Learning to Rank Tweets with Author-Based Long Short-Term Memory Networks

Guangyuan Piao(✉) and John G. Breslin

Insight Centre for Data Analytics,
National University of Ireland, Galway, Ireland
guangyuan.piao@insight-centre.org, john.breslin@nuigalway.ie

Abstract. Recommending tweets that a user might retweet plays an important role either in satisfying users' information needs or in the dissemination of information in microblogging services such as Twitter. In this paper, we propose a deep neural network for tweet recommendations with author-based Long Short-Term Memory networks for learning the latent representations/embeddings of tweets. Our approach predicts the preference score of a tweet based on (1) the similarity between the embeddings of a user and the tweet, (2) the similarity between the embeddings of the user and the author (who posted the tweet). Despite its simplicity, we present that our approach can significantly outperform state-of-the-art methods with or without explicit features for recommending tweets in terms of five evaluation metrics.

1 Introduction

Twitter[1] users are often overwhelmed by the large number of tweets from their followees, and this makes it difficult for users to consume information relevant to their interests. Recommending tweets that a user might be interested in (or might be retweeted by the user) plays an important role in dealing with the information overload. Deep learning techniques such as Convolutional Neural Networks (CNNs) [7] for processing data in the form of multiple arrays, or Recurrent Neural Networks (RNNs) for tasks with sequential inputs, have been widely adopted in various research domains such as computer vision, natural language processing, etc. Motivated by the state-of-the-art performance achieved by deep learning approaches in different research problems, researchers have started leveraging deep neural networks for several tasks in online social networks such as retweet prediction [13] or hashtag recommendations on Twitter [8]. However, leveraging Long Short-Term Memory neural networks (LSTMs) [3] for tweet recommendations is still relatively unexplored. Also, to the best of our knowledge, none of previous studies have incorporated the author information when learning the latent representation of a tweet. In particular, we combine the author and the

[1] https://twitter.com/.

T. Mikkonen et al. (Eds.): ICWE 2018, LNCS 10845, pp. 288–295, 2018.
https://doi.org/10.1007/978-3-319-91662-0_22

content of a tweet to learn the tweet embedding with LSTMs instead of relying only on the content. That is, for two tweets with the same texts but from different authors, our approach learns different latent representations for those tweets while previous approaches do not. The intuition behind this idea is that tweets such as "*our paper got accepted at @ICWE2018*" should have different latent representations based on who tweeted them. While leveraging deep learning approaches is able to learn a good latent representation of a given tweet, it still lacks contextual information which can be provided by explicit features (e.g., the popularity of the author of a tweet). This information might play an important role in tweet recommendations, which we will show in Sect. 4.2. **Our contributions in this paper are as follows**: (1) We propose a deep neural network for tweet recommendations, which leverages author-based LSTMs for learning tweet embeddings (Sect. 3); (2) We evaluate our approach by comparing it with deep learning approaches (Sect. 4.1) and with other approaches in the context of leveraging explicit features (Sect. 4.2).

2 Related Work

A popular line of approaches for tweet recommendations is using feature-aware factorization approaches [1,2,4]. For example, Collaborative Tweet Ranking (`CTR`) [1] is a factorization model, which predicts the preference score of a tweet based on explicit features and the relationships between the latent representations with respect to users, authors, and words in the tweet. The difference between our approach and those factorization models is that they treat tweets as bags of word embeddings, and therefore, the sequence of words is not preserved. Another line of work has focused on learning useful features for predicting the retweetability of a tweet [12,13] such as the large scale study conducted by [12]. More recently, [13] proposed several CNN-based models for retweet prediction, and showed that a CNN-based approach with user and author embeddings outperforms a full model with an additional attention-based neural network [6] in terms of precision, and outperforms other baselines. However, tweet embeddings have been learned based on their content only without considering their author information. Also, a comparison with previous studies leveraging explicit features was not conducted, and the contribution of explicit features in the context of using deep learning techniques is unclear.

3 Proposed Approach

The basic idea of our approach is to compute the similarity between a target user u and a given tweet t, and the similarity between u and the author of t, and predict the preference score of t based on those similarities. Our approach is one of the Deep Semantic Similarity Model-based recommendation approaches (DSSMs), which has shown to be extremely suitable for top-N recommendation tasks [14]. In short, DSSMs project different entities into a common latent space for computing their similarities with similarity measures such as cosine similarity.

3.1 Author-Based Long Short-Term Memory Networks for Tweets

With the ability to learn long-term memory dependencies, LSTMs have been widely used in many natural language processing problems such as machine translation. In this work, we use author-based LSTMs to learn the latent representations for tweets. To this end, we first represent a tweet as a sequence of words where stopwords and URLs are removed. Next, we convert these words to a matrix in order to use it as an input to LSTMs: $mt = [x_1, \cdots, x_m]$, where i-th column of mt denotes the embedding of a word x_i in a tweet.

LSTMs. Similar to other RNNs, LSTM also has the form of a chain of repeating components in a neural network called the *memory cell*. The key to an LSTM is a *cell state* in the memory cell which works as a conveyor belt, and controls the information flow with some minor linear interactions through the entire chain. The LSTM can remove or add information to the cell state via *gates*, which consists of a sigmoid layer and a pointwise multiplication operation. The output value of a sigmoid layer ranges from 0 to 1, which represents how much information should be let through. There are three steps in each LSTM. The first step is going through a *forget gate layer* which decides what information to keep from the cell state by looking at h_{t-1} and x_t, as shown below:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{1}$$

where $[h_{t-1}, x_t]$ denotes the concatenated vector of h_{t-1} and x_t, σ is a sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$, and b_f denotes a bias term. W_f is a weight vector to be learned for the forget gate layer.

Next, LSTM decides what new information to store in the cell state, which consists of two parts. The first part is an *input gate layer*:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$

The second part is a tanh layer (Eq. 3), which creates a vector of new candidate values, $\tilde{C}_t$, that could be added into the cell state.

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3}$$

Finally, the new cell state C_t will be created based on linear interactions of the previous cell state C_{t-1}, f_t, i_t, and $\tilde{C}_t$ as follows.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{4}$$

The last step is filtering the cell state to generate the final output, which can be formulated as below:

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \\ h_t &= o_t * \tanh(C_t) \end{aligned} \tag{5}$$

Here, o_t decides what parts of the cell state to keep for the final output, and the cell state goes through a tanh layer before multiplying by o_t.

Incorporating author information. The output of LSTMs based on the content of a tweet is the latent representation/embedding of a tweet. However, considering the content of a tweet only for LSTMs will produce the same latent representations for the tweets with the same content, even if these tweets are posted by different authors. Our assumption is that the representation of a tweet should reflect the content of the tweet as well as the author. This means that tweets with the same content should have different representations if the authors of these tweets are different. To this end, we incorporate the author information to learn tweet embeddings using LSTMs. That is, we use the author embedding v_a with respect to a tweet t as an initial input to LSTMs instead of the first word in t to learn the embedding for t, and therefore, a tweet matrix is changed to $mt = [v_a, x_1, \cdots, x_m]$. The final output of author-based LSTMs is used as the final representation of t, which is denoted as v_t.

Given the latent representation of a tweet (v_t) learned from the author-based LSTMs, we then measure the similarities between user, the tweet, and the author of it. Let v_u denote the embedding of a target user u. The final concatenation layer v_{concat} consists of two similarity scores s_{ua} and s_{ut}. s_{ua} denotes the *cosine* similarity between v_u and v_a: $s_{ut} = \frac{v_u \cdot v_a}{\|v_u\|_2 \|v_a\|_2}$, and s_{ut} denotes the *cosine* similarity between v_u and v_t. Finally, the preference score of t is measured as $score = W \cdot v_{concat}^T + b$, where W denotes the weights for each element in the concatenation layer, and b denotes a bias term.

Regularization. We use the dropout [11] for regularization, which refers to dropping out units in a neural network, is one of the widely used regularization techniques for preventing overfitting in training neural networks. Individual nodes are either "disabled" with probability $1 - p$ or kept with probability p. The "thinned" outputs of a hidden layer are then used as an input to the next layer. We also constrain l_2-norms of the weight vectors to a threshold ϵ as follows, $\|w\|_2 = \epsilon$, $if\ \|w\|_2 > \epsilon$, which normalizes a word vector w so that its l_2-norm is equal to ϵ, and will be performed whenever the l_2-norm of w is bigger than ϵ.

Training. We use a *pairwise* approach, Bayesian Personalized Ranking (BPR) [10], for formulating the loss function, which has been widely used for ranking problems. The intuition behind BPR is that the preference score of a retweeted tweet should be higher than that of a tweet which has not been retweeted. The loss function can be defined as below:

$$\ell(x_1, x_2) = \sum_{x_1 \in \mathcal{X}^+} \sum_{x_2 \in \mathcal{X}^-} -\log[\sigma(\hat{y}(x_1) - \hat{y}(x_2))]$$

where σ is a sigmoid function, $\hat{y}(\cdot)$ denotes the prediction score obtained by our proposed approach, and $\mathcal{X}^+$ and $\mathcal{X}^-$ are the sets of positive and negative training instances, respectively. The objective of training is to learn parameters that minimize the loss. To learn the parameters for minimizing the loss, we use a Stochastic Gradient Descent (SGD) with the Adam update rule [5] to train the model until the loss has converged.

4 Evaluation

We use a Twitter dataset from [9] for our experiment. This dataset consists of 480 random users and their tweets. In order to obtain explicit features for our experiment (Sect. 4.2), we used the Twitter API[2] to crawl additional information about users and authors such as the count of followees/followers. We removed the users that did not exist on Twitter at the time of crawling. For each user, we first sorted the tweet stream of their followees in chronological order. For each user, the latest 25% of followees' tweets were considered as the test set, and the rest of their tweets were used as the training set. The users that have less than one retweet in the training or test set were removed. In addition, we limited five previous and five subsequent tweets of a retweeted tweet to construct the final training and test sets. This results in ten negative instances for each retweeted tweet. Finally, the dataset consists of 307 users with 71,039 retweets, 710,390 training instances in the training set, and 18,544 retweets in the test set.

Evaluation Metrics. To evaluate the performance of tweet recommendations, we use five widely used evaluation metrics: MAP (Mean Average Precision), MRR (Mean Reciprocal Rank), P@N (Precision at the top-N recommendations), R@N (Recall), and nDCG@N (Normalized Discounted Cumulative Gain) where N = 1, 5, 10, respectively. We use the *bootstrapped paired t-test* for testing the significance where the significance level $\alpha = 0.05$.

4.1 Experiment 1

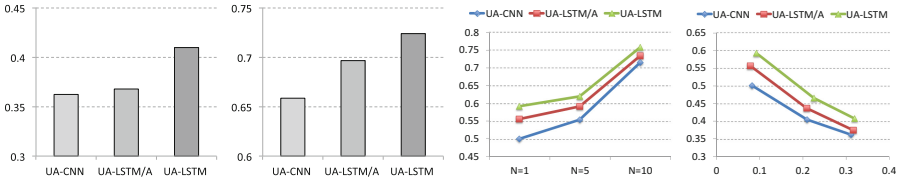
In the first experiment, we aim to (1) compare our approach against a CNN-based approach proposed recently, and (2) investigate whether incorporating the author information for learning the latent representations of tweets with LSTMs improves the recommendation performance.

- `UA-CNN` [13]: This method uses CNNs for learning the latent representation of a tweet, and concatenate the tweet representation with user and author embeddings for predicting the preference score of the tweet. We re-implemented a BPR formulation of this approach for our experiment as the original model proposed in [13] is for the retweet prediction task with a *point-wise* loss function which is not optimal for tweet recommendations.
- `UA-LSTM/A`: This baseline denotes `UA-LSTM` without incorporating the author embedding when learning the latent representation of a tweet. Therefore, it does not distinguish the same tweets from different authors.

Parameter Settings. The dropout rate p is set to 0.5, the l_2-norm threshold is set to 3, and the mini-batch size is set to 40. As some improvements might be due to the difference of embedding sizes for compared models, we set the dimensionality of all embeddings to 300 in the same way as [13].

[2] https://developer.twitter.com/.

Results. Figure 1 shows the recommendation performance based on `UA-LSTM` and other methods compared. As we can see from the figure, `UA-LSTM` significantly outperform both `UA-CNN` as well as `UA-LSTM/A`. Overall, both approaches using LSTMs outperform `UA-CNN`, which indicates the efficiency of LSTMs for tweet recommendations. The improvement of `UA-LSTM` over `UA-LSTM/A` in terms of all evaluation metrics clearly indicates that incorporating author information for learning the latent representation of a tweet improves the performance.



(a) MAP (b) MRR (c) nDCG@N (d) P@N (y) R@N (x) curve

Fig. 1. Recommendation performance based on `UA-LSTM` and two other deep learning approaches.

4.2 Experiment 2

In the second experiment, we are interested in our approach compared to other approaches relying on explicit features. Instead of "reinventing the wheel", we use a comprehensive set of features used in the literature [1,4] as our focus here is understanding the contribution of explicit features with respect to tweet recommendations in the context of deep neural networks rather than proposing new features. The detailed descriptions of features can be found in [1,4]. In short, these features aim to capture three factors: information about (1) the user themselves, (2) the tweet author, (3) the tweet content, and the relationships between these factors. The compared baselines for this experiment are as below:

- `RTCount`: This is a baseline method ranks tweets based on their retweet count.
- `RankSVM`: `RankSVM` is a widely used learning-to-rank model with explicit features, which transforms the ranking problem into a classification problem.
- `CTR` [1]: This is a state-of-the-art feature-aware factorization model for predicting the preference score of a candidate tweet. We re-implemented this method with the same parameter settings as in [1].
- `UA-CNN-F`. This method adds the explicit features into the concatenation layer of `UA-CNN` for final prediction.
- `UA-LSTM-F`. This is our proposed approach with those explicit features (f_e) in the concatenation layer, i.e., $v_{concat} = [s_{ua}, s_{ut}, f_e]$.

Results. Table 1 shows the tweet recommendation results based on different approaches with explicit features. Overall, our `UA-LSTM-F` which incorporates explicit features provides the best performance while the straightforward baseline

Table 1. The recommendation performance of our proposed approach and other methods with explicit features.

	RTCount	RankSVM	CTR	UA-CNN-F	UA-LSTM-F	UA-CNN	UA-LSTM
MAP	0.2513	0.4390	0.4306	0.4331	**0.4778**	0.3623	0.4097
MRR	0.3822	0.6549	0.6568	0.7115	**0.7290**	0.6585	0.7238
nDCG@1	0.2280	0.4756	0.4658	0.5570	0.5733	0.5016	**0.5928**
nDCG@5	0.3230	0.5764	0.5511	0.6196	**0.6412**	0.5552	0.6206
nDCG@10	0.4401	0.7312	0.7289	0.7644	**0.7823**	0.7152	0.7577
P@1	0.2280	0.4756	0.4658	0.5570	0.5733	0.5016	**0.5928**
P@5	0.1941	0.4534	0.4098	0.4808	**0.4984**	0.4059	0.4664
P@10	0.1674	0.4104	0.3879	0.4228	**0.4453**	0.3635	0.4088
R@1	0.0485	0.0798	0.0928	0.0920	**0.0958**	0.0822	0.0921
R@5	0.1550	0.2547	0.2468	0.2461	**0.2744**	0.2091	0.2250
R@10	0.2420	0.3621	0.3537	0.3570	**0.3695**	0.3106	0.3188

method RTCount achieves the worst performance. Both UA-CNN-F and UA-LSTM-F outperform pure deep learning approaches without explicit features, i.e., UA-CNN and UA-LSTM. This indicates that although the recommendation performance achieved with deep learning approaches is promising, explicit features still play an important role in boosting the performance. One of the possible explanations might be that using deep learning approaches based on retweets only cannot capture other contextual information which can be complemented by explicit features, e.g., the popularity of authors based on features such as # of followers/followees. We also observe that UA-LSTM provides comparable results (i.e., no statistical difference) to UA-LSTM-F in terms of some evaluation metrics such as MRR, nDCG@1, P@1, and R@1. This suggests that UA-LSTM performs well on ranking the first relevant tweet higher, and on top-1 recommendation.

Compared to UA-CNN-F, our proposed approach consistently outperforms this baseline method. A significant improvement of UA-LSTM-F over UA-CNN-F in MAP (+10%), P@10 (+5%), and R@5 (+11.5%) can be noticed. Regarding recall, CTR provides the best performance in terms of R@1, and RankSVM provides the best performance in terms of R@5 and R@10 among the baseline methods. UA-LSTM-F performs consistently better than CTR and RankSVM in terms of recall. Furthermore, UA-LSTM-F outperforms CTR significantly in terms of R@5 and R@10, and outperforms RankSVM significantly in terms of R@5. Despite of its simplicity, the results indicate the efficiency of our proposed approach compared to other approaches with explicit features.

5 Conclusions

In this paper, we presented a novel deep neural network architecture for tweet recommendations. The approach employs author-based LSTMs for learning

tweet embeddings, and predicts the preference scores of tweets based on the similarity between user and author embeddings as well as the similarity between user and tweet embeddings. Results show that our proposed approach can outperform other deep learning based baselines as well as the approaches with explicit features. In addition, the overall improvements of `UA-LSTM-F` and `UA-CNN-F` over `UA-LSTM` and `UA-CNN` indicate that explicit features still play an important role in the context of tweet recommendations.

References

1. Chen, K., Chen, T., Zheng, G., Jin, O., Yao, E., Yu, Y.: Collaborative personalized tweet recommendation. In: Proceedings of the 35th ACM SIGIR Conference, SIGIR 2012, pp. 661–670, vol. 800. ACM, Portland (2012)
2. Feng, W., Wang, J.: Retweet or not?: personalized tweet re-ranking. In: Proceedings of the Sixth ACM WSDM Conference, pp. 577–586. ACM (2013)
3. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
4. Hong, L., Doumith, A.S., Davison, B.D.: Co-factorization machines: modeling user interests and predicting individual decisions in Twitter. In: Proceedings of the Sixth ACM WSDM Conference, pp. 557–566. ACM (2013)
5. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
6. Larochelle, H., Hinton, G.E.: Learning to combine foveal glimpses with a third-order Boltzmann machine. In: Lafferty, J.D., Williams, C.K.I., Shawe-Taylor, J., Zemel, R.S., Culotta, A. (eds.) Advances in Neural Information Processing Systems, vol. 23, pp. 1243–1251. Curran Associates, Inc. (2010)
7. LeCun, Y., Boser, B.E., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.E., Jackel, L.D.: Handwritten digit recognition with a back-propagation network. In: Advances in Neural Information Processing Systems, pp. 396–404 (1990)
8. Li, J., Xu, H., He, X., Deng, J., Sun, X.: Tweet modeling with LSTM recurrent neural networks for hashtag recommendation. In: 2016 International Joint Conference on Neural Networks (IJCNN), pp. 1570–1577. IEEE, July 2016
9. Piao, G., Breslin, J.G.: Analyzing aggregated semantics-enabled user modeling on Google+ and Twitter for personalized link recommendations. In: User Modeling, Adaptation, and Personalization, pp. 105–109. ACM, Halifax (2016)
10. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, pp. 452–461. AUAI Press (2009)
11. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
12. Suh, B., Hong, L., Pirolli, P., Chi, E.H.: Want to be Retweeted? Large scale analytics on factors impacting retweet in Twitter network. In: 2010 IEEE Second International Conference on Social Computing, pp. 177–184. IEEE, August 2010
13. Zhang, Q., Gong, Y., Wu, J., Huang, H., Huang, X.: Retweet prediction with attention-based deep neural network. In: Proceedings of the 25th ACM CIKM 2016, pp. 75–84. ACM (2016)
14. Zhang, S., Yao, L., Sun, A.: Deep learning based recommender system: a survey and new perspectives. CoRR abs/1707.0 (2017)