingual correlation for improved word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 250–256.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Bilingual word representations with monolingual quality in mind. In *Proceedings of NAACL-HLT*, pages 151–159.

Luong, T., Socher, R., and Manning, C. (2013). Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Meng, X., Bradley, J. K., Yavuz, B., Sparks, E. R., Venkataraman, S., Liu, D., Freeman, J., Tsai, D. B., Amde, M., Owen, S., Xin, D., Xin, R., Franklin, M. J., Zadeh, R., Zaharia, M., and Talwalkar, A. (2015). Mllib: Machine learning in apache spark. *CoRR*, abs/1505.06807.

Mihalcea, R. and Tarau, P. (2004). Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Mihalcea, R. F. and Radev, D. R. (2011). *Graph-based Natural Language Processing and Information Retrieval*. Cambridge University Press, New York, NY, USA, 1st edition.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013b). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Mikolov, T., Le, Q. V., and Sutskever, I. (2013c). Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013d). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013e). Distributed representations of words and phrases and their compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013f). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Miller, G. A. and Charles, W. G. (1991). Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

Mitchell, J. and Lapata, M. (2010). Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429.

Mnih, A. and Teh, Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.

Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer.

Moro, A., Raganato, A., and Navigli, R. (2014). Entity linking meets word sense disambiguation: a unified approach. *TACL*, 2:231–244.

Nissim, M., van Noord, R., and van der Goot, R. (2019). Fair is better than sensational: Man is to doctor as woman is to doctor. *arXiv preprint arXiv:1905.09866v1*.

Peixoto, T. P. (2014). The graph-tool python library. *figshare*.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018a). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018b). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Pradhan, S., Moschitti, A., Xue, N., Uryupina, O., and Zhang, Y. (2012). Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40. Association for Computational Linguistics.

Radinsky, K., Agichtein, E., Gabrilovich, E., and Markovitch, S. (2011). A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM.

Rapp, R. (2003). Word sense discovery based on sense descriptor dissimilarity. In *Proceedings of the ninth machine translation summit*, pages 315–322.

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Rocchio, J. J. (1971). The smart retrieval system: Experiments in automatic document processing. *Relevance feedback in information retrieval*, pages 313–323.

Rogers, A., Hosur Ananthakrishna, S., and Rumshisky, A. (2018). What's in your embedding, and how it predicts task performance. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2690–2703. Association for Computational Linguistics.

Rong, X. (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

Rousseau, F. and Vazirgiannis, M. (2013). Graph-of-word and tw-idf: New approach to ad hoc ir. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 59–68, New York, NY, USA. ACM.

Rubenstein, H. and Goodenough, J. B. (1965). Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Ruder, S. (2016). On word embeddings - part 2: Approximating the softmax. http://ruder.io/word-embeddings-softmax. Last accessed 11 May 2018.

Ruder, S., Vulić, I., and Søgaard, A. (2017). A survey of cross-lingual word embedding models. *arXiv preprint arXiv:1706.04902*.

Salle, A., Villavicencio, A., and Idiart, M. (2016). Matrix factorization using window sampling and negative sampling for improved word representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 419–424, Berlin, Germany. Association for Computational Linguistics.

Salton, G. (1962). Some experiments in the generation of word and document associations. In *Proceedings of the December 4-6, 1962, fall joint computer conference*, pages 234–250. ACM.

Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Schuster, T., Ram, O., Barzilay, R., and Globerson, A. (2019). Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1599–1613, Minneapolis, Minnesota. Association for Computational Linguistics.

Schütze, H. (1993). Word space. In *Advances in neural information processing systems*, pages 895–902.

Smith, S. L., Turban, D. H., Hamblin, S., and Hammerla, N. Y. (2017). Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Søgaard, A., Agić, Ž., Alonso, H. M., Plank, B., Bohnet, B., and Johannsen, A. (2015). Inverted indexing for cross-lingual nlp. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1713–1722.

Steiger, J. H. (1980). Tests for comparing elements of a correlation matrix. *Psychological bulletin*, 87(2):245.

Tsvetkov, Y., Faruqui, M., Ling, W., Lample, G., and Dyer, C. (2015). Evaluation of word vector representations by subspace alignment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2049–2054.

Turney, P. D. (2006). Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

Turney, P. D. and Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.

Vulić, I. and Moens, M.-F. (2015). Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 719–725.

Vulić, I. and Moens, M.-F. (2016). Bilingual distributed word representations from document-aligned comparable data. *Journal of Artificial Intelligence Research*, 55:953–994.

Walt, S. v. d., Colbert, S. C., and Varoquaux, G. (2011). The numpy array: A structure for efficient numerical computation. *Computing in Science and Engg.*, 13(2):22–30.

Xin, X., Yuan, F., He, X., and Jose, J. M. (2018). Batch is not heavy: Learning word representations from all samples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1853–1862, Melbourne, Australia. Association for Computational Linguistics.

Xing, C., Wang, D., Liu, C., and Lin, Y. (2015). Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Yeh, A. (2000). More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, COLING '00, pages 947–953, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yin, Z. and Shen, Y. (2018). On the dimensionality of word embedding. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 887–898. Curran Associates, Inc.

Zhang, M., Liu, Y., Luan, H., and Sun, M. (2017a). Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1934–1945.

Zhang, M., Peng, H., Liu, Y., Luan, H.-B., and Sun, M. (2017b). Bilingual lexicon induction from non-parallel data with minimal supervision. In *AAAI*, pages 3379–3385.

Zhang, Z., Yin, R., and Zweigenbaum, P. (2018). Efficient generation and processing of word co-occurrence networks using corpus2graph. In *Proceedings of TextGraphs-12: the Workshop on Graph-based Methods for Natural Language Processing*. Association for Computational Linguistics.

Zhang, Z. and Zweigenbaum, P. (2018). GNEG: Graph-based negative sampling for word2vec. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 566–671. Association for Computational Linguistics.

Zhu, X., Goldberg, A., Van Gael, J., and Andrzejewski, D. (2007). Improving diversity in ranking using absorbing random walks. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 97–104.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Explorations de plongements lexicaux : apprentissage de plongements à base de graphes et apprentissage de plongements contextuels multilingues

Mots clés : Vecteurs de mots, Traitement automatique des langues, Multilingue

Résumé : Les plongements lexicaux sont un composant standard des architectures modernes de traitement automatique des langues (TAL). Chaque fois qu'une avancée est obtenue dans l'apprentissage de plongements lexicaux, la grande majorité des tâches de traitement automatique des langues, telles que l'étiquetage morphosyntaxique, la reconnaissance d'entités nommées, la recherche de réponses à des questions, ou l'inférence textuelle, peuvent en bénéficier. Ce travail explore la question de l'amélioration de la qualité de plongements lexicaux monolingues appris par des modèles prédictifs et celle de la mise en correspondance entre langues de plongements lexicaux contextuels créés par des modèles préentraînés de représentation de la langue comme ELMo ou BERT.
Pour l'apprentissage de plongements lexicaux monolingues, je prends en compte des informations globales au corpus et génère une distribution de bruit différente pour l'échantillonnage d'exemples négatifs dans word2vec. Dans ce but, je précalcule des statistiques de cooccurrence entre mots avec corpus2graph, un paquet Python en source ouverte orienté vers les applications en TAL : il génère efficacement un graphe de cooccurrence à partir d'un grand corpus, et lui applique des algorithmes de graphes tels que les marches aléatoires. Pour la mise en correspondance translingue de plongements lexicaux, je relie les plongements lexicaux contextuels à des plongements de sens de mots. L'algorithme amélioré de création d'ancres que je propose étend également la portée des algorithmes de mise en correspondance de plongements lexicaux du cas non-contextuel au cas des plongements contextuels.

Title : Explorations in word embeddings: graph-based word embedding learning and cross-lingual contextual word embedding learning

Keywords : Word embeddings, NLP, Multilingual

Abstract: Word embeddings are a standard component of modern natural language processing architectures. Every time there is a breakthrough in word embedding learning, the vast majority of natural language processing tasks, such as POS-tagging, named entity recognition (NER), question answering, natural language inference, can benefit from it. This work addresses the question of how to improve the quality of monolingual word embeddings learned by prediction-based models and how to map contextual word embeddings generated by pre-trained language representation models like ELMo or BERT across different languages.
For monolingual word embedding learning, I take into account global, corpus-level information and generate a different noise distribution for negative sampling in word2vec. In this purpose I pre-compute word co-occurrence statistics with corpus2graph, an open-source NLP-application-oriented Python package that I developed: it efficiently generates a word co-occurrence network from a large corpus, and applies to it network algorithms such as random walks. For cross-lingual contextual word embedding mapping, I link contextual word embeddings to word sense embeddings. The improved anchor generation algorithm that I propose also expands the scope of word embedding mapping algorithms from context-independent to contextual word embeddings.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France