# Incorporating Metadata into Content-Based User Embeddings

Anonymous EMNLP submission

## Abstract

Low-dimensional vector representations of social media users can benefit applications like recommendation systems and user attribute inference. Recent work has shown that user embeddings can be improved by combining different types of information, such as text and network data. We propose a data augmentation method that allows novel feature types to be used within off-the-shelf embedding models. Experimenting with the task of friend recommendation on a dataset of 5,019 Twitter users, we show that our approach can lead to substantial performance gains with the simple addition of network and geographic features.

## 1 Introduction

A variety of social media tasks benefit from having dense vector representations of users. For example, "who to follow" recommendations can be done by calculating cosine similarity between user vectors. Recent work has experimented with neural embeddings of social media users, most commonly based on text content (Amir et al., 2016; Wan et al., 2016), with some work combining input features from other metadata, including social network information (Li et al., 2015; Benton et al., 2016; Yang et al., 2016).

Since social media like Twitter provide different types of data (e.g., text, network, location), constructing user embeddings with appropriate features can improve the performance based on the target task's requirements. For instance, for recommending tweets a user may be interested in, the user's text content will be a crucial feature. For recommending users to follow, information about the user's current follow graph is likely to be important.

How to efficiently integrate diverse types of features into user representations is a challenge we address in this work. While "multiview" models that combine different feature types have been proposed for user embeddings, there is cost in adapting any particular model to a multiview setting. As an alternative, we propose a simple solution that treats all discrete features as "words," but preprocesses the data in a way that removes word order effects from non-textual features. This approach can be applied to most models of text content, such as the popular *paragraph2vec* model (Le and Mikolov, 2014), allowing diverse features without constructing specialized models.

Our primary contributions are as follows:

- We describe a preprocessing step that allows the inclusion of non-textual discrete features (e.g., followers, locations) into off-the-shelf text embedding methods. Our method is easy to use, requiring no special implementation.
- We introduce a novel type of feature for user embeddings—the geographic locations of users' friends—and show that this improves performance over standard text and network features on a new Twitter dataset.
- We find that jointly modeling all types of features improves performance over combining independent models of different feature sets, offering evidence that there are informative interactions between text content and metadata, and demonstrating that simply combining independent models is insufficient.

## 2 Previous Work

A number of recent studies have proposed models for constructing social media user embeddings.

Amir et al. (2016) generated user embeddings capture users' individual word usage patterns with a model similar to paragraph2vec. In this method, only users' tweets are taken into account. Wan et al. (2016) proposed two neural network models, also based on paragraph2vec, to obtain users' vector representations from word representations obtained previously. Since their task was recommending tweets to users, only text was considered to construct user embeddings.

Other work has considered multiple types of features, or "views." Benton et al. (2016) proposed an approach based on Weighted Generalized Canonical Correlation Analysis (WGCCA) to turn several aspects of user information into low-dimensional vectors, including tweets and social network information. Before applying the WGCCA model, each type of information is first converted into an appropriate vector representation. Yang et al. (2016) considered text, social relationships and mentioned entities as features for user embeddings. This work used different models for learning each feature type. While trained separately, once a representation was learned for each feature type, a final user representation was learned with a composition model that included additional parameters to learn interactions between feature types. Li et al. (2015) proposed a similar approach, which uses different models for different types of user information, which are then linearly combined into a full model. During training, the parameters are learned jointly.

The multiview models above all used different view-specific models to capture the feature types. In contrast, we use a simple input representation that can be plugged into a single model.

## 3 Social Media Dataset

To motivate our methods, we will first describe our dataset of over 5,000 Twitter users. We randomly sampled users who follow American universities. Specifically, we collected the usernames of up to 5,000 followers of 25 universities (the top 25 undergraduate programs ranked by US News). Among the 5,000 followers of each university, we randomly sampled 400 users, for a total of 10,000 users. After removing accounts that were private or non-English, we were left with 5,019 users.

From each user, we collected their 200 most recent tweets (collected January 2017), as well as the usernames and locations of up to 100 followees by collecting the profiles of 100 randomly sampled accounts that are followed by the user. Our dataset contained an average of 155.4 tweets per user (with an average of 6.7 tokens per tweet, after pre-processing), and an average of 91.9 followees per user (with an average of 32.0 followees for whom we resolved a location).

### 3.1 Types of Metadata

In this work, we will train embeddings using two types of features in addition to the tweet content of each user: the **users** they follow, and the geographic **locations** of the users they follow. These features were selected to support our experimental task of friend recommendation (Section 5).

The motivation for the first type of feature is that if two users' followee lists have substantial overlap, then it indicates they are more likely to have a connection. We implement this by including the usernames of the accounts that each user follows.

The motivation for the second type of feature, with which we opted to use the locations of each user's followees rather than the the user's own location, is perhaps less obvious. First, this attribute is sparse (fewer than half of the users had a valid location), so including their followees' locations provides more information. Second, in many cases, friends' locations may be a more informative predictor of relationships. For example, suppose Users A, B, and C live in Kansas. Many of A and B's friends are located in California, but most C's friends are from New York. In this scenario, A and B may be more likely to have a relationship than C, while if we only used the users' own locations, then location would not differentiate them.

To extract high-precision locations, we extracted only locations from user profiles of the form "City, State", where the state had to match a dictionary of US states. We used the dictionary to rewrite state names in a canonical form (e.g., "California" $\rightarrow$ "CA").

## 4 User Embedding Model

This work uses paragraph2vec (Le and Mikolov, 2014) as our content embedding model. This is an unsupervised model that encodes text sequences (canonically, paragraphs) as low-dimensional vectors. The model is related to word2vec (Mikolov et al., 2013), with a modification that each paragraph is given a unique paragraph token at the beginning, which is treated like other tokens in the

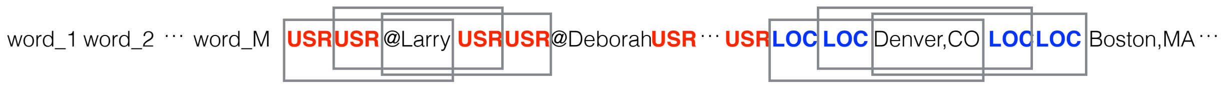
word_1 word_2 ··· word_M
USR USR @Larry USR USR @Deborah USR ··· USR LOC LOC Denver,CO LOC LOC Boston,MA ···

Figure 1: We augment text features by concatenating each user's input string with username and location features, padded with "dummy" tokens to prevent word order effects for non-textual features. Since each token's probability in paragraph2vec depends on the $k$ tokens before and $k$ tokens after it, we separate the username and location tokens with $k$ dummy 'USR' and 'LOC' tokens so that no usernames or locations will appear in the same window.

paragraph to help predict words. The result is a vector for each paragraph, in addition to the word vectors for each word. In our setting, we treat each user's concatenated stream of tweets as a single "paragraph" and apply the model as is. The training objective for each user $u$ under this model is:

$$\frac{1}{M}\sum_{i=1}^{M}\log P(w_i|u, w_{i-k}, \cdots, w_{i+k}) \quad (1)$$

where $M$ is the number of tokens in the user stream and $k$ is the window size. $u$ is the vector representation of the "paragraph" token that uniquely corresponds to the user, and $\boldsymbol{w}$ are the word tokens in the tweet stream.

### 4.1 Incorporating Metadata

We propose to add additional metadata features by simply appending the text sequences (the user tweet streams) with additional tokens for each username and each location string, representing the features described in Section 3.1. However, doing this naively will not work as intended, because the order of the word tokens within each window affects the probabilities, which is not appropriate for features that have no ordering.

To address this, we format the text such that two metadata features never appear within the same window. We pad the features with "dummy" tokens that appear before and after each username ('USR') or location ('LOC'). There are $2\times k$ dummy tokens in between each feature, where $k$ is the window size, as illustrated in Figure 1.

A side effect of this approach is that there will be redundant metadata features from different window positions that will always co-occur (e.g., "LOC Seattle LOC" and "LOC LOC Seattle"). This creates colinearity in the model, but this does not diminish the predictive performance, nor is this unusual when applying machine learning to text (e.g., including different length $n$-grams).

## 5 Experiments

To evaluate our proposed approach, we experiment with different vector representations of our dataset for the task of friend recommendation (Liben-Nowell and Kleinberg, 2007; Lo and Lin, 2006; Backstrom and Leskovec, 2011).

### 5.1 Implementation Details

We preprocessed the text to remove punctuation, stop words, hyperlinks, and usernames. All tweets were concatenated into one long string. Additional features were concatenated to the same string using the token padding procedure described in Section 4.1.

We used the paragraph2vec implementation from the Python package, Gensim (Řehůřek and Sojka, 2010), called `doc2vec`. The window size and vector dimensionality were set to 3 and 100.

### 5.2 Experimental Design

For friend recommendation, we calculate the cosine similarity between all pairs of users based on their vector representation. For each user, we select the top $k$ users with the highest similarity. Using the user's followee list as ground truth, we measure the precision, recall, and mean reciprocal rank for $k$ in $\{10, 20, 30, 40, 50\}$.

Similar to the design of (Benton et al., 2016), we used the most popular accounts as our test set. Specifically, we selected 58 users who were followed by more than 50 users in our dataset for evaluation. These users were excluded when generating the username features described in 3.1.

We experimented with our paragraph2vec-based embedding models with different feature sets. We consider the model with only text features to be our primary baseline, which we compare against our model that adds username features as well as both username and location features.

For our full model with all three feature types, we compared three different approaches for com-

| | k = 10 | | | k = 20 | | | k = 30 | | | k = 40 | | | k = 50 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P | R | MRR | P | R | MRR | P | R | MRR | P | R | MRR | P | R | MRR |
| Random | .022 | .003 | .056 | .030 | .010 | .070 | .046 | .014 | .083 | .029 | .019 | .052 | .021 | .018 | .053 |
| TF-IDF | .174 | **0.019** | .192 | .160 | .033 | .152 | .137 | .039 | **0.131** | .120 | .046 | **0.118** | .110 | .051 | .104 |
| Text only (T) | .133 | .013 | .205 | .121 | .027 | .159 | .116 | .033 | .127 | .111 | .042 | .116 | .108 | .051 | .104 |
| Text+Users (T+U) | .168 | .016 | .210 | .158 | .031 | .166 | .148 | .042 | .129 | .140 | .053 | .108 | .135 | .065 | .108 |
| T+U+L (Addition) | .131 | .013 | .135 | .121 | .027 | .123 | .131 | .038 | .092 | .115 | .042 | .087 | .120 | .057 | .077 |
| T+U+L (Concat.) | .150 | .014 | .16 | .146 | .031 | .112 | .132 | .038 | .110 | .128 | .050 | .088 | .121 | .061 | .078 |
| T+U+L (Joint) | **.193**$^{\dagger\ddagger}$ | **.019**$^{\dagger}$ | **.227**$^{\ddagger}$ | **.171**$^{\dagger\ddagger}$ | **.033**$^{\dagger}$ | **.174**$^{\ddagger}$ | **.162**$^{\dagger\ddagger}$ | **.046**$^{\dagger}$ | .120$^{\ddagger}$ | **.153**$^{\dagger\ddagger *}$ | **.059**$^{\dagger *}$ | .114$^{\dagger\ddagger}$ | **.150**$^{\dagger\ddagger *}$ | **.071**$^{\dagger\ddagger *}$ | **.110**$^{\dagger\ddagger}$ |

Table 1: Overview of results with precision (P), recall (R), and mean reciprocal rank (MRR) at $k$. This compares our non-embedding baselines with our embedding model using various feature sets: text only (T), text and usernames (T+U), and text with usernames and locations (T+U+L), either jointly modeled or independently combined by addition or concatenation. Markers indicate if the results of our joint T+U+L model are significantly different ($p<0.05$) from the text only (†), concatenated T+U+L (‡), or TF-IDF ($*$) models.

bining the features. In addition to training a single paragraph2vec model on all features jointly, as we proposed in Section 4.1, we also trained three separate paragraph2vec models on the three feature types and then combined them, by adding the vectors in one version and by concatenating the three vectors in another. Since some prior work trained independent models for different views (Section 2), it is important to understand how joint versus independent training affects performance.

Finally, we add two other baselines to put our results in context. First is a random baseline that randomly chooses $k$ users in the ranking, which gives an approximate lower bound on performance. Second, we compare to a high-dimensional bag-of-words representation of text with TF-IDF weighting.

We measured the statistical significance of the results using a paired t-test to compare our full model to the baselines with close performance.

### 5.3 Results and Discussion

Results are shown in Table 1. The full joint model outperforms all others at precision and recall in all cases, and at MRR in a majority of cases.

There is a substantial improvement in the embedding model using all three feature types compared to the model using only text. The differences in precision and recall are highly significant, with p-values $<0.001$ in all cases. This demonstrates that our proposed features are useful features for friend recommendation, and that our proposed method for encoding them in the data is effective.

We also find that training a representation using all three feature types jointly gives significantly better performance than concatenating (which performed better than adding) three independently trained representations.

While our full model outperformed the TF-IDF baseline in most cases, most differences were not significant, and the TF-IDF baseline outperformed the paragraph2vec model using only text. We believe the strong performance of TF-IDF relative to the paragraph2vec representations may be due to the fairly small size of our dataset. Since our goal was to investigate how to improve the quality of embeddings, we primarily focus on the comparison other embedding models rather than TF-IDF, but we present these results to show how the different representation types compare on this dataset.

## 6 Conclusion

We have described and evaluated a simple method for adding non-textual discrete features into a text embedding model for constructing embeddings of social media users. We constructed a novel geographic feature—the locations of a user's friends—and showed that the addition of network and geographic features significantly improves precision and recall over a text-only baseline up to $0.06$ and $0.02$, respectively. We also showed that including all feature types in one model leads to significantly better performance than combining independently trained models of different features. Our approach is based on modifying the input rather than the model, therefore requiring no special implementation and can be easily adapted to other embedding models or feature types in future work.

## References

Silvio Amir, Byron C. Wallace, Hao Lyu, Paula Carvalho, and Mario J. Silva. 2016. Modelling context with user embeddings for sarcasm detection in social media. arXiv preprint arXiv:1607.00976.

Lars Backstrom and Jure Leskovec. 2011. Supervised random walks: Predicting and recommending links in social networks. In *Fourth ACM International Conference on Web Search and Data Mining (WSDM)*.

Adrian Benton, Raman Arora, and Mark Dredze. 2016. Learning multiview embeddings of twitter users. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 14–19.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. Proceedings of The 31st International Conference on Machine Learning, pages 1188–1196.

Jiwei Li, Alan Ritter, and Dan Jurafsky. 2015. Learning multi-faceted representations of individuals from heterogeneous evidence using neural networks. arXiv preprint arXiv: 1510.05198.

David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology* 58(7):1019–1031.

Shuchuan Lo and Chingching Lin. 2006. WMR–a graph-based algorithm for friend recommendation. In *IEEE/WIC/ACM International Conference on Web Intelligence*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, pages 3111–3119.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. pages 45–50.

Xiaojun Wan, Yang Yu, and Xinjie Zhou. 2016. User embedding for scholarly microblog recommendation. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 449–453.

Yi Yang, Ming-Wei Chang, and Jacob Eisenstein. 2016. Toward socially-infused information extraction: Embedding authors, mentions, and entities. Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1452–1461.