# Learning Word Embeddings from Wikipedia for Content-Based Recommender Systems

Cataldo Musto(✉), Giovanni Semeraro, Marco de Gemmis, and Pasquale Lops

Department of Computer Science, University of Bari Aldo Moro, Bari, Italy
{cataldo.musto,giovanni.semeraro,marco.gemmis,pasquale.Lops}@uniba.it

**Abstract.** In this paper we present a preliminary investigation towards the adoption of *Word Embedding* techniques in a content-based recommendation scenario. Specifically, we compared the effectiveness of three widespread approaches as Latent Semantic Indexing, Random Indexing and Word2Vec in the task of learning a vector space representation of both items to be recommended as well as user profiles.

To this aim, we developed a content-based recommendation (CBRS) framework which uses textual features extracted from Wikipedia to learn user profiles based on such Word Embeddings, and we evaluated this framework against two state-of-the-art datasets. The experimental results provided interesting insights, since our CBRS based on Word Embeddings showed results comparable to those of well-performing algorithms based on Collaborative Filtering and Matrix Factorization, especially in high-sparsity recommendation scenarios.

## 1 Introduction

Word Embedding techniques recently gained more and more attention due to the good performance they showed in a broad range of natural language processing-related scenarios, ranging from *sentiment analysis* [10] and *machine translation* [2] to more challenging ones as learning a textual description of a given image[1].

However, even if some recent research gave new lymph to such approaches, Word Embedding techniques took their roots in the area of Distributional Semantics Models (DSMs), which date back in the late 60's [3]. Such models are mainly based on the so-called *distributional hypothesis*, which states that the meaning of a word depends on its *usage* and on the *contexts* in which it occurs. In other terms, according to DSMs, it is possible to infer the meaning of a term (e.g., leash) by analyzing the other terms it co-occurs with (dog, animal, etc.). In the same way, the correlation between different terms (e.g., leash and muzzle) can be inferred by analyzing the similarity between the contexts in which they are used. Word Embedding techniques have inherited the vision carried out by DSMs, since they aim to learn in a totally unsupervised way a *low-dimensional*

[1] http://googleresearch.blogspot.it/2014/11/a-picture-is-worth-thousand-coherent.html


N. Ferro et al. (Eds.): ECIR 2016, LNCS 9626, pp. 729–734, 2016.
DOI: 10.1007/978-3-319-30671-1_60

vector space representation of *words* by analyzing the usage of the terms in (very) large corpora of textual documents. Many popular techniques fall into this class of algorithms: Latent Semantic Indexing [1], Random Indexing [8] and the recently proposed Word2Vec [5], to name but a few.

In a nutshell, all these techniques carry out the learning process by encoding linguistic regularities (e.g., the co-occurrences between the terms or the occurrence of a term in a document) in a huge matrix, as a term-term or term-document matrix. Next, each Word Embedding technique adopts a different technique to reduce the overall dimension of the matrix by maintaining most of the *semantic nuances* encoded in the original representation. One of the major advantages that comes from the adoption of Word Embedding techniques is that the dimension of the representation (that is to say, the size of the vectors) is just a parameter of the model, so it can be set according to specific constraints or peculiarities of the data. Clearly, the smaller the vectors, the bigger the loss of information.

Although the effectiveness of such techniques (especially when combined with deep neural network architectures) is already taken for granted, just a few work investigated how well they do perform in recommender systems-related tasks. In [6], Musto et al. proposed a content-based recommendation model based on Random Indexing. Similarly, the effectiveness of LSI in a content-based recommendation scenario is evaluated in [4]. However, none of the current literature carried out a comparative analysis among such techniques: to this aim, in this work we defined a simple content-based recommendation framework based on *word embeddings* and we assessed the effectiveness of such techniques in a content-based recommendation scenario.

## 2 Methodology

### 2.1 Overview of the Techniques

Latent Semantic Indexing (LSI) [1] is a word embedding technique which applies Singular Value Decomposition (SVD) over a word-document matrix. The goal of the approach is to *compress* the original information space through SVD in order to obtain a smaller-scale word-*concepts* matrix, in which each column models a *latent concept* occurring in the original vector space. Specifically, SVD is employed to unveil the latent relationships between terms according to their usage in the corpus.

Next, Random Indexing (RI) [8] is an incremental technique to learn a low-dimensional word representation relying on the principles of the Random Projection. It works in two steps: first, a *context vector* is defined for each context (the definition of context is typically scenario-dependant. It may be a paragraph, a sentence or the whole document). Each context vector is ternary (it contains values in $\{-1, 0, 1\}$) very sparse, and its values are *randomly distributed*. Given such context vectors, the vector space representation of each word is obtained by just summing over all the representations of the contexts in which the word occurs. An important peculiarity of this approach is that it is incremental and

scalable: if any new documents come into play, the vector space representation of the terms is updated by just adding the new occurrences of the terms in the new documents.

Finally, Word2Vec (W2V) is a recent technique proposed by Mikolov et al. [5]. The approach learns a vector-space representation of the terms by exploiting a two-layers neural network. In the first step, weights in the network are randomly distributed as in RI. Next, the network is trained by using the Skip-gram methodology in order to model fine-grained regularities in word usage. At each step, weights are updated through Stochastic Gradient Descent and a vector-space representation of each term is obtained by extracting the weights of the network at the end of the training.

### 2.2 Recommendation Pipeline

Our recommendation pipeline follows the classical workflow carried out by a content-based recommendation framework. It can be split into four steps:

1. Given a set of items $I$, each $i \in I$ is mapped to a Wikipedia page through a semi-automatic procedure. Next, textual features are gathered from each Wikipedia page and the extracted content is processed through a Natural Language Processing pipeline to remove noisy features. More details about this process are provided in Sect. 3.
2. Given a vocabulary $V$ built upon the description of the items in $I$ extracted from Wikipedia, for each word $w \in V$ a vector space representation $w_T$ is learnt by exploiting a word embedding technique $T$.
3. For each item $i \in I$, a vector space representation of the item $i_T$ is built. This is calculated as the centroid of the vector space representation of the words occurring in the document.
4. Given a set of users $U$, a user profile for each $u \in U$ is built. The vector space representation of the profile is learnt as the centroid of the vector space representation of the items the user previously liked
5. Given a vector space representation of both items to be recommended and user profile, recommendations are calculated by exploiting classic similarity measures: items are ranked according to their decreasing similarity and top-K recommendations are returned to the user.

Clearly, this is a very basic formulation, since more *fine-grained representations* can be learned for both items and users profiles. However, this work just intends to preliminarily evaluate the effectiveness of such representations in a simplified recommendation framework, in order to pave the way to several future research directions in the area.

## 3 Experimental Evaluation

Experiments were performed by exploiting two state-of-the-art datasets as MovieLens[2] and DBbook[3]. The first one is a dataset for movie recommendations,

[2] http://grouplens.org/datasets/movielens/.
[3] http://challenges.2014.eswc-conferences.org/index.php/RecSys.

while the latter comes from the ESWC 2014 Linked-Open Data-enabled Recommender Systems challenge and focuses on book recommendation. Some statistics about the datasets are provided in Table 1.

A quick analysis of the data immediately shows the very different nature of the datasets: even if both of them resulted as very sparse, MovieLens is more dense than DBbook (93.69 % vs. 99.83 % sparsity), indeed each Movielens user voted 84.83 items on average (against the 11.70 votes given by DBbook users). DBbook has in turn the peculiarity of being unbalanced towards negative ratings (only 45 % of positive preferences). Furthermore, MovieLens items were voted more than DBbook ones (48.48 vs. 10.74 votes for item, on average).

**Experimental Protocol.** Experiments were performed by adopting different protocols: as regards MovieLens, we carried out a 5-folds cross validation, while a single training/test split was used for DBbook. In both cases we used the splits which are commonly used in literature. Given that MovieLens preferences are expressed on a 5-point discrete scale, we decided to consider as *positive* ratings only those equal to 4 and 5. On the other side, the DBbook dataset is already available as *binarized*, thus no further processing was needed. Textual content was obtained by mapping items to Wikipedia pages. All the available items were successfully mapped by querying the title of the movie or the name of the book, respectively. The extracted content was further processed through a NLP pipeline consisting of a stop-words removal step, a POS-tagging step and a lemmatization step. The outcome of this process was used to learn the Word Embeddings. For each word embedding technique we compared two different sizes of learned vectors: 300 and 500. As regards the baselines, we exploited MyMediaLite library[4]. We evaluated User-to-User (U2U-KNN) and Item-to-Item Collaborative Filtering (I2I-KNN) as well as the Bayesian Personalized Ranking Matrix Factorization (BPRMF). U2U and I2I neighborhood size was set to 80, while BPRMF was run by setting the factor parameter equal to 100. In both cases we chose the optimal values for the parameters.

**Table 1.** Description of the datasets

| | MovieLens | DBbook |
|---|---|---|
| **Users** | 943 | 6,181 |
| **Items** | 1,682 | 6,733 |
| **Ratings** | 100,000 | 72,372 |
| **Sparsity** | 93.69 % | 99.83 % |
| **Positive Ratings** | 55.17 % | 45.85 % |
| **Avg. ratings/user ± stdev** | 84.83±83.80 | 11.70±5.85 |
| **Avg. ratings/item ± stdev** | 48.48±65.03 | 10.74±27.14 |

[4] http://www.mymedialite.net/.

**Table 2.** Results of the experiments. The best word embedding approach is highlighted in bold. The best overall configuration is highlighted in bold and underlined. The baselines which are overcame by at least a word embedding are reported in italics.

| MovieLens | W2V | | RI | | LSI | | U2U | I2I | BPRMF |
|---|---|---|---|---|---|---|---|---|---|
| Vector size | 300 | 500 | 300 | 500 | 300 | 500 | | | |
| F1@5 | **0.5056** | 0.5054 | 0.4921 | 0.4910 | 0.4645 | 0.4715 | **0.5217** | *0.5022* | 0.5141 |
| F1@10 | **0.5757** | 0.5751 | 0.5622 | 0.5613 | 0.5393 | 0.5469 | **0.5969** | 0.5836 | 0.5928 |
| F1@15 | 0.5672 | **0.5674** | 0.5349 | 0.5352 | 0.5187 | 0.5254 | **0.5911** | 0.5814 | 0.5876 |
| DBbook | W2V | | RI | | LSI | | U2U | I2I | BPRMF |
| Vector size | 300 | 500 | 300 | 500 | 300 | 500 | | | |
| F1@5 | 0.5183 | **0.5186** | 0.5064 | 0.5039 | 0.5056 | 0.5076 | 0.5193 | *0.5111* | **0.5290** |
| F1@10 | 0.6207 | 0.6209 | 0.6239 | 0.6244 | 0.6256 | **0.6260** | *0.6229* | *0.6194* | **0.6263** |
| F1@15 | 0.5829 | 0.5828 | 0.5892 | 0.5887 | 0.5908 | **0.5909** | *0.5777* | *0.5776* | *0.5778* |

**Discussion of the Results.** The first six columns of Table 2 provide the results of the comparison among the word embedding techniques. As regards MovieLens, W2V emerged as the best-performing configuration for all the metrics taken into account. The gap is significant when compared to both RI and LSI. Moreover, results show that the size of the vectors did not significantly affect the overall accuracy of the algorithms (with the exception of LSI). This is an interesting outcome since with an even smaller word representation, word embeddings can obtain good results. However, the outcomes emerging from this first experiments are controversial, since DBbook data provided opposite results: in this dataset W2V is the best-performing configuration only for F1@5. On the other side, LSI, which performed the worst on MovieLens data, overcomes both W2V and RI on F1@10 and F1@15. At a first glance, these results indicate non-generalizable outcomes. However, it is likely that such behavior depends on specific peculiarities of the datasets, which in turn influence the way the approaches learn their vector-space representations. A more thorough analysis is needed to obtain general guidelines which drive the behavior of such approaches.

Next, we compared our techniques to the above described baselines. Results clearly show that the effectiveness of word embedding approaches is directly dependent on the sparsity of the data. This is an expected behavior since content-based approaches can better deal with cold-start situations. In highly sparse dataset such as DBbook (99.13 % against 93.59 % of MovieLens), content-based approaches based on word embedding tend to overcome the baselines. Indeed, RI and LSI, overcome I2I and U2U on F1@10 and F1@15 and W2V overcomes I2I on F1@5 and I2I and U2U on F1@15. Furthermore, it is worth to note that on F1@10 and F@15 word embeddings can obtain results which are comparable (or even better on F1@15) to those obtained by BPRMF. This is a very important outcome, which definitely confirms the effectiveness of such techniques, even compared to matrix factorization techniques. Conversely, on less sparse datasets as MovieLens, collaborative filtering algorithms overcome their content-based counterpart.

## 4 Conclusions and Future Work

In this paper we presented a preliminary comparison among three widespread techniques in the task of learning Word Embeddings in a content-based recommendation scenario. Results showed that our model obtained performance comparable to those of state-of-the art approaches based on collaborative filtering. In the following, we will further validate our results by also further investigating both the effectiveness of novel and richer textual *data silos*, as those coming from the Linked Open Data cloud, and more expressive and complex Word Embedding techniques, as well as by extending the comparison to hybrid approaches such as those reported in [9] or in context-aware recommendation settings [7].

## References

1. Deerwester, S., Dumais, S., Landauer, T., Furnas, G., Harshman, R.: Indexing by latent semantic analysis. JASIS **41**, 391–407 (1990)
2. Gouws, S., Bengio, Y., Corrado, G. Bilbowa: Fast bilingual distributed representations without word alignments (2014). arXiv:1410.2455
3. Harris, Z.S.: Mathematical Structures of Language. Interscience, New York (1968)
4. McCarey, F., Cinnéide, M.Ó., Kushmerick, N.: Recommending library methods: an evaluation of the vector space model (VSM) and latent semantic indexing (LSI). In: Morisio, M. (ed.) ICSR 2006. LNCS, vol. 4039, pp. 217–230. Springer, Heidelberg (2006)
5. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 3111–3119 (2013)
6. Musto, C., Semeraro, G., Lops, P., de Gemmis, M.: Random indexing and negative user preferences for enhancing content-based recommender systems. In: Huemer, C., Setzer, T. (eds.) EC-Web 2011. LNBIP, vol. 85, pp. 270–281. Springer, Heidelberg (2011)
7. Musto, C., Semeraro, G., Lops, P., de Gemmis, M.: Combining distributional semantics and entity linking for context-aware content-based recommendation. In: Dimitrova, V., Kuflik, T., Chin, D., Ricci, F., Dolog, P., Houben, G.-J. (eds.) UMAP 2014. LNCS, vol. 8538, pp. 381–392. Springer, Heidelberg (2014)
8. Sahlgren, M.: An introduction to random indexing. In: Methods and Applications of Semantic Indexing Workshop, TKE 2005 (2005)
9. Semeraro, G., Lops, P., Degemmis, M.: Wordnet-based user profiles for neighborhood formation in hybrid recommender systems. In: Fifth International Conference on Hybrid Intelligent Systems, HIS 2005, pp. 291–296. IEEE (2005)
10. Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., Qin, B.: Learning sentiment-specific word embedding for twitter sentiment classification. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, vol. 1, pp. 1555–1565 (2014)