Word Embedding for Social Book Suggestion

Nawal Ould-Amer[1], Philippe Mulhem[1], Mathias Géry[2], and Karam Abdulahhad[1]

[1] Univ. Grenoble Alpes, LIG, F-38000 Grenoble, France
CNRS, LIG, F-38000 Grenoble, France
[2] Université de Lyon, F-42023, Saint-Étienne, France,
CNRS, UMR 5516, Laboratoire Hubert Curien, F-42000, Saint-Étienne, France
Université de Saint-Étienne, Jean-Monnet, F-42000, Saint-Étienne, France
{Nawal.Ould-Amer, Philippe.Mulhem, Karam.Abdulahhad}@imag.fr,
mathias.gery@univ-st-etienne.fr

Abstract. This paper presents the joint work of the Universities of Grenoble and Saint-Étienne at CLEF 2016 Social Book Search Suggestion Track. The approaches studied are based on personalization, considering the user's profile in the ranking process. The profile is filtered using *Word Embedding*, by proposing several ways to handle the generated relationships between terms. We find that tackling the problem of "non-topical" only queries is a great challenge in this case. The official results show that *Word Embedding* methods are able to improve results in the SBS case.

Keywords: social network, word embedding, probabilistic retrieval, profile selection

1 Introduction

This paper presents the joint participation of the MRIM group from the Laboratoire d'Informatique de Grenoble and the Laboratoire Hubert Curien de Saint-Étienne at CLEF 2016 Social Book Search (SBS) Suggestion Track. The goal of this track is to evaluate book retrieval approaches using the metadata of the book and user-generated content [6]. Considering our previous participation to this track in 2015 [9], our finding was that taking into account the whole user profile is not adequate. Hence, we believe that a selection of an adequate set of terms related both to the user's profile and to his query could enhance the results. Such selection of terms is then matched against the documents, and fused with the query/document matching values. The work described here relies on *Word Embedding* (word2vec [7]) as a mean to help to select those terms.

Recent works investigate the use of *Word Embedding* for enhancing IR effectiveness [1, 3, 8] or classification [5]. *Word Embedding* techniques seek to embed representations of words. For example, two vectors $\overrightarrow{t_0}$ and $\overrightarrow{t_1}$, corresponding to the words t_0 and t_1, are close in a space of N dimensions 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 the work described here, we used this approach to select words from the user's profile that occur in the same context as the query terms. These selected words represent a user sub-profile according to the context of his query.

More precisely, we investigate here several ways to filter these terms and their impact on the quality of the results. In a way to determine the impact of such term filtering, we also study the use of non-personalized word embeddings for the query terms, i.e., without using the user's profile.

The rest of the paper is organized as follows: Section 2 presents the proposed approach and its four steps. The experiments and the official results are presented in Section 3. We conclude this work in Section 4.

2 Proposed approach

2.1 Global Process

The task described here aims to suggest books according to a user's query and to his profile. It consists of four steps:

1. Learning of the *Word Embedding*: this step describes the learning of a set of word embeddings using word2vec [7] and the features applied in the training process;
2. User Model: this step describes the way user profile is modeled;
3. Profile Query-Filtering: this step describes the building process for the user sub-profile based on his query;
4. Ranking Model: this step depicts the ranking model using a Profile Query-Filtering.

2.2 Notations

We present now the notations used in this paper:

– $u \in U$: a user u from the whole set of users U;
– $q = \{t_1, t_2, ..., t_m\}$: the original query of a user u.
– $q_f = \{t_1, t_2, ..., t_o\}$: the filtered query of a user u.
– $P(u) = \{t_1 : w_1, t_2 : w_2, ..., t_n : w_n\}$: the profile of a user u, composed of couples <term : weight>.
– $Terms(u) = \{t_1, t_2, ..., t_n\}$: a set of user terms belonging to the user profile $P(u)$.
– $WordEmbedding_{w2v}(q)$: a set of word embeddings (i.e. output of the section 2.3).
– $P_{Filtered}(q, u)$: a sub-profile of the user u according to the user query q.
– $P_{Weighted\_filtered}(q, u)$: a weighted sub-profile of the user u according to the user query q.

2.3 Learning of Word Embeddings

Word Embedding is the generic name of a set of NLP-related learning techniques to map words to low-dimensional (w.r.t. vocabulary size) vectors of real numbers. Word Embedding [7], which falls in the same category as Latent Semantic Analysis (LSA) [2], enables us to deal with a richer representation of words, namely words are represented as vectors of more elementary components or features. The similarity between vectors is supposed to be a good indicator to the '*closeness*' between the respective words [11]. In addition, the arithmetic operations between vectors reflect a type of semantic-composition, e.g. *bass + guitar = bass guitar* [11].

In this section, we describe the process of learning word-vectors. To this end, we train *word2vec* [7] on the Social Book Search corpus. *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, where the training set is expected to be consistent with the data on which we want to test.

To construct the training set from the Social Book Search corpus, we simply concatenate the contents of all documents in one document, 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 obtained document is the input training set of *word2vec*. The size of the training set is $\sim 12.5GB$, where it contains 2.3 billions words, and the vocabulary size is about $600K$. The training parameters of *word2vec* are set as follows:

- we used the continuous bag of words model instead of the skip-gram (word2vec options: cbow=1);
- the output vectors size or the number of features of resulting word-vectors is set to 500 (word2vec options: size=500);
- the width of the word-context window is set to 8 (word2vec options: window=8);
- the number of negative samples is set to 25 (word2vec options: negative=25).

For example, based on the output of *word2vec*, the 10 most similar (using cosine similarity) words of "*novel*" in the Social Book Search corpus are: story, novella, thriller, tale, storyline, masterpiece, plot, debut, engrossing, and genre.

2.4 User Modeling

We model each user u by a set of terms that describe his interests. These terms are the tags that a user u used to describe or annotate the documents (*books* present in his catalogue). Each user u is represented by his profile $P(u)$ as a vector over the set of tags as follow:

$$P(u) = \{t_1 : w_1, t_2 : w_2, ..., t_n : w_n\} \tag{1}$$

Where t_i is a term and w_i a weight of this term, corresponding to his importance in the user profile. This weight w_i is the *tf* of t_i: the number of times the user u used this tag to annotate books.

2.5 Profile Query-Filtering

2.5.1 Term Filtering

As stated before, *Word Embedding* can potentially be helpful in the selection of terms related to the query. Despite the effectiveness of *Word Embedding* to select embedded terms, it could happen that they decrease the effectiveness of the system. Usually, embedded terms are used as extensions of a term, as query expansion. 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 noise in the extended query. For example, in the queries (taken from the topics of Social Book Search 2016) *"Help! i Need more books"*, *"Can you recommend a good thesaurus?"*, *"New releases from authors that literally make you swoon..."* or *"Favorite Christmas Books to read to young children"*, the majority of words are not useful for expansion, like "new, good, recommend, make, you ...". Therefore, we need to filter the words of the queries before the expansion process. We chose to remove from the words to be expanded all the words that are adjectives. To do that, we use an English stop-adjective list in addition to the standard English stop-word. Then, this process generates, from initial query q, a filtered query $q_f = \{t_1, t_2, ..., t_o\}$ that is expected to contain more relevant topical elements as output.

2.5.2 *Word Embedding* Selection

From each of the terms in the filtered query q_f obtained by the previous step 2.5.1, we then select their set of word embeddings. For each term as an input, we obtain the top-k most related terms as an output. The metric used is the cosine similarity between the query term and all words in the training corpus. From these top-k terms, we do not want to over-emphasize variations of the query term. So, we filter out from the top-k terms the ones that have the same stem as the initial query term. Such stems are generated using the well know Porter stemmer for English.

The expanded query generated as output of this step is:

$$WordEmbedding_{w2v}(q_f) = \left\{ \begin{array}{l} emt_{11}, emt_{12}, ..., emt_{1k}, \\ emt_{21}, emt_{22}, ..., emt_{2k}, \\ ... \\ emt_{o1}, emt_{o2}, ..., emt_{ok} \end{array} \right\} \quad (2)$$

Where $WordEmbedding_{w2v}(q_f)$ denotes the function that returns a set of word embedding for a given filtered query q_f, and em_t_{ij} denotes the j^{th} element of the top-k word embeddings of t_i.

2.5.3 Generation of user's sub-profile

The goal of our proposal is to construct a sub-profile using a set of word embeddings based on a user query. Our approach here is to apply a naive method based on the **Intersection** between a) the query word embeddings set and b) the user profile, as follows:

$$P_{Filtered}(q_f, u) = WordEmbedding_{w2v}(q_f) \cap Terms(u) \tag{3}$$

We propose two variations of such a user query sub-profile:

– **Non-Weighted Sub-Profile** ($P_{Filtered}$): this user sub-profile is represented by only the terms without weights, then all terms in the query have a same weight (i.e. corresponds to $P_{Filtered}(q_f, u)$ defined in Eq. 3).
– **Weighted Sub-Profile** ($P_{Weighted\_filtered}$): for this user sub-profile, the weight for each term in the profile is the cosine between of the embeddings of the filtered term and its respective query term.

2.6 Ranking

In this part, we present the ranking used to suggest books to the user according to his query and to his profile. We take into account two elements: if the document is related to the user query, and if the document is related to the user profile, as follows:

$$Score(q_f, u, d) = \alpha\ Score(q_f, d) + (1 - \alpha)\ Score(u, d) \tag{4}$$

Where $Score(q_f, d)$ is the matching score between the filtered query q_f and the document d, $Score(u, d)$ is the matching score between the user u and the document d, and $\alpha \in [0, 1]$ determines the relative importance of each part of formula.

The $Score(q_f, d)$ is a fusion of four parts:

$$\begin{aligned} Score(q_f, d) = \beta\ Score_{BM25}(q_f, d) + \gamma\ Score_{LM}(q_f, d) \\ + \lambda\ Score_{BM25}(WordEmbedding_{w2v}(q_f), d) \\ + \delta\ Score_{LM}(WordEmbedding_{w2v}(q_f), d) \end{aligned} \tag{5}$$

Where $Score_{BM25}(q_f, d)$ (resp. $Score_{LM}(q_f, d)$) denotes a matching using the classical BM25 model (resp. language model), and β, γ, λ and δ are parameters that determine the importance of each part of the document overall score ($\beta + \gamma + \lambda + \delta = 1$).

The function $Score(u, d)$ in equation (4) is computed as a classical matching function (BM25 or Language Model) between the document d and the filtered user's profile (weighted: $P_{Weighted\_filtered}$, or not weighted: $P_{Filtered}$) of u.

3 Experimental Results

3.1 Parameters and post-processing

For all the runs described below, for the BM25 runs the parameters are the defaults of Terrier [10], except for one parameter $b = 0.5$, according to previous experiments on SBS 2015. For the Language Models with Dirichlet smoothing, we use the default Terrier parameter $\mu = 2500$.

For the re-ranking post process, we used the same protocol used in [9]. The documents that belong to the user's catalogue are removed for the result. This process is applied for each run.

3.2 Submitted runs

We submitted the 6 following runs:

1. RUN1: This run is the non-personalized approach ($\alpha = 1$ in the Eq. 4). The $Score(q_f, d)$ is computed using Eq. 5 with the following parameters: $\beta = 0.6$, $\gamma = 0.2$, $\lambda = 0.2$, $\delta = 0.0$.
2. RUN2: This run is also a non-personalized approach ($\alpha = 1$ in the Eq. 4), very similar to the run RUN1. The $Score(q_f, d)$ is computed using Eq. 5 with the following parameters: $\beta = 0.6$, $\gamma = 0.2$, $\lambda = 0$, $\delta = 0.2$.
3. RUN3: This run is a personalized approach where the user u is represented by his filtered profile $P_{Filtered}$ in the $Score(u, d)$ function (Eq.4). The $Score(q_f, d)$ represents the score of d according to the RUN1 above. The score $Score(u, d)$ of equation (4) is equal to $Score_{BM25}(P_{Filtered}(q_f, u), d)$. The parameter α is equal to 0.7.
4. RUN4: This run is a personalized approach where the user is represented by his filtered profile $P_{Filtered}$ in the $Score(u, d)$ function (Eq.4). The $Score(q_f, d)$ represents the score of d according to the RUN1 above. The score $Score(u, d)$ of equation (4) is equal to $Score_{LM}(P_{Filtered}(q_f, u), d)$. The parameter α is equal to 0.7.
5. RUN5: This run is a personalized approach where the user is represented by his weighted profile $P_{Weighted\_filtered}$ in the $Score(u, d)$ function (Eq.4). The $Score(q_f, d)$ represents the score of d according to the RUN1 above. The score $Score(u, d)$ of equation (4) is equal to $Score_{BM25}(P_{Weighted\_Filtered}(q_f, u), d)$. The parameter α is equal to 0.7.
6. RUN6: This run is a personalized approach where the user is represented by his weighted profile $P_{Weighted\_filtered}$ in the $Score(u, d)$ function (Eq.4). The $Score(q_f, d)$ represents the score of d according to the RUN1 above. The score $Score(u, d)$ of equation (4) is equal to $Score_{LM}(P_{Weighted\_Filtered}(q_f, u), d)$. The parameter α is equal to 0.7.

All these runs parameters are described in table 1.

Table 1. Runs Parameters

Run	α	$Score(q_f, d)$				$Score(u, d)$
		β	γ	λ	δ	profile
RUN1	1.0	0.6	0.2	0.2	0.0	-
RUN2	1.0	0.6	0.2	0.0	0.2	-
RUN3	0.7	0.6	0.2	0.2	0.0	filtered
RUN4	0.7	0.6	0.2	0.2	0.0	filtered
RUN5	0.7	0.6	0.2	0.2	0.0	filtered, weighted
RUN6	0.7	0.6	0.2	0.2	0.0	filtered, weighted

3.3 Official Results

According to table 2, we see that all of our results are quite close to each other. Even the weighted profiles compared to unweighted ones does not change results: runs $RUN6$ and $RUN3$ on one side and runs $RUN4$ and $RUN5$ lead to exactly the same result lists, and therefore the same evaluation results. This may be explained by the fact that the top-10 terms are usually very close (according to the cosine similarity) to the query terms, leading to weights close to 1.0. This has to be studied more carefully in the future. We remark that our two best runs ($RUN2$ and $RUN6$) include word embeddings. Both of them use *Word Embedding* applied on Language Models. It seems so that the integration of word embeddings behaves a bit better on language models than on BM25. Our best run, $RUN2$, is not personalized. This shows that using word embeddings in the context of personalization must be further studied.

Table 2. Official Results

Rank	Run	NDCG@10	MRR	MAP	R@1000
18	RUN2	0.0889	0.1889	0.0518	0.3491
19	RUN6	0.0872	0.1914	0.0538	0.3652
20	RUN3	0.0872	0.1914	0.0538	0.3652
21	RUN1	0.0864	0.1858	0.0529	0.3654
22	RUN5	0.0861	0.1866	0.0525	0.3652
23	RUN4	0.0861	0.1866	0.0524	0.3652

3.4 Unofficial Results

We depict here a strange behavior that we faced after the release of the official relevance judgments: the reranking (described in Section 3.1) that removes the documents that belong to the user catalogue does lower the evaluations results. Table 3 presents the $NDCG@10$ values with and without reranking, with the relative improvement. This is explained by the fact that we chose to remove all the ISBN documents that correspond to the documents from the user catalogue, and not only the documents that correspond to the LT id of the books. This approach increased the quality of the results in 2015, but is clearly not adequate for this year. This may come from a different removal, in the relevance judgments, of the documents in the user's catalogue.

Table 3. Unofficial Results

Run	NDCG@10 official	NDCG@10 non reranked (%)
RUN1	0.0864	0.1073 (+24.2%)
RUN2	0.0889	0.1092 (+22.8%)
RUN3	0.0872	0.1073 (+23.1%)
RUN4	0.0861	0.1063 (+23.5%)
RUN5	0.0861	0.1063 (+23.(%)
RUN6	0.0872	0.1073 (+23.1%)

4 Conclusion

We presented in this paper a joint work between the universities of Grenoble and Saint-Étienne. Our focus was to study the integration of word embeddings for the Social Book Suggestion task. We find that the nature of the queries pose a great challenge to an effective use of word embeddings in this context. Weighting the expansion terms does not lead to better results. Our future works may help to understand this behavior. One last point is related to the removal of the documents that belong to the user's catalogue: removing all the ISBN ids and not only the LT ids from the results, assuming that is a user already bought one edition of a book he does not want another edition, is clearly not the right choice, as such removal decreased substantially our official results.

References

1. Almasri, M., Berrut, C., Chevallet, J.: A comparison of deep learning based query expansion with pseudo-relevance feedback and mutual information. In: European Conference on IR Research. pp. 709–715. ECIR'16 (2016)
2. 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 41(6), 391–407 (1990)
3. Ganguly, D., Roy, D., Mitra, M., Jones, G.J.: Word embedding based generalized language model for information retrieval. In: Conference on Research and Development in Information Retrieval. pp. 795–798. SIGIR'15, New York, USA (2015)
4. Goldberg, Y., Levy, O.: word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. CoRR abs/1402.3722 (2014)
5. Kim, Y.: Convolutional neural networks for sentence classification. CoRR abs/1408.5882 (2014)
6. Koolen, M., Bogers, T., Gäde, M., Hall, M.A., Huurdeman, H.C., Kamps, J., Skov, M., Toms, E., Walsh, D.: Overview of the CLEF 2015 Social Book Search lab. In: Conference and Labs of the Evaluation Forum. pp. 545–564. CLEF'15, Toulouse, France (2015)

7. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR abs/1301.3781 (2013)
8. Nalisnick, E., Mitra, B., Craswell, N., Caruana, R.: Improving document ranking with dual word embeddings. In: Conference Companion on World Wide Web. pp. 83–84. WWW'16 Companion, Montreal, Quebec, Canada (2016)
9. Ould-Amer, N., Mulhem, P., Géry, M.: LIG at CLEF 2015 SBS Lab. In: Conference and Labs of the Evaluation Forum. CLEF'15, Toulouse, France (2015)
10. Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C., Lioma, C.: Terrier: A High Performance and Scalable Information Retrieval Platform. In: Workshop on Open Source Information Retrieval. OSIR'06, Seattle, Washington, USA (2006)
11. Widdows, D.: Geometry and Meaning. Center for the Study of Language and Information/SRI (2004)