Toward Word Embedding for Personalized Information Retrieval
Nawal Ould Amer, Philippe Mulhem, Mathias Géry

To cite this version:

HAL Id: ujm-01377080
https://hal-ujm.archives-ouvertes.fr/ujm-01377080
Submitted on 6 Oct 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
ABSTRACT
This paper presents preliminary works on using Word Embedding (word2vec) for query expansion in the context of Personalized Information Retrieval. Traditionally, word embeddings are learned on a general corpus, like Wikipedia. In this work we try to personalize the word embeddings learning, by achieving the learning on the user’s profile. The word embeddings are then in the same context than the user interests. Our proposal is evaluated on the CLEF Social Book Search 2016 collection. The results obtained show that some efforts should be made in the way to apply Word Embedding in the context of Personalized Information Retrieval.

CCS Concepts
• Information systems → Personalization;

Keywords
Word Embedding, word2vec, Personalization, Social Book Search, Query expansion

1. INTRODUCTION
Recent works investigate the use of Word Embedding for enhancing IR effectiveness [1, 3, 8] or classification [5]. Word Embedding [7] is the generic name of a set of NLP-related learning techniques that seek to embed representations of words, leading to a richer representation: words are represented as vectors of more elementary components or features. Similarly to Latent Semantic Analysis (LSA) [2], Word Embedding maps the words to low-dimensional (w.r.t. vocabulary size) vectors of real numbers. For example, two vectors \( t_0 \) and \( t_1 \), corresponding to the words \( t_0 \) and \( t_1 \), are close in a N-dimensional space if they have similar contexts and vice-versa, i.e. if the contexts in turn have similar words [4]. In this vector space of embedded words, the cosine similarity measure is classically used to identify words occurring in similar contexts. In addition, the arithmetic operations between vectors reflect a type of semantic-composition, e.g. \( \text{bass} + \text{guitar} = \text{bass guitar} \) [10].

In this paper, we present an approach using Word Embedding for Personalized Information Retrieval. The goal of our works is to provide clues about the following questions:

- Can Word Embedding be used for query expansion in the context of social collection?
- Can Word Embedding be used to personalize query expansion?

The first question is motivated by our participation in the CLEF Social Book Search task in 2016 (similar to the Social Book Search task in 2015 [6]). Our concern was related to the fact that the topics provided contain non-topical terms that may impact the usage of Word Embedding.

The second question tackles more specifically the usage of Word Embedding when the learning is based on the user’s profile in order to select personalized words for query expansion. The idea is to select words that occur in the same context as the terms of the query. We compare then Word Embedding learned on the whole collection of Social Book Search, called the Non Personalized Query Expansion, versus Word Embedding learned on the user’s profiles, called the Personalized Query Expansion.

The paper is organized as follows: Section 2 presents the proposed approach. The experiments on the official CLEF Social Book Search collection are presented and discussed in Section 3, and the results are commented in Section 4. We conclude this work in Section 5.

2. PERSONALIZED QUERY EXPANSION

2.1 User Modeling
In the context of Social Book Search, each user is represented by his catalog (i.e. set of books) and other information such as tags and the ratings that he assigns to the books [6]. All these information describe the interests of the user.

We represent a user \( u \) as one document \( d_u \) which is the concatenation of all the documents present in his catalog.
The profile of \( u \), noted \( p_u \), is then represented by the set of words in \( d_u \):

\[
p_u = \{ w_1, w_2, w_3, ..., w_n \} \quad (1)
\]

2.2 Term Filtering

As stated before, \textit{Word Embedding} can potentially be helpful in the selection of terms related to the query. Usually, embedded terms are used for query expansion, as extensions of one of the query term. Despite the effectiveness of \textit{Word Embedding} to select embedded terms, it could happen that their use for query expansion decreases the effectiveness of the system.

In fact, if an extended term is a noisy term (because of its ambiguity for instance, or because it is not a topical term of the query), then the set of its resulting word embeddings will increase the non-topical noise in the extended query. For example, in the queries (taken from the topics of Social Book Search 2016) “Help! I Need more books”, “New releases from authors that literally make you swoon...” or “Favorite Christmas Books to read to young children”, the majority of these words are not useful for expansion, like “new, good, recommend, make, you, etc.”. Therefore, these words need to be filtered out of the queries before the expansion process. We chose to remove all the adjectives words from the queries. To do that, we use an English stop-adjective list in addition to the standard English stop-list.

Then, from a user query \( q = \{ t_1, t_2, ..., t_m \} \), we note \( q_f = \{ t_1, t_2, ..., t_n \} \) the filtered query.

2.3 Word Embedding Selection

Once we have a filtered query \( q_f \) as described above (cf. subsection 2.2), we select the top-k word embeddings to be used as extensions. This selection is achieved in three steps for each term \( t \) of the filtered query \( q_f \):

i) Building a set of word embeddings for \( t \) using the cosine similarity between \( t \) and all words in the training corpus;

ii) Filtering out from the set of word embeddings of i) the terms that have the same stem than \( t \) using English Porter Stemmer, in order to avoid overemphasizing the variations of \( t \);

iii) Selecting the top-k word embeddings of \( t \) from the filtered set of word embeddings of ii).

Then, the output of the selection of word embeddings is:

\[
\text{WordEmbedding}_{\text{word2vec}}(q_f) = \left\{ \begin{array}{c}
\text{em}_{t_{11}}, \text{em}_{t_{12}}, ..., \text{em}_{t_{1k}}, \\
\text{em}_{t_{21}}, \text{em}_{t_{22}}, ..., \text{em}_{t_{2k}}, \\
... \\
\text{em}_{t_{m1}}, \text{em}_{t_{m2}}, ..., \text{em}_{t_{mk}} \end{array} \right\} \quad (2)
\]

Where \( \text{WordEmbedding}_{\text{word2vec}}(q_f) \) denotes the function that returns a set of word embedding for a given filtered query \( q_f \), and \( \text{em}_{t_{ij}} \) denotes the \( j^{th} \) element of the top-k word embeddings of \( t_i \).

2.4 Ranking Model

The final expanded query \( q_{\text{new}} \) is the union of the original user query \( q \) and the word embeddings set as follow:

\[
q_{\text{new}} = q \cup \text{WordEmbedding}_{\text{word2vec}}(q_f) \quad (3)
\]

The score for each document \( d \) according to the expanded query \( q_{\text{new}} \) and the user \( u \) is computed according to a classical Language Model with Dirichlet smoothing.

3. EXPERIMENTS

3.1 Dataset

Our experiments are conducted on Social Book Search dataset [6].

- Documents: The documents collection consists of 2.8 millions of books descriptions with meta-data from Amazon and LibraryThing. Each document is represented by book-title, author, publisher, publication year, library classification codes and user-generated content in the form of user ratings and reviews.

- Users: The collection provides profiles of 120 users. Each user is described by his catalog (i.e. a set of books), tags, and rating.

- Queries: The collection contains 120 user queries. Due to the nature of the queries, we chose 28 queries for our experiments: these queries have topical content (i.e. a classical IR system has then chances to get relevant results) and the profile of the query issuer is not empty.

3.2 Learning of Word Embedding

Here, we describe the process of learning word vectors (see section 1). We use two training sets.

- The first one is called \textit{The Non Personalized Corpus}: the train is built on the whole Social Book Search Corpus.

- The second one is called \textit{The Personalized Corpus}: we build a personalized train corpus for each user.

The training process is the same for the two corpora:

1. \textit{Non Personalized Corpus train process:}

We train \textit{word2vec} [7] on the Social Book Search corpus. \textit{word2vec} represents each word \( w \) of the training set as a vector of features, where this vector is supposed to capture the contexts in which \( w \) appears. Therefore, we chose the Social Book Search corpus to be the training set, as the training set is expected to be consistent with the data on which we want to test. To construct the training set from the corpus, we simply concatenate the content of all the documents, without any particular pre-processing such as stemming or stop-words removal, except some text normalization and cleaning operations such as lower-case normalization, removing HTML tags, etc. The concatenated document is the input training set of \textit{word2vec}.

The size of the training set is \( \sim 12.5GB \); it contains 2.3 billions words, and the vocabulary size is about 600K.

The training parameters of \textit{word2vec} are set as follows:

- continuous bag of words model instead of the skip-grain (\textit{word2vec} options: cbow=1);  

\(^1\)http://social-book-search.humanities.uva.nl/#!/suggestion
2. Personalized Corpus train process:
   For this corpus, each user is described as a document $d_u$ (containing the documents belonging to his catalog, cf. Section 2.1) and we train word2vec on each user document $d_u$ using the same process than the Non Personalized Corpus train process.

3.3 Parameters and Tested Configurations

The ranking model is achieved using the Language Model with Dirichlet smoothing. All documents are retrieved using Terrier search engine [9] with $\mu = 50$.

We compare the following variations:

1. Query filtering: the query is filtered or not, by removing the stop-word and the adjectives as stated in section 2.2;
2. Query expansion: the query is expanded or not using the Word Embedding function;
3. Personalization: the query is personalized (using the Personalized Corpus) or not (using the Non Personalized Corpus).

The tested configurations are presented in Table 3.3.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Query filtering</th>
<th>Expansion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conf₁ (baseline)</td>
<td>Original (q)</td>
<td>-</td>
</tr>
<tr>
<td>Conf₂</td>
<td>Filtered (q_f)</td>
<td>-</td>
</tr>
<tr>
<td>Conf₃</td>
<td>Filtered (q_f)</td>
<td>Non Personalized</td>
</tr>
<tr>
<td>Conf₄</td>
<td>Filtered (q_f)</td>
<td>Personalized</td>
</tr>
<tr>
<td>Conf₅</td>
<td>Original (q)</td>
<td>Non Personalized</td>
</tr>
<tr>
<td>Conf₆</td>
<td>Original (q)</td>
<td>Personalized</td>
</tr>
</tbody>
</table>

Table 1: Tested configurations

4. RESULTS

In this section, we present and comment the results of the above configurations. All of these configurations lead to quite low MAP values, but this is consistent with the official CLEF Social Book Search results.

4.1 Query Filtering

Table 4.1 reports the Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), and precision at 10 documents (P@10) evaluation measures obtained with the configuration Conf₁ (without query filtering and without query expansion) and the configuration Conf₂ (with query filtering and without query expansion). As we can see, filtering the query terms with the stop-adjective list improves the MAP values, but surprisingly nor the MRR neither the P@10. The choice of filtering the adjectives may be not the most effective way to deal with noisy terms.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>MAP</th>
<th>MRR</th>
<th>P@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conf₁ (q, No QE)</td>
<td>0.0266</td>
<td>0.1478</td>
<td>0.0464</td>
</tr>
<tr>
<td>Conf₂ (q_f, No QE)</td>
<td>0.0309</td>
<td>0.1436</td>
<td>0.0393</td>
</tr>
</tbody>
</table>

Table 2: With vs without query filtering

4.2 Personalized Query Expansion (with filtering)

Figure 1 reports the MAP effectiveness over the number of word embeddings. As we can see, in most of the cases the non personalized approach outperforms the personalized approach. The personalized approach shows a better result only when the word embeddings are limited to the top-2 terms. We also remark that both the two approaches underperform the filtered non expanded approach Conf₂, which shows that inadequate words are added. For Conf₃, we see that adding more that 8 terms lead to better results, which is not really the case to the personalized configuration Conf₄ where the MAP evolution is quite flat. In this case, no added term seems to play a positive role.

![Figure 1: Non Personalized vs Personalized Query Expansion (with filtering)](image)

4.3 Personalized Query Expansion (without filtering)

The figure 2 reports the MAP value evolution over the number of word embeddings for the configurations Conf₅ (non personalized query expansion, without filtering) and the Conf₆ (personalized query expansion, without filtering). As we can see, in most of the cases, the non personalized approach outperforms the personalized one. However, these two curves in figure 2 behave quite differently: the personalized approach is better only when adding the top-1 and top-2 words, leading to consider that the first two personalized terms are interesting, whereas the non personalized expansion behaves better and better as the number of terms increases, leading to consider that “good” terms (according to the user) appear in the non personalized case.

4.4 Discussion

The figure 3 reports the MAP value over the number of word embeddings for the six configurations. As we can see, the results of the configurations: 1) Conf₃ and Conf₅, and 2) Conf₄ and Conf₆ are quite similar.

We observe that the configurations with query expansion
Figure 2: Personalized Query Expansion (without filtering)

Figure 3: All Configurations together

As presented above, in most of the cases, the expanded approach fails to improve results. We may explain these results by the quality of the corpus. In fact, the documents collection describes the reviews of users for books. Therefore, it’s still difficult to extract the context of terms and select the similar terms in the similar context. The personalized Word Embedding fails to improve the results comparing to any configurations. We first can explain the results by the same quality problem than for the documents collection. In fact, the user is represented by the documents that appear in his catalog. So, these documents present the reviews of the user about the books. Secondly, the learning of word2vec is effective if a large amount of data is available. However, the users’ profiles correspond to short documents. Therefore, the amount of learning data may not reach the limit under which no convergence is possible for word2vec.

5. CONCLUSION

The focus of this paper was to study the integration of word embeddings for the Social Book Suggestion task of CLEF 2016, according to non-personalized and to personalized query expansion.

We found that the nature of the queries poses a great challenge to an effective use of Word Embedding in this context. Future works may help to better understand this behavior. The second point is related to the fact that the personalization using word embeddings did not lead to good results. The reasons could be multiple: the quality of the description of the user’s profiles, the lack of data that we get to describe the profile which does not allow word embeddings to be used effectively. Here also, future works may help to find solutions to these problems. Especially, is it possible to counteract this limitation by adding other inputs (user neighbors for instance)? This question is largely open for now.

Acknowledgements. This work is supported by the ReSPIr project of the région Auvergne Rhône-Alpes.

6. REFERENCES