Combining Subword information and Language model for Information Retrieval

Jibril Frej — Philippe Mulhem — Didier Schwab— Jean-Pierre Chevallet

Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble, France*
** Institute of Engineering Univ. Grenoble Alpes*

RÉSUMÉ. *En recherche d'information, certains procédés sont utilisés pour améliorer les performances des modèles de langue. Lorsque l'on considère la sémantique des mots, il a été montré que les plongements de mots neuronaux capturent des similarités sémantiques entre les mots (Mikolov* et al.*, 2013). De telles représentations distribuées qui plongent les mots dans un espace vectoriel dense sont apprises de façon efficace sur de grandes collections. Récemment, elles ont été utilisées pour calculer les probabilités de traduction entre termes dans le cadre des modèles de langue neuronaux (Zuccon* et al.*, 2015) pour la recherche d'information afin de gérer le problème de la disparité des termes. Dans cet article, nous proposons d'utiliser de nouvelles représentations distribuées qui prennent en compte la structure interne des mots (Bojanowski* et al.*, 2016) dans le cadre des modèles de langue neuronaux.*

ABSTRACT. *Information Retrieval (IR) classically relies on several processes to improve performance of language modeling approaches. When considering semantic of words, Neural Word Embeddings (Mikolov* et al.*, 2013) have been shown to catch semantic similarities between words. Such Distributed Representations represent terms in a dense vector space are efficiently learned from large corpora. Lately, they have been used to compute the translation probabilities between terms in the Neural Translation Language Model (NTLM) (Zuccon* et al.*, 2015) framework for Information Retrieval in order to deal with the* vocabulary mismatch *issue. In this work, we propose to test this model with recent vectorial representations (Bojanowski* et al.*, 2016) that take into account the internal structure of words.*

MOTS-CLÉS : *Recherche d'information, Modèle de langue, Représentation Distribuée de Mots*

KEYWORDS: *Information Retrieval, Language Models, Distributed Word Representations.*

1. Introduction

Information Retrieval Systems (IRS) are computer assistants that help to retrieve digital documents, in which user is supposed to find relevant information for his task.

In most IRS, documents and queries are seen as simple bags of words, and the matching between queries and documents is based on statistical word distributions and term intersections. Though, these systems provide satisfaction to the users, as long as documents are large enough and queries are expressed using the same vocabulary as documents. However, when documents to be retrieved are very short and/or when there is a strong discrepancy between document and user vocabularies, IRS are facing the problem of *vocabulary mismatch*.

Several methods have been proposed to tackle the vocabulary mismatch issue such as query expansion that consist in completing the query with words that are semantically close to the query terms (Almasri *et al.*, 2016) and document expansion that consist in expanding the document with its neighborhood information (Tao *et al.*, 2006).

Another way to deal with *vocabulary mismatch* is to directly modify the matching function using Word Embeddings that are distributed representations[1] of words. As shown by (Zuccon *et al.*, 2015), Word Embeddings produced by the skipgram model (Mikolov *et al.*, 2013) can be used to capture semantic similarities between words and improve classical language models.

In this paper, we propose to test the Neural Translation Language Model (NLTM) proposed by (Zuccon *et al.*, 2015) with vectors produced by the subword model (Bojanowski *et al.*, 2016) that take into account subword information and that can associate a vector to any word and not just to the word in the training sequence. We first present the skipgram and the subword models recall the NTLM, then we recall the NTLM and finally we describe the implementation before presenting the results obtained.

2. Related work

2.1. *Latent Semantic Indexing*

One approach developed in 1990 to produce document and query vector representation is the Latent Semantic Analysis (LSI) (Deerwester *et al.*, 1990 ; Manning *et al.*, 2008). It is a method for indexing and retrieving documents based on a low-rank (denoted k) approximation of the term-document matrix $\mathbf{X}$ using it's Singular Value Decomposition (SVD) : $X = U\Sigma V^T$. The vector representations of the documents are computed $\vec{d_k} = \Sigma_k^{-1} U_k^T \vec{d}$. We can also map the vector representation of a query $\vec{q}$ or a term $\vec{t}$ into the "LSI's latent space" the same way as before : $\vec{q_k} = \Sigma_k^{-1} U_k^T \vec{q}$ and

1. vectors in $\mathbb{R}^n$

$\vec{t_k} = \Sigma_k^{-1} U_k^T \vec{t}$. The cosine similarity measure can be used to compare $\vec{d_k}$ and $\vec{q_k}$ in order to rank documents with respect to a query.

We use the cosine similarity measure between the document vector and the query vector to rank the documents with respect to a query : $RSV(q, d) = cos(\vec{q}, \vec{d})$.

One of the main issue with LSI is that it relies on SVD which is computationally expansive especially for huge datasets and can be hard to update when a new document appears.

2.2. *Latent Dirichlet Allocation*

LDA is a probabilistic generative model for discrete data such as text corpora (Blei *et al.*, 2003) that has been proposed in 2003. Within the frame of LDA, each document of a corpus is represented as a mixture of K latent topics, each of them is represented by a distribution over all the words of the vocabulary. This model assumes that a document d is generated the following way :

– First we choose K, the number of latent topics

– Then, the topic weight vector of document d θ, a K-dimensional Dirichlet variable is generated with respect to a parameter α : $\theta \sim Dir(\alpha)$

– For each term t of the considered document :

- Choose a topic $z \in \mathcal{Z}$ from the multinomial distribution $p(z = z_k|\theta) = \theta^k$ with $k \in \{1, ..., K\}$

- Given the topic z, choose a term t from the distribution $p(t = t_i|z = z_k, \beta) = \beta_{ik}$ with $i \in \{1, ..., V\}$

The parameters α and β are estimated with the EM algorithm. The reader can refer to the original LDA article (Blei *et al.*, 2003) for details about the computation of an approximation of the posterior distribution of the hidden variables θ and z given a document : $p(\theta, z|d, \alpha, \beta)$.

In the IR framework, LDA can be used to estimate the probability of a word in a document (Wei et Croft, 2006) in the language modelling framework.

Finally, the LDA also follows the bag-of-words assumption : the order of the terms in a document will not have any influence over it's associated distribution of topics. In the next subsection we will present the skip-gram model that has been proposed recently (Mikolov *et al.*, 2013) which is an efficient algorithm that produces vector representation of words without the bag-of-word assumption.

3. Word Embedding Models

3.1. *Skipgram Model*

The skipgram model follows the distributionnal hypothesis (Harris, 1954) by learning a distributed representation by trying to predict the context of a word given the word itself. Given a training corpus represented by a sequence of words $w_1, ..., w_T$, the objective of skipgram is to minimize the following negative log-likelihood :

$$\sum_{t=1}^{T} \left[\sum_{c \in \mathcal{C}_t} \ell(s(w_t, w_c)) + \sum_{n \in \mathcal{N}_{t,c}} \ell(-s(w_t, n)) \right] \quad [1]$$

Where $\mathcal{C}_t$ is the context of word w_t and is represented as the set of indices of words surrounding word w_t. ℓ is the logistic loss function : $x \rightarrow log(1 + e^{-x})$; $\mathcal{N}_{t,c}$ is a set of negative examples sampled from the vocabulary (Mikolov *et al.*, 2013) (Goldberg et Levy, 2014) and s is a score function that maps pairs of words in $\mathbb{R}$. In the skipgram framework, the score function s is simply the scalar product between the vectors associated to the word we are considering : $s(w_t, w_c) = \overrightarrow{w_t} \cdot \overrightarrow{w_c}$.

3.2. *Subword Model*

Instead of associating a single unique vector to each word of the training data, the subword model will learn representation for character n-grams. The words are represented by the sum of the vectors of the n-grams that compose it : $\overrightarrow{w} = \sum_{g_w \in \mathcal{G}_w} \overrightarrow{g_w}$ with $\mathcal{G}_w$ the set of character n-gram that are present in word w. The only difference between the skipgram and the subword models is the scoring function : $s(w_t, w_c) = \left(\sum_{g_{w_t} \in \mathcal{G}_{w_t}} \overrightarrow{g_{w_t}} \right) \cdot \left(\sum_{g_{w_c} \in \mathcal{G}_{w_c}} \overrightarrow{g_{w_c}} \right)$. This model allows word that do not appear in the training data to have an embeddings associated. This option of the model can be useful for rare words that do not appear in the training data enough times to have an embedding associated[2]. One of the main advantage of this method compared to the skipgram model is that it is able to associate an embedding to words that were not in the training data.

2. On the TREC collections about 60% of the words of the vocabulary appear less than 5 times in the collections, consequently they do not have an associated embedding

4. Neural Translation Language Model

In this part, we present the NTLM proposed by (Zuccon *et al.*, 2015). This Language Model (LM) ranks documents with respect to a query using the following Relevance Status Value (RSV) formula :

$$RSV(q,d) \underset{rank}{\simeq} log\left(p(q|\theta_d)\right) = \sum_{i=1}^{|q|} log\left(p(q_i|\theta_d)\right) \quad [2]$$

With q_i the i^{th} word of query q, $|q|$ the number of words of query q and θ_d the model associated to document d. The probability $p(q_i|\theta_d)$ of a query term q_i to be generated by a document model θ_d uses a Dirichlet smoothing, and integrates similarity between non-stemmed terms coming from Word Embeddings :

$$p(q_i|\theta_d) = \frac{|d|}{\mu + |d|} p_{cos}(q_i|\theta_d) + \frac{\mu}{\mu + |d|} p_{ml}(q_i|\theta_C) \quad [3]$$

With θ_C the model of the document collection C, $\mu \in \mathbb{R}^+$, the smoothing parameter and $p_{ml}(q_i|\theta_C)$ is estimated using the maximum likelihood (i.e., $c(q_i, C)/|C|$, with $c(q_i, C)$, the frequency of q_i in C and $|C|$ the number of words in the collection). $p_{cos}(q_i|\theta_d)$ corresponds to the probability that q_i has been produced from a translation of document d, defined as :

$$p_{cos}(q_i|\theta_d) = \sum_{u \in d} p_{cos}(q_i|u) p_{ml}(u|\theta_d) \quad [4]$$

With $p_{ml}(u|\theta_d) = c(q_i, d)/|d|$ and $p_{cos}(q_i|u)$ corresponds to the probability of translating word u into word q_i and is estimated using the cosine similarity between their associated embeddings $\vec{q_i}$ and $\vec{u}$:

$$p_{cos}(q_i|u) = \frac{cos(\vec{q_i}, \vec{u})}{\sum\limits_{u' \in V} cos(\vec{u'}, \vec{u})} \quad [5]$$

The NTLM approach fails when a query term and a document term are semantically close but one of them does not have an associated embedding and therefore their semantic similarity will not be taken into account. To tackle this issue, we propose to use the subword model that can associate an embedding to words that were not in the training data. Do to that, we simply replace the skipgram vectors by the subword vectors.

5. Implementation and data

We used the standard Porter algorithm (Porter, 2001) to perform stemming and SMART [3] stop-Words list to remove non relevant words. Besides, instead of using an already existing IRS such as Terrier, we developed our own IRS in C++ to easily add word embeddings to the classical models.

We evaluated the NTLM on 4 standard TREC collections : AP88-89 (disk 1 & 2) with topics 51-200, FT91-94 (disk 5)with topics 251-450, LA (disk 5) with topics 301-450 and WSJ87-92 (disk 1) with topics 51-200. The statistics of these collections are summed up in Table 1. The preprocessing of the collections consisted in removing non alphanumeric characters, deleting words that contained more than 3 identical successive characters and delete words than contained more than 4 digits (as in Terrier). We also replaced upper case letters by lower case. The MAP and P@5 were computed using trec_eval.

The smoothing parameter μ was set to 1000 for all datasets (Wei et Croft, 2006) and we also applied a threshold $\mathbf{T} = 0.7$ to filter out cosine similarities between words (Rekabsaz *et al.*, 2016)

Collection	**#Docs**	**Average document length**	**Vocabulary Size**	**Stemmed Vocabulary Size**
AP88-89	164 597	245.04	240 239	189 301
FT91-94	210 158	205.66	271 137	216 339
LA	131 896	243.86	235 534	180 982
WSJ87-92	173 252	226.46	211 990	162 576

Tableau 1 : **Statistics of the collections used**

For our experiments we explored the impact of the model and of the embedding training data on the quality of the retrieval system. We compared 3 different models : skipgram (denoted sk), subword (denoted sb-) and the subword model where we used the option to associate a vector to words that are not in the training vocabulary (denoted sb). The embeddings were either trained on the collection used for retrieval or on the English Wikipedia dump from the 1^{st} February 2018 [4] (denoted wiki-). We also combined the vectors trained on the retrieval collections and on the Wikipedia dump by concatenating them (denoted concat-) since combining embeddings has been proven to yield improvements on word similarity and other NLP tasks (Ghannay *et al.*, 2016).

When training the embeddings with the skipgram model, we choose the same parameters as (Rekabsaz *et al.*, 2016) : 300 dimensions, sub-sampling parameter equal

3. https://github.com/igorbrigadir/stopwords/blob/master/en/smart.txt
4. We also stemmed the collections before training the embeddings

to 10^{-5}, context window of 5, word count threshold of 5, 20 negative samples and 25 epoch of training (when training on Wikipedia we only performed 5 epoch since the collection is way bigger than TREC collections with more than 2 billion words). The same parameters where used for the subword model. All the other parameters of the model were kept to their default values (see the implementation of the skipgram [5] and the subword [6] models).

6. Results

As we can see on Table 2, with the exception of the LA dataset, none of the proposed embeddings significantly outperformed the Dirichlet Language Model baseline. We attempted to compare our results with those by (Zuccon *et al.*, 2015), unfortunately their baseline on the AP88-89 collection (with the same topics) has a MAP of 22.69, which is way below our baseline results on the AP88-89 collection since we obtained a MAP of 27.36. The same can be observed for the WSJ87-92 collection. Moreover our baseline also outperforms the best results obtained by (Zuccon *et al.*, 2015) using word embeddings. Further investigation is needed on the effect of our IRS and the one used by (Zuccon *et al.*, 2015) on the Language Models and on the use of word embeddings.

We also reported results when doing the experiments on non-stemmed collections on Table 3. The observation are the same as before, except for the LA collection, the models do not outperform significantly the Dirichlet baseline. However we can observe that when they are trained on the TREC collections the vectors produced by the subword model outperform skipgram vectors. These results are coherent with the work of (Bojanowski *et al.*, 2016) which suggest that the subword model performs better than the skipgram model when trained on small dataset (the TREC collections are only 40 million words compared to the 2 billion of Wikipedia). Also the complete subword model that associate a vector to out-of-training-vocabulary words does not perform better than the simple subword model. This suggest that taking into account rare words hurts the performance of the model. Also the concatenation of vectors does not seem to have an impact on the quality of the model, more refined methods such as the concatenation of vectors produced by different methods followed by dimension reduction methods (such as PCA) may lead to better results.

Overall if the vectors are trained on large enough collections, skipgram and the subword models produce vectors that lead to similar performance for the NTLM model.

5. https://github.com/dav/word2vec
6. https://github.com/facebookresearch/fastText

Method	AP88-89		FT91-94		LA		WSJ87-92	
	MAP	P@5	MAP	P@5	MAP	P@5	MAP	P@5
Dirichlet LM	27.36	45.73	23.31	32.60	20.99	28.53	24.72	46.80
sk	27.30	45.69	22.82	32.40	21.59	30.53	24.35	46.67
sb-	27.19	45.43	**23.41**	32.90	21.96	30.93*	24.23	46.40
sb	27.18	45.87	22.71	31.90	22.21	30.40	24.32	46.00
wiki-sk	27.41	45.47	23.20	**33.30**	22.23	30.13	24.43	**47.20**
wiki-sb-	27.50	45.20	22.85	32.80	21.72	29.87	24.46	**47.20**
wiki-sb	27.44	45.33	22.96	32.80	21.76	29.47	24.45	46.93
concat-sk	27.41	45.47	23.04	32.90	22.40*	30.93*	24.50	46.27
concat-sb-	**27.53**	45.47	23.13	32.90	**22.74***	**31.20***	**24.80**	46.53
concat-sb	27.50	**46.53**	22.87	32.20	22.63*	30.67	24.69	46.27

Tableau 2 : **MAP and P@5 of the different models using Porter stemmer.** Statistically significant differences with the Dirichlet baseline was computed using bilateral paired t-test and are denoted by a * (p-value < 0.05)

Method	AP88-89		FT91-94		LA		WSJ87-92	
	MAP	P@5	MAP	P@5	MAP	P@5	MAP	P@5
Dirichlet LM	**26.00**	**46.27**	21.47	30.90	20.67	29.60	23.48	44.93
sk	23.56	42.80	20.26	30.20	19.69	29.20	21.23	43.33
sb-	25.09	43.73	22.03	31.80	21.58	30.67	22.46	44.27
sb	24.59	43.87	21.73	31.30	20.82	30.00	22.07	43.73
wiki-sk	25.99	44.27	**22.51**	32.70	21.84	30.67	23.27	45.47
wiki-sb-	25.79	44.93	22.24	**33.10**	21.39	31.20	23.07	46.27
wiki-sb	25.64	44.53	22.48	32.70	21.34	31.07	23.01	**46.40**
concat-sk	24.47	43.60	21.16	30.60	20.60	30.00	22.31	44.40
concat-sb-	25.70	44.67	22.24	31.80	22.05	30.40	**23.59**	44.93
concat-sb	25.28	44.00	21.98	31.50	**22.18***	**31.60**	23.37	44.67

Tableau 3 : **MAP and P@5 of the different models without stemming.** Statistically significant differences with the Dirichlet baseline was computed using bilateral paired t-test and are denoted by a * (p-value < 0.05)

7. Conclusion

The goal of this work was to compare the skipgram and the subword models in the NTLM framework.

Our conclusion is four-fold : (1) The IRS we developed leads to very different results than the ones obtained by (Zuccon *et al.*, 2015) : our baseline outperforms their best results and the NTLM model does not significantly outperforms the Dirichlet baseline. (2) If the collection is large enough (few billion words) the embeddings produced by the skipgram and the subword models have a similar influence of the NTLM. (3) Assigning a vector to rare words hurts the performance of the NTLM

model on the collections we used during our experiments. (4) A simple concatenation of the vectors trained on the retrieval collection and trained on the Wikipedia dump does not improve results. Future work may include exploring the effect of our IRS and possible ways to combine the vectors to obtain better distributed representations for IR.

Remerciements

Ce travail a été effectué dans le cadre du projet Guimuteic financé par le Fonds Européen de Développement Régional (FEDER) et de la région Auvergne Rhône-Alpes.

8. Bibliographie

Almasri M., Berrut C., Chevallet J.-P., « A Comparison of Deep Learning Based Query Expansion with Pseudo-Relevance Feedback and Mutual Information », *Conférence ECIR*, vol. 42, Padoue, Italy, p. 369 - 715, March, 2016. 2

Blei D. M., Ng A. Y., Jordan M. I., « Latent Dirichlet Allocation », *J. Mach. Learn. Res.*, vol. 3, p. 993-1022, March, 2003. 3

Bojanowski P., Grave E., Joulin A., Mikolov T., « Enriching Word Vectors with Subword Information », *CoRR*, 2016. 2, 7

Deerwester S., Dumais S. T., Furnas G. W., Landauer T. K., Harshman R., « Indexing by latent semantic analysis », *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, vol. 41, n° 6, p. 391-407, 1990. 2

Ghannay S., Favre B., Estève Y., Camelin N., « Word embedding evaluation and combination », *10th edition of the Language Resources and Evaluation Conference (LREC 2016)*, Portorož, Slovenia, 2016. 6

Goldberg Y., Levy O., « word2vec Explained : deriving Mikolov et al.'s negative-sampling word-embedding method », *CoRR*, 2014. 4

Harris Z. S., « Distributional structure », *Word*, vol. 10, n° 2-3, p. 146-162, 1954. 4

Manning C. D., Raghavan P., Schütze H., *Introduction to Information Retrieval*, Cambridge University Press, New York, NY, USA, 2008. 2

Mikolov T., Sutskever I., Chen K., Corrado G., Dean J., « Distributed Representations of Words and Phrases and Their Compositionality », *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, Curran Associates Inc., USA, p. 3111-3119, 2013. 2, 3, 4

Porter M. F., « Snowball : A language for stemming algorithms », 2001. 6

Rekabsaz N., Lupu M., Hanbury A., Zuccon G., « Generalizing Translation Models in the Probabilistic Relevance Framework », *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, ACM, New York, NY, USA, p. 711-720, 2016. 6

Tao T., Wang X., Mei Q., Zhai C., « Language Model Information Retrieval with Document Expansion », *Proceedings of the Main Conference on Human Language Technology Confe-*

rence of the North American Chapter of the Association of Computational Linguistics, HLT-NAACL '06, Association for Computational Linguistics, Stroudsburg, PA, USA, p. 407-414, 2006. 2

Wei X., Croft W. B., « LDA-based Document Models for Ad-hoc Retrieval », *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, ACM, New York, NY, USA, p. 178-185, 2006. 3, 6

Zuccon G., Koopman B., Bruza P., Azzopardi L., « Integrating and Evaluating Neural Word Embeddings in Information Retrieval », *Proceedings of the 20th Australasian Document Computing Symposium*, ADCS '15, ACM, New York, NY, USA, p. 12 :1-12 :8, 2015. 2, 5, 7, 8