Ask Me Any Rating: A Content-based Recommender System based on Recurrent Neural Networks

Cataldo Musto, Claudio Greco, Alessandro Suglia, and Giovanni Semeraro

Department of Computer Science, University of Bari Aldo Moro,
Via E. Orabona 4, 70125 Bari, Italy

Abstract. In this work we propose *Ask Me Any Rating (AMAR)*, a novel content-based recommender system based on deep neural networks which is able to produce top-N recommendations leveraging user and item embeddings which are learnt from textual information describing the items. A comprehensive experimental evaluation conducted on state-of-the-art datasets showed a significant improvement over all the baselines taken into account.

1 Introduction

Internet users have access to a huge amount of information, which grows exponentially every day. They are forced to dive among a lot of different information sources from which they have to choose the most appealing contents. Recommender systems can help users to solve the information overload problem because they are able to filter contents according to user preferences. Content-based recommender systems need to effectively represent user profiles and items to recommend items similar to those a given user has liked in the past.

In recent years, *Deep Learning* techniques have proved to be particularly effective in different machine learning fields such as computer vision, language modeling and speech recognition, reaching state-of-the-art performance. *Deep Learning* models learn a hierarchy of levels of representations from data by using multiple processing layers [2].

In this work, we present a novel deep neural network model called *Ask Me Any Rating (AMAR)* which exploits Recurrent Neural Networks (RNNs) to jointly learn a representation for user preferences and items to be recommended and generates a ranked list of items which may be of interest for a given user. Up to our knowledge, this is the first try to use *Deep Learning* models for content-based recommender systems in a top-N recommendation task.

The paper is organized as follows: section 2 introduces basic terminology and depicts the *AMAR* architecture. Next, in section 3 we report details about the experimental evaluation conducted to establish the quality of the recommendations generated by this model over different baselines on two well-known datasets. In the last section 4, we underline advantages and disadvantages of the architecture we proposed and we sketch future research directions.

2 Ask Me Any Rating

Our model took inspiration from the model based on Long Short-Term Memory (LSTM) network [4] proposed as a baseline for a Question Answering scenario. Indeed, our insight is that the analogy between questions and users profiles with answers and items to be recommended can be exploited to adapt the previously proposed solution in a recommendation scenario as well.

The proposed architecture implements a content-based recommender system able to predict a score $s(u, i)$ which defines the probability of a like given by a user u to a specific item i. In a nutshell, our approach is based on two different modules which jointly learn a *user embedding* and an *item embedding* that are used to feed a classifier which generates the preference estimation.

Our architecture is based on a *lookup table* to represent users and items. Given a set of elements A, each of them can be represented as a d-dimensional vector contained in a $W \in \mathbb{R}^{|A| \times d}$ embedding matrix. Let $e_j \in \mathbb{R}^{|A|}$, which is all zeros except for the j-th index in which there is the value 1. For each $a \in A$ we assign an index j, $1 \leq j \leq |A|$. The embedding $v(e_j)$ associated to the element a is given by $e_j^{\intercal} W$.

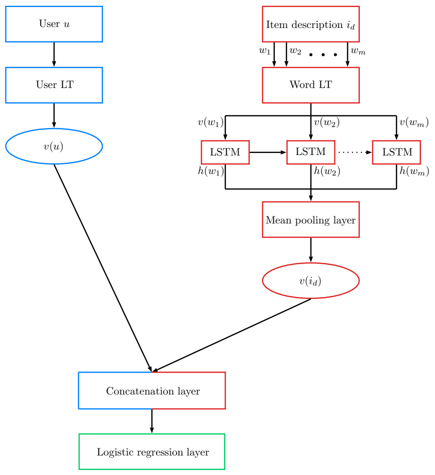
Figure 1a shows the *AMAR* architecture. Each user u is given to a *user lookup table (User LT)* to obtain a learnt d_u-dimensional *user embedding* $v(u)$. Each word $w_1, w_2, \ldots, w_m$ of the item description i_d associated to the item i is given in input to a *word lookup table (Word LT)* which generates a d_w-dimensional embedding $v(w_k)$ for each word w_k, $1 \leq k \leq m$. These word representations $v(w_k)$ are sequentially passed through an *LSTM network* [1] which generates a d_{i_d}-dimensional latent representation $h(w_k)$ for each of them. The *item embedding* $v(i_d)$ is obtained by a *mean pooling layer* which averages the latent representations $h(w_k)$.

The resulting representations $v(u)$ and $v(i_d)$ are concatenated through a *concatenation layer* obtaining a $(d_u + d_{i_d})$-dimensional feature vector given in input to a *logistic regression layer* to predict the score $s(u, i)$. The list of recommendations for a given user u is generated sorting items in decreasing order by the score $s(u, i)$ for each item i.

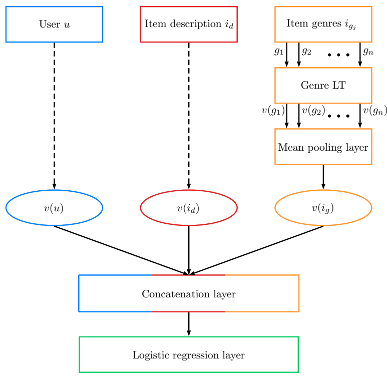
AMAR can be extended by adding an additional module to process supplementary features associated to each item. The *AMAR* extended architecture proposed in this work associates a list of genres $g_1, g_2, \ldots, g_n$ to each item, as shown in figure 1b. Each of them is passed in input to a *genre lookup table (Genre LT)* which generates a d_g-dimensional embedding $v(g_k)$ for each genre g_k, $1 \leq k \leq n$. A *mean pooling layer* averages the resulting representations $v(g_k)$ giving a *genres embedding* $v(i_g)$ which is concatenated to the user and item embeddings to evaluate the recommendation score.

3 Experimental evaluation

In the experimental evaluation the performance of *AMAR* architectures is compared on top-n recommendation leveraging binary user preferences against two



(a) *AMAR* base



(b) *AMAR* extended

state-of-the-art datasets as *Movielens 1M (ML1M)* [1] using 5-fold cross validation and *DBbook* [2] using holdout evaluation, as in [3]. *ML1M* user preferences are binarized setting to 1 all ratings equal or greater than 4. Textual content and genres are obtained from *DBpedia*, *IMDb* and *Goodreads*. The produced recommendation list is evaluated according to F_1*-measure* considering only items in the test set rated by each user using *RiVal* framework [3]. The results are validated using *Wilcoxon signed-rank test*.

Table 1 shows *AMAR* performance against state-of-the-art algorithms (the best configurations are marked in bold) as *User-to-User (U2U)* and *Item-to-Item (I2I)* collaborative filtering, *Bayesian Personalized Ranking Matrix Factorization (BPRMF)*, *Weighted Regularized Matrix Factorization (WRMF)*, *Sparse Linear Methods with BPR-Opt (BPRSlim)* [4] and *Vector Space Model* with *TF-IDF weighting scheme (TF-IDF)* [5]. Additional algorithms are proposed using pre-trained word embeddings like *Word2vec Google News* [6] and *GloVe Wikipedia 2014 + Gigaword 5*[7] (# dimensions = 300). The recommendation score is generated using the *cosine similarity* between the item represented by averaging word embeddings of its description and the user profile represented by averaging positively rated items representations.

AMAR architectures are trained using *RMSprop* ($\alpha = 0.9$, learning rate $= 0.001$) for 25 epochs by optimizing the *binary cross-entropy* criterion. Embedding sizes are fixed to 10. Batch sizes are 1536 for *ML1M* and 512 for *DBbook*. *U2U*

[1] `http://grouplens.org/datasets/movielens/1m/`
[2] `http://challenges.2014.eswc-conferences.org/index.php/RecSys`
[3] `http://rival.recommenders.net/`
[4] Implementations provided by `http://www.mymedialite.net/`
[5] Implementation provided by `http://scikit-learn.org/`
[6] `https://code.google.com/archive/p/word2vec/`
[7] `http://nlp.stanford.edu/projects/glove/`

and *I2I* neighborhood sizes are 30, 50, 80. *BPRMF* and *WRMF* number of factors are 10, 30, 50. The configuration with the highest average F_1-measure among the chosen cut-off is reported (*I2I-30*, *U2U-30*, *BPRMF-30*, *WRMF-50* on both datasets).

ML1M	**AMAR base**	**AMAR ext**	**I2I**	**U2U**	**WRMF**	**BPRMF**	**BPRSlim**	**Word2vec**	**GloVe**	**TF-IDF**
F1@5	0.555	**0.558**	0.432	0.427	0.425	0.425	0.446	0.5	0.485	0.5
F1@10	0.641	**0.644**	0.527	0.525	0.525	0.524	0.548	0.587	0.575	0.59
F1@15	0.64	**0.642**	0.541	0.539	0.539	0.537	0.561	0.591	0.581	0.594
DBbook										
F1@5	0.564	**0.565**	0.536	0.536	0.519	0.508	0.511	0.548	0.552	0.564
F1@10	**0.662**	**0.662**	0.64	0.639	0.636	0.631	0.632	0.656	0.655	0.662
F1@15	**0.611**	**0.611**	0.595	0.595	0.595	0.595	0.595	0.61	0.61	0.611

Table 1: Results of the experiments

Using a p-value of 0.05, on the *DBbook* dataset all the differences are statistically significant, instead on the *ML1M* dataset the differences between *U2U* and *GloVe*, *BPRSlim* and *GloVe*, *GloVe* and *Word2vec* are not statistically significant. All the others differences are statistically significant.

4 Conclusions and Future Work

According to the presented results, *AMAR* architectures outperform all the other recommenders on the *ML1M* dataset. On *DBbook* there is not an improvement which is probably due to the very high sparsity of the dataset.

This preliminary work can be extended in different ways, such as improving training by applying early stopping and regularization techniques, by using different weight initialization strategies and by designing a proper cost function for top-n recommendation. An additional improvement could be obtained by doing hyperparameter optimization.

5 Acknowledgments

This work is supported by the *IBM Faculty Award "Deep Learning to boost Cognitive Question Answering"*. The Titan X GPU used for this research was donated by the *NVIDIA Corporation.*

References

1. A. Graves, A. Mohamed, and G. E. Hinton. Speech recognition with deep recurrent neural networks. *CoRR*, 2013.
2. Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 2015.
3. C. Musto, G. Semeraro, M. de Gemmis, and P. Lops. Learning word embeddings from wikipedia for content-based recommender systems. In *ECIR proceedings*, 2016.
4. J. Weston, A. Bordes, S. Chopra, and T. Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, 2015.