ECAI 2020
G.D. Giacomo et al. (Eds.)

doi:10.3233/FAIA200449

Shallow Neural Models for Top-N Recommendation

Alfonso Landin[1] and **Daniel Valcarce**[1] and **Javier Parapar**[1] and **Álvaro Barreiro**[1]

Abstract. In the field of Information Retrieval, word embedding models have shown to be effective in several tasks. In this paper, we show how one of these neural embedding techniques can be adapted to the recommendation task. This adaptation only makes use of collaborative filtering information, and the results show that it is able to produce effective recommendations efficiently.

1 INTRODUCTION

In recent years, user interaction with information systems has shifted from a proactive approach, where the user actively seeks the information, to a more passive role, where the system suggests content to the user. Recommender Systems (RS) have played a pivotal role in this change, allowing users of a system to wade through the vast amounts of available information.

Nowadays, top-N recommendation is the most popular task [1]. The goal of this task is to produce an ordered list of N items for each user. Ranking metrics are commonly used for evaluating the accuracy of the suggestions. Moreover, there are other properties, such as diversity and novelty, usually considered in RS evaluation.

In this paper, we briefly present `prefs2vec` [6] and PRIN [3], two collaborative filtering recommenders. While the approaches differ, both systems share a common underlying neural network architecture and training method, our adaptation of a popular Natural Language Processing word embedding model to CF. In `prefs2vec` the embeddings are used with a memory-based recommender, and PRIN is a probabilistic model-based approach.

2 WORD EMBEDDING MODELS

In the Natural Language Processing (NLP) and Information Retrieval (IR) fields, words and documents have been traditionally represented using sparse high-dimensional vectors with one-hot and bag-of-words models. Recently, the use of densely distributed representations with lower dimensionality has gained attraction. In particular, `word2vec` [4, 5] has proven successful at several NLP tasks.

In `word2vec` two neural models are defined, the continuous bag-of-words (CBOW) and the skip-gram (SG) models. The neural architecture is the same for both models, a fully connected feedforward network with a single hidden layer. Their difference lies in how they are trained, with CBOW learning to predict a word given its context, i.e., the words surrounding target term in a fixed-length window, while SG learns to predict the context given the word.

[1] Information Retrieval Lab, Centro de Investigación en Tecnoloxías da Información e as Comunicacións (CITIC), Universidade da Coruña, Spain, email: {alfonso.landin,daniel.valcarce,javierparapar,barreiro}@udc.es

3 A USER AND ITEM EMBEDDING MODEL

Word embedding models assume that words appearing inside a fixed-length context are related, and also that words that appear in different documents surrounded by comparable contexts are similar. We postulate that those relations also hold on collaborative filtering data.

In an item-based scenario, items that have been rated by the same user are assumed to have some relation. Moreover, if we take two users that have rated all the same items when one user rates a new item and the other rates another new item, we consider these two items to be similar. In the user-based counterpart, all the users of an item profile are related, and users that appear in different item profiles surrounded by the same users are alike.

We adapted the CBOW model because we thought predicting an item from a profile is a better fit for the recommendation task. Moreover, CBOW is more efficient than the SG model [4]. Figure 1 shows the architecture of the model for the user-based (UB) case. The size of the input layer is the number of users $|\mathcal{U}|$, the size of the hidden layer is d, a hyper-parameter of the model, and the output layer's size is also the number of users. Given an item, i and its profile $\mathcal{U}_i$, i.e., the users that rated that item, the model is trained to predict each user u of the item profile given the rest of the users in that profile. An analogous architecture can be constructed for the item-based (IB) counterpart.

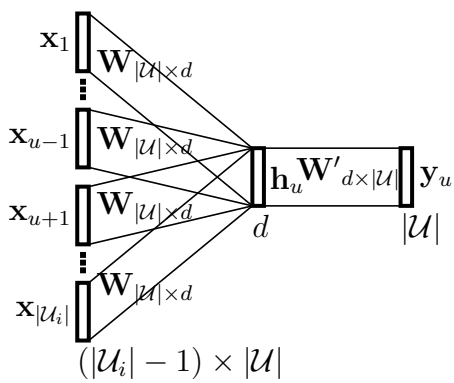

Figure 1. Neural model of `prefs2vec` and PRIN

The input of the model for a particular target user u in the profile of item i consists of the context vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_{u-1}, \mathbf{x}_{u+1}, \ldots, \mathbf{x}_{|\mathcal{U}_i|}\}$, all the users that have rated i except u, using one-hot encoding. The matrix $\mathbf{W} \in \mathbb{R}^{|\mathcal{U}|\times d}$ contains the weights of the connections of the input layer with the hidden layer. Each row of this matrix, $\mathbf{v}_v$, is taken as the input embedding of the user v. The activation function for the hidden layer is a linear function. The output of the hidden layer for the target user u, $\mathbf{h}_u$, is calculated by averaging

the input embeddings weighted by the rating given by each user:

$$\mathbf{h}_u = \frac{\mathbf{W}}{\sum_{v \in \mathcal{U}_i \setminus \{u\}} r_{v,i}} \sum_{v \in \mathcal{U}_i \setminus \{u\}} r_{v,i}\, \mathbf{x}_v = \frac{\sum_{v \in \mathcal{U}_i \setminus \{u\}} r_{v,i}\, \mathbf{v}_v}{\sum_{v \in \mathcal{U}_i \setminus \{u\}} r_{v,i}} \quad (1)$$

where $r_{v,i}$ is the rating given by user v to item i. We use the weighted average to incorporate the user ratings into the training process.

The ouput layer has one output for each user of the system, making its size $|\mathcal{U}|$. This layer uses a softmax activation function. The weights between the hidden and the output layer are represented in the matrix $\mathbf{W}' \in \mathbb{R}^{d \times |\mathcal{U}|}$, with each column $\mathbf{v}'_u$ being the output embedding for user u. The component u of the output vector for target user, $\mathbf{y}_u$, is calculated, using the softmax function, as:

$$p(u \,|\, \mathcal{U}_i \setminus \{u\}) = (\mathbf{y}_u)_u = \frac{\exp\left(\mathbf{v}'^T_u \mathbf{h}_u\right)}{\sum_{v \in \mathcal{U}} \exp\left(\mathbf{v}'^T_v \mathbf{h}_u\right)} \quad (2)$$

We generate a training example for each user u in the profile of item i, for all the items. We train the model to maximize the likelihood of the data, which is equivalent to minimizing the negative log likelihood. Therefore the loss function is:

$$\mathcal{L} = -\sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{U}_i} \log p\big(u \,|\, \mathcal{U}_i \setminus \{u\}\big) \quad (3)$$

We learn $\mathbf{W}$ and $\mathbf{W}'$ by back-propagation. The use of the softmax function makes training this model impractical for large scale scenarios [4, 5]: the cost of calculating the output of each training example which is proportional to the number of users (see Eq. 2). Therefore, we decided to use the negative sampling instead of softmax.

Additionally, Eq. 3 does not include any regularization. Our experiments showed that the model was over-fitting the data. We choose to use dropout regularization in the input layer to alleviate the problem. We choose this over other types of regularization, such as ℓ_2, because it provided better effectiveness and training times. At the same time, it allows us to leverage existing `word2vec` implementations.

4 USING THE MODEL TO RECOMMEND

For recommending with this network, the simplest approach is to use the embeddings produced with the model in a memory-based recommender. We use the cosine similarity between the user embeddings to compute the user neighborhoods. We named this approach `prefs2vec` [6], and source code is available online[2]. Producing recommendations this way yields better results than the baseline memory-based recommender using cosine similarity over the classical user representation, in terms of accuracy, novelty, and diversity. It also surpasses several model-based baselines.

Another possibility is to use the prediction capabilities of the model directly to make the recommendations. We do so by feeding a complete item profile without removing any user. The output is the posterior probability distribution of users given the item. This output is not suitable to produce a ranking, as $p(u|i)$ and $p(u|j)$, $i \neq j$, are not comparable because they are in different event spaces. We can use Bayes' rule to calculate the probability of the items given the user, which is the basis for the ranking. The prior probability of the users can be obviated to produce the ranking. A prior probability distribution of the items needs to be provided. We dubbed this recommender PRIN [3], and the source code is available online[3]. By choosing between different estimations of the prior probability of items, we can tune the properties of the recommender.

Results of experiments with the MovieLens 20M dataset for the proposed models and recognized state-of-the-art model-based [2] and memory-based [7] baselines can be seen in Table 1.

Table 1. Results on accuracy (nDCG@100), diversity (Gini@100) and novelty (MSI@100) on the MovieLens 20M dataset. Statistical significant improvements (wilcoxon test with $p < 0.01$) in nDCG@100 and MSI@100 with respect to WSR-UB, WSR-IB, WRMF, `prefs2vec`-UB, `prefs2vec`-IB and PRIN are annotated with a, b, c, d, e and f respectively.

Model	nDCG@100	Gini@100	MSI@100
WSR-UB	0.4449^{be}	0.0310	210.0885^{df}
WSR-IB	0.3842	0.0310	213.6453^{adf}
WRMF	0.4466^{abe}	0.0481	248.0235^{abdf}
`prefs2vec`-UB	$\mathbf{0.4504}^{abcef}$	0.0317	205.9805
`prefs2vec`-IB	0.4172^{b}	**0.0666**	$\mathbf{257.8653}^{abcdf}$
PRIN	$0{,}4470^{abce}$	0,0366	$207{,}2515^{d}$

5 CONCLUSIONS

We briefly described in this paper how a popular shallow network topology from the NLP field could be adapted to the collaborative filtering scenario. We show how to train the model and make recommendations. Results show that the resulting recommenders are more effective than the baselines in the task. Moreover, the training method preserves the efficiency of the original model, making the proposal suitable for large scale scenarios.

ACKNOWLEDGEMENTS

This work was supported by project RTI2018-093336-B-C22 (MCIU & ERDF), project GPC ED431B 2019/03 (Xunta de Galicia & ERDF) and accreditation ED431G 2019/01 (Xunta de Galicia & ERDF). The first author also acknowledges the support of grant FPU17/03210 (MCIU)

REFERENCES

[1] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin, 'Performance of Recommender Algorithms on Top-N Recommendation Tasks', in *4th ACM Conference on Recommender Systems*, pp. 39–46, (2010).

[2] Yifan Hu, Yehuda Koren, and Chris Volinsky, 'Collaborative Filtering for Implicit Feedback Datasets', in *Eighth IEEE International Conference on Data Mining*, pp. 263–272. IEEE, (2008).

[3] Alfonso Landin, Daniel Valcarce, Javier Parapar, and Álvaro Barreiro, 'PRIN: A probabilistic recommender with item priors and neural models', in *41st European Conference on IR Research, Part I*, pp. 133–147, (2019).

[4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, 'Efficient estimation of word representations in vector space', in *1st International Conference on Learning Representations, Workshop Track Proceedings*, (2013).

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean, 'Distributed Representations of Words and Phrases and Their Compositionality', in *26th International Conference on Neural Information Processing Systems*, pp. 3111–3119, (2013).

[6] Daniel Valcarce, Alfonso Landin, Javier Parapar, and Álvaro Barreiro, 'Collaborative filtering embeddings for memory-based recommender systems', *Eng. Appl. Artif. Intell.*, **85**, 347 – 356, (2019).

[7] Daniel Valcarce, Javier Parapar, and Álvaro Barreiro, 'Language Models for Collaborative Filtering Neighbourhoods', in *38th European Conference on Information Retrieval*, pp. 614–625. Springer, (2016).

[2] https://gitlab.irlab.org/alfonso.landin/prefs2vec
[3] https://gitlab.irlab.org/alfonso.landin/prin