Cross Language Tweets Recommendation Writeup

Doudou Zhou

Jan.10.2016

The idea is to map the English tweets and Spanish tweets that share the same hashtag into the same low dimension, with the assumption that tweets sharing the same hashtag are likely to be topically relevant/similar. We use an on-line learning algorithm with stochastic gradient descent (SGD) to learn the mapping between the two languages. SGD is used for incrementally updating and learning the corresponding parameter matrix for each language. For new data, the learned parameters are used to calculate the similarity scores between new tweet and existing tweets, and the top ranked tweets are popped up for recommendation.

In details, the procedures are as following:

1. Twitter Stream Tweets Collecting:

Use tweepy to sample live stream tweets and save the crawled JSON data in files. Each file contains 5000 tweets.

2. Tweets data preprocessing and cleaning:

The raw tweets texts are noisy, with some containing nothing but URLs.

Read twitter stream data and preprocess all the English/ Spanish tweets' texts as below:

all change to lower case -> remove 'RT' -> remove urls -> remove usernames -> remove hashtags/ or only remove '#' -> turn all whitespaces to single spaces -> remove punctuations -> remove stopwords (tried stemming, but detrimental on results)

The pruned English/ Spanish tweets texts are saved in the en_corpus list and es_corpus list respectively.

Then for all the pruned tweets texts, count the occurrences of unique tokens in each language. Remove the frequent and infrequent tokens from each tweet text. The frequent and infrequent thresholds are tested several times in order to reduce the total number of English tokens to about 20K, together with less than 20K Spanish tokens.

It is noteworthy that the number of English and Spanish tweets are significantly reduced after removing frequent and infrequent tokens. For instance, there are 1,316,682 preprocessed English tweets, but only 72,269 tweets (~5.5% of original size) are retained after removing frequent and infrequent tokens, in order to get a 21K size of tokens. While for Spanish tweets, only 62,786 tweets (~16% of original size) are left after removing frequent and infrequent tokens from 392,018 tweets, obtaining a 17K size of tokens.

3. Building the training and testing data for stochastic gradient descent algorithm:

The newly formed en_corpus and es_corpus lists after removing frequent and infrequent tokens are then transformed into two sparse matrices using TfidfVectorizer from sklearn.feature_extraction.text in python. The number of rows in the English tweets sparse matrix is the number of English tweets left so far, and the number of columns is the vocabulary (the number of unique tokens) built from all the processed English tweets saved so far. The values of non-zero elements of the sparse matrix are the TF-IDF values. So it is with the Spanish tweets sparse matrix.

While reading stream data, save all the hashtags of English tweets if any, together with en_tweetIDs in the hashtag_dict with structure {hashtag: {en: {id1, id2, …}}}. When a new Spanish tweet is saved, first check if its hashtag in the hashtag_dict. If yes, add {es: {id1}} in the hashtag_dict. If not, ignore this Spanish tweet and move on to next one.

In the end, if the length of hashtag_dict[hashtag] is two, then we are certain that we have saved both English and Spanish tweets IDs that share a common hashtag.

To build the data set for train and test, I established a dictionary with the structure

{(en_id, es+_id) : es-_id}. To build this dictionary, first I build a en_es+ dictionary: {en_id: {es_id1, es_id2, ...}} to store the relationship between English tweets and Spanish tweets that share a common hashtag.

The train and test data are store in csv files with each row as one data point.

Each data point/each row in a csv file is in the format:

['en_tweet_ID', 'en_tweet_text', 'en_tweet_hashtag', 'es_pos_tweet_ID', 'es_pos_tweet_text', 'es_pos_tweet_hashtag', 'es_neg_tweet_ID', 'es_neg_tweet_text', 'es_neg_tweet_hashtag',].

Since sparse matrix cannot be stored in csv files with the correct format, I save the sparse matrixes separately in .mtx files. Also I save the dictionaries for mapping English/Spanish tweetIDs to their sparse matrix's rowIDs in .pkl files.

In summary, three types of files (.csv, .mtx, .pkl) are created for saving all the train and test data information for further SGD implementation.

4. Implement the SGD algorithm:

After gaining all the train and test data, the SGD algorithm is implemented to learn the mapping matrice.

It is noteworthy that we use pair-wise loss function:

$$l(e_i, s_j^+, s_j^-) = -\log \sigma(h(e_i, s_j^+, s_j^-))$$, where

$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

$$h(e_i, s_j^+, s_j^-) = f(e_i, s_j^+) - f(e_i, s_j^-) = e_i^T W Q^T (s_j^+ - s_j^-)$$

$$f(e_i, s_j) = e_i^T W (s_j^T Q)^T = e_i^T W Q^T s_j$$, where

e_i	Tweet i in English	$V_e \times 1$
s_j	Tweet j in Spanish	$V_s \times 1$
W	Project matrix for English tweets	$V_e \times k$
Q	Project matrix for Spanish tweets	$V_s \times k$

I randomly choose a data point when updating the parameter matrices each time. And altogether do 2~3 or 3~5 passes on all the data points. Before each pass, shuffle all the data to avoid falling into cycles.

For every 500/1000 iterations, check if the total loss L is decreasing.

The stop condition is $|L_{t+1} - L_t| / |L_t| <$ threshold, such as 0.0001.

Further a regularization term is added to penalize model complexity and reduce overfitting, so the total loss function becomes $L + \lambda||W||_2$. Take derivative, then the former gradient becomes gradient + $2\lambda W$.