Machine Learning Final Project Final Report

Linzi Xing, Xiaolei Huang, Yoshinari Fujinuma

1 Who did what

(1) Linzi reproduced the baseline based on Weighted General Canonical Corelation Analysis (WGCCA) model and basic views data provided by the original paper's author.
(2) Xiaolei scraped and cleaned data. These included users' Twitter profiles, geolocations, active dates and tweets.
(3) Yoshinari also cleaned and normalized data and made these into standard formation of views. Doing experiment and organized and analyzed results.
What is described above is a brief division of work. All three of us have participated in most parts of this project.

2 Background and motivation

Nowadays, social networks, such as Twitter, are becoming a part of life for many people. Twitter is today the most prominent micro-blogging service available on the Web. With users number explodes, its worth doing some researches and finding some patterns from users part. But how to model and understand user's online behaviors is still in its infancy. So we seek for an efficient way to represent and model users in this project.

Low-dimensional vector representations are widely used for text of words and these embeddings often have nice properties. In the field of natural language processing (NLP), word embeddings models, such as word2vec [2], has shown to improve many NLP tasks. User embeddings might be an option, which maps tweets as well as other social networks information into low dimensions. We treat Twitter as a social network, relation and metadata are rich information and also need to be concerned and included. Metadata includes some attributes like geolocation users counts and language. Our project is based on the previous paper [1] which has already covered the network relation and tweets to create user embeddings. We are expanding their work by considering more views associated with Twitter users. Figure 1 highlights various views. In this project, we extract more user metadata and add them to user embeddings to see if the performance improves on some downstream tasks.

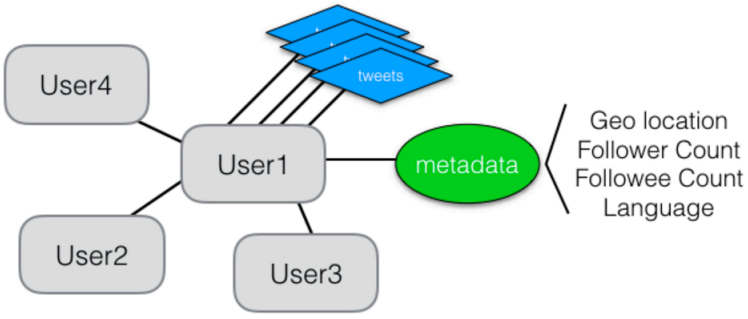

Figure 1: Various views associated with a Twitter user.

3 Baseline

In this section, we will briefly go over the method and data used in [1]. The data provided by the authors[1] of the previous paper contains 6 views:

1. Ego Tweets
2. Mention Tweets
3. Friend (Users that a given user follows) Tweets
4. Follower Tweets
5. Friend Network
6. Follower Network.

Each line of dataset represents a user's views and it is created by concatenating all views together. For tweets views, they are first reduced in dimensionality by projecting its bag-of-words (BOW) TF-IDF weighted representation to a 1000-dimensional vector through Principal Component Analysis (PCA). For the other two network's views, they are both represented by a summation of the one-hot representation of each neighboring user in each user's social network. After that, we apply PCA to project these to 1000-dimensional vector.

After we got data, we input those into WGCCA model to obtain user embeddings. This is a decent way because WGCCA can include multi views at the same time and treat views slightly differently because of the importance difference. The objective function of Generalized Canonical Correlation Analysis (GCCA) is as follows:

$$\arg\min_{G,U_i} \sum_i ||G - X_i U_i||_F^2$$

We minimize the result of this formula and get G and U under the optimal condition. U is the set of matrices which maps from the latent space to observed views. G is a matrix that contains all users representations.

For WGCCA, a weight variable w_i is added for each view i:

$$\arg\min_{G,U_i} \sum_i w_i ||G - X_i U_i||_F^2.$$

We also minimize the result of this formula and get G and U under the optimal condition. Here, "w_i explicitly expresses the importance of the ith view in determining the joint embedding." [1]

We follow the previous paper [1] and established baselines for the 2 tasks. The first task is the hashtag prediction task. First, we get 400 most frequently used hashtags created by our users during March 2015 and then divide them into two equal size datasets (dev and test). Then we pick the top 10 users who already use the hashtag and calculate their average user embeddings to represent the embeddings of hashtags. We rank all other users by computing the cosine distance

[1] Available at `http://www.cs.jhu.edu/~mdredze/datasets/multiview_embeddings/`

between their embeddings and all hashtag embedding got from 10 users. Performance is measured using precision and recall for the top 1000 users ranked according to its cosine similarities, as well as mean reciprocal rank (MRR). The second task is the friend recommendation task. We selected the 500 celebrity accounts and do the same thing, using 10 followers vectors to represent celebrities and rank by cosine similarity, measured by the same evaluation metrics. Here, we represent user embeddings in 20 dimension and 200 dimension to investigate the relation between user dimension and accuracy. The results are as follows:

From these results, it's clear that in most of scenarios, 200 dimensional embeddings can predict

embedding	Precision@1000	Recall@1000	MRR
dimension@20	0.021045	0.619385	0.189527
dimension@200	**0.022640**	**0.663512**	**0.259563**

Table 1: Evaluation results for hashtag prediction

embedding	Precision@1000	Recall@1000	MRR
dimension@20	0.226172	0.072322	**0.470136**
dimension@200	**0.229116**	**0.074218**	0.450480

Table 2: Evaluation results for the friend recommendation task

hashtags and recommend friends more accurate. This is because the higher the vector dimension is, the more granular the represented information is. Note that the authors of the original paper [1] compared their experimental results with naive embedding methods such as taking the average of word embeddings as user embeddings. The baseline result we present here are from multiviews with WGCCA, which is one of the most recent successful methods for user embeddings.

4 Data Collection & Vectorization

The dataset has near 10,000 users and over millions of vectorized tweets content. We collected dataset from Twitter API and request dataset from Adrian, the author of the paper. Due to the Twitter data usage term, we can not request raw dataset from the author. We collected part of the dataset by our own using Twitter API. The dataset we collected contains two parts:

- User profile dataset: we collected around 10,000 users' information, as well as their friend lists.
- Tweets dataset: we collected millions of vectorized tweets, as well as popular hashtags for each user.

The user profiles are collected through Twitter API and tweepy. The dataset is saved as json format, and each line is an user profile. An example of user profile is shown in Figure **??**

Our extensions are based on user profile information. The list of additional features are listed in Table 3. We also tried to measure the frequency of posting tweets, and ratio of retweets and original tweets, but due to time limitation and Twitter request limits, we did not finish them.

Figure 2: Example of a user profile

Table 3: Feature reference table

Feature Name	Original Format	Encoding Format
Geolocation	Raw String	Latitude, Longitude
Date of latest Tweet	Raw String Or None	Hour, Day, Month, Year
Language	Raw String	One hot encoding
Status count	Integer	Float
Favorite count	Integer	Float
Friend and Follower count	Integer	Float

However, many features in user profile are not well-formatted. For example, the date of latest tweet is "Wed Dec 07 16:00:32 +0000 2016", and the geolocation might be "NYC" or "Someplace behind you". To vectorize the features, we take three basic strategies:

- If it is a numerical data, we normalized it by maximum, such as follower count
- If it is a categorical data, we used one-hot encoding, such as language.
- If it is a arbitrary data, we sought help from third party, such as Google Map API.

Specifically, the processes of vectorizing two features, "Geolocation" and "Date of latest Tweet", are discussed in the following. To collect geolocation (we took a week to collect the data), we took such a strategy:

- We first checked the user profile's three columns: location; coordinate; geo. If none, we took the further step.
- We collected user's latest 20 tweets. We tried to find the latest geolocation in the column, location. If none, we took the further step.
- We used "Time Zone" as another geolocation indicator.
- If all none above, it will return the default geolocation value, [0, 0], for latitude and longitude respectively.

However, those collected information are all raw data string and most of them are not well-formatted and specific. To vectorize the information, we used four different Map APIs: Google Map API, Bing Map API, Mapzen API and Nominatim API (only limits to location in United States). However, both Google Map API and Bing Map API only have 2500 limitations per day, and Nominatim API only recognizes places in United States, thus, we requested Mapzen first, if it can not find the dataset, we sent request to Google[2] or Bing. We normalized latitude by dividing 90 and longitude by dividing 180 respectively.

[2]Google does not support Time Zone data well.

To vectorize "Date of latest Tweet", we took the following strategy:

- If the year is in 2016, we categorized it as "1", otherwise as "0". This feature might indicate that whether the user is active now.
- We used one-hot-encoding to vectorize both Month and Day, for 12 vectors and 31 vectors respectively.
- We split 24 hours into 4 parts:0-5; 6-11; 12-17; 18-23. And we used one-hot-encoding to vectorize the 4 parts. The 4 vectors might indicate user's timetable behaviors.

5 Experiment Results

Add geolocation and other features (language, etc.) to create the user embeddings.

Table 4 shows the additional views and its number of features we conducted experiments on.

Additional Views	**Num. of Features**
Language	42
Num. of users following	1
Num. of followers	1
Geolocation	2

Table 4: Additional Views used.

embedding dim.	Additional View	Precision@1000	Recall@1000	MRR
20	None	0.021045	0.619385	0.189527
200	None	0.022640	0.663512	0.259563
200	language	0.022305	0.651957	0.251101
200	following	0.018980	0.557607	0.218879
200	following (normalized)	0.022620	0.664022	0.262896
200	followers (normalized)	0.022670	0.663153	0.265102
200	followers + following	0.022665	**0.666763**	0.267265
200	geolocation	**0.022700**	0.665689	**0.269473**

Table 5: Evaluation results for hashtag prediction

Table 6 shows that the number of followers did not improve the baseline results; however, the number of users a given user follows ("following" feature) improved the baseline for all evaluation metrics. The "following" feature contributed on improving the baseline because users who follow a lot of users to tend to follow more users. On the other hand, users who tend to not follow other users, tend not to follow new users.

Note that since the default user features provided by the authors of [1] are normalized. Table 6 shows that the number of followers did not improve the baseline results; however, the number of users a given user follows ("following" feature) improved the baseline for all evaluation metrics. The "following" feature contributed on improving the baseline because users who follow a lot of users to tend to follow more users. On the other hand, users who tend to not follow other users, tend not to follow new users.

embedding dim.	Additional View	Precision@1000	Recall@1000	MRR
20	None	0.226172	0.072322	0.470136
200	None	0.229116	0.074218	0.450480
200	language	0.210572	0.068318	0.404000
200	following	0.175628	0.056333	0.381227
200	following (normalized)	**0.235188**	**0.076074**	**0.472213**
200	followers (normalized)	0.227000	0.073557	0.439606
200	followers + following	0.232188	0.075178	0.463236
200	geolocation	0.224488	0.072801	0.437948

Table 6: Evaluation results for the friend recommendation task

Normalization of features also helped improving evaluation metrics for both tasks. Note that since the default user features provided by the authors of [1] are normalized. Therefore, normalizing the "following" feature improved the all evaluation metrics significantly (5% on Precision@1000, 2% on Recall@1000, and around 10% on MRR).

6 Conclusion & Future Direction

In this project report, we conducted experiments on improving user embeddings. Specifically, we reported the results of adding more views to create Twitter user embeddings using GCCA. The experimental results on two tasks, friend recommendation and hashtag prediction, shows that additional views, such as the number of users following and the geolocation feature, lead to better user embeddings in terms of precision, recall, and MRR. In geolocation feature extraction, Mapzen sometimes returned strange geolocations, such as "Live behind you", thus, if we can get better feature representations, we might achieve better performance.

In the future work, there are three major directions: better interpretable model, improve optimization process, and explore better feature representations.

References

[1] Adrian Benton, Raman Arora, and Mark Dredze. Learning multiview embeddings of twitter users. In *Proc. of ACL*, pages 14–19, Berlin, Germany, August 2016.

[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.