Representing Items as Word-Embedding Vectors and Generating Recommendations by Measuring their Linear Independence*

Ludovico Boratto, Salvatore Carta, Gianni Fenu, and Roberto Saia
Dipartimento di Matematica e Informatica, Università di Cagliari
Via Ospedale 72, 09124 Cagliari - Italy
ludovico.boratto@acm.org, {salvatore, fenu, roberto.saia}@unica.it

ABSTRACT

In order to generate effective results, it is essential for a recommender system to model the information about the user interests in a profile. Even though word embeddings (i.e., vector representations of textual descriptions) have proven to be effective in many contexts, a content-based recommendation approach that employs them is still less effective than collaborative strategies (e.g., SVD). In order to overcome this issue, we introduce a novel criterion to evaluate the word-embedding representation of the items a user rated. The proposed approach defines a vector space in which the similarity between an unevaluated item and those in a user profile is measured in terms of linear independence. Experiments show its effectiveness to perform a better ranking of the items, w.r.t. collaborative filtering, both when compared to a latent-factor-based approach (SVD) and to a classic neighborhood user-based system.

CCS Concepts

•Information systems → Data mining; Recommender systems; Learning to rank;

Keywords

Semantic Analysis; Word Embeddings; Algorithms; Metrics.

1. INTRODUCTION

Recommender systems are essential for e-commerce companies, to filter the huge amounts of items they can provide and improve the quality and efficiency of the sales criteria [3]. In order to perform this task, these systems need to define a set of profiles that model the preferences of their customers. In this context, the collaborative techniques, which represent a user with the ratings given to the items she evaluated, are usually the most effective. Even though semantic technologies are moving at a very rapid pace and state-of-the-art solutions, such as deep learning algorithms able to extract *word embeddings* [1] from a text corpus, have been successfully employed in numerous information filtering and retrieval tasks, at the moment collaborative filtering approaches continue to be more accurate at generating recommendations [2]. The extraction of the word embeddings from a corpus is made possible thanks to several state-of-the-art tools, which create a vector representation of each word (Google's *word2vec*[1]) or document (a *word2vec* extension, usually known as *doc2vec*).

A user profile represented by a unique vector of features allows a system to perform quick comparisons, e.g., in a content-based system, it can be easily compared to the vector that represents an item with a simple metric, like the cosine similarity. The idea behind this paper is to *represent a user profile as a matrix of word-embeddings*, where each row is represented by an item a user positively evaluated, and to *define a metric able to evaluate the correlation between an item not rated by a user and those in her matrix-based user profile, in terms of linear independence.*

We observe that if the vector representation of an unevaluated item is linearly dependent to the items in a user profile that have been positively evaluated, their features match, thus it is similar to the user preferences. This leads to a higher accuracy of a recommender system w.r.t. collaborative approaches.

2. APPROACH

Here, we present the steps our approach performs:

Item Vectorization. Given a set I of items, we first define and train a model by using the *doc2vec* neural net (by using as source the textual description of the items in I). The result is the vector representation of the items in I, where the cardinality of each vector (item) depends on the number L of features used in the *doc2vec* vectorization process. Given a user, the output of this step will be a matrix that contains the embeddings of the items positively evaluated by her (denoted as I_u), plus an empty row to employ during the filtering process to evaluate the items not yet evaluated by the user.

Linear Independence Rate. To evaluate the similarity between the items in the matrix-based user profile and an

*This work is partially funded by Regione Sardegna under project NOMAD (Next generation Open Mobile Apps Development), through PIA - Pacchetti Integrati di Agevolazione "Industria Artigianato e Servizi" (annualità 2013).

RecSys 2016. September 15–19, 2016, Boston, MA, USA.

[1] http://deeplearning4j.org/word2vec

unevaluated one, we define the Linear Independence Rate (LIR) coefficient. It is the average of the determinants of all square sub-matrices, defined by decomposing the user profile matrix in square sub-matrices of size $|I_u| \times |I_u|$ (with $|I_u| \leq L$, otherwise we have only a matrix of size $L \times L$). We calculate the LIR value by moving on the entire user profile matrix, extracting the determinant of each sub-matrix, without overlaps. We can note that the maximum size of the square sub-matrices is the cardinality of the vectors, i.e., the L parameter used to build the *doc2vec* model. Through this compositional process, we evaluate the LIR of an item by placing its vector representation as last element of the user profile. We consider closer to the preferences of a user the items with a LIR value as close as possible to zero.

Ranking Algorithm. The Algorithm 1 takes as input the items I, a user u, the items I_u she evaluated, and the number of features L in the vectors created by *doc2vec* (i.e., the *layerSize* parameter). It returns as output a list R of the items not evaluated by the user u (ranked by LIR value).

Algorithm 1 *Items evaluation and ranking*

Input: I=Set of items, u=User, I_u=Items of u, L=layerSize
Output: R = List of ranked items

```
1: procedure GetRankedItems(I,u,I_u,L)
2:     if |I_u| > L then I_u=GetLastLItems(I_u, L)
3:     end if
4:     V=Doc2VecVectorization(I)
5:     M=DefineUserProfileMatrix(V,I_u)
6:     M=AddEmptyVectorAsLastRow(M);
7:     for each i in I do
8:         if i NOT IN I_u then
9:             v=GetItemVector(V, i)
10:            M=FillLastMatrixRow(M, v)
11:            LIR=CalculateLIR(M)
12:            R ← (i, LIR)
13:        end if
14:    end for
15:    Return SortItemsByDescLIR(|R|)
16: end procedure
```

It should be noted that our approach is scalable by employing distributed computing models (e.g., *MapReduce*). Indeed, the computation of the LIR metric for each unevaluated item can be distributed over different machines.

3. EVALUATION

We compared our approach with two collaborative filtering approaches: CF, which is based on a classic neighborhood model, and SVD, based on the latent factor model. The experiments have been performed by using a dataset that represents a standard benchmark for recommender systems, i.e., Movielens 1M, composed by *6,040* users, *3,900* items, and *1,000,209* ratings.

The criterion adopted for obtaining the training and the test sets was the *K-fold cross validation* with $K = 3$, and the independent-samples *two-tailed Student's t-tests*. The adopted metric is the *Mean Reciprocal Rank* (MRR), a statistical measure able to evaluate the ranking generated for a set of elements that belong to a certain domain.

The first experiment evaluates the metric capability to measure the similarity between a user profile and an item, in terms of linear independence (Figure 1: Top). On the basis of our approach, we rank all the items in decreasing order, by verifying that almost all of those in the test set have been placed in the top positions (i.e., $1 \div 20$), proving the LIR capability to effectively rank the items.

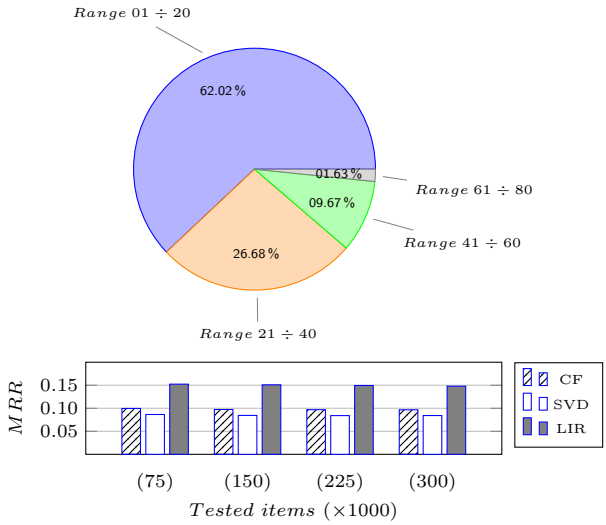

Figure 1: (Top) ***Linear similarity evaluation.*** **(Bottom)** ***Ranking accuracy evaluation***

The second experiment evaluates the metric capability to infer the future choices of the users (Figure 1: Bottom), by comparing the rank assigned to the items in the test set by using the LIR metric, with those assigned to the same items by the other state-of-the-art approaches taken into account.

The t-tests highlighted a statistical difference between the results ($p < 0.05$).

4. DISCUSSION AND CONCLUSIONS

The experimental results show the effectiveness of the compositional approach used by our LIR metric, as well as its capability to overcome the canonical state-of-the-art metrics, in terms of modeling of the user preferences. Indeed, we have a strong improvement in the process of rating of the unevaluated items, and this means that our metric assigns an higher score (w.r.t. the state-of-the-art approaches to which we compared) to the items positively evaluated by a user. The use of the LIR metric can be also extended to other contexts that do not use word embeddings, e.g., those based on a canonical term-document matrix.

Future work. We will consider our metric to evaluate the items negatively evaluated by the users, in order to obtain additional information in terms of unpreferred items, exploiting it to improve the recommendations accuracy, e.g., by verifying the preferences collision (i.e., very similar items rated both positively and negatively by a user).

5. REFERENCES

[1] R. Fu, J. Guo, B. Qin, W. Che, H. Wang, and T. Liu. Learning semantic hierarchies via word embeddings. In *ACL (1)*, pages 1199–1209, 2014.

[2] C. Musto, G. Semeraro, M. de Gemmis, and P. Lops. Learning word embeddings from wikipedia for content-based recommender systems. In *Advances in Information Retrieval*, pages 729–734. Springer, 2016.

[3] S. Sivapalan, A. Sadeghian, H. Rahnama, and A. M. Madni. Recommender systems in e-commerce. In *World Automation Congress (WAC), 2014*, pages 179–184. IEEE, 2014.