SI630 Project Report

Junqi Chen, Gengchen Yang, Yangqin Yan

Abstract

Giving a tweet a corresponding tag can help classify the tweets, evaluate the topics and archive the corpus. In this project, we proposed a BERT model that can generate a tag for a given tweet. First we collected data using Twitter API, then we applied BERT, RoBERTa and BerTweet to train our model, and finally we trained a Word2Vec ourselves to compute the relevance of our generated hashtags using cosine similarity (Sanh et al., 2019). We also implemented 2 baselines to compare with our proposed model. The result shows that our model is not novel enough and needs further improvement.

1 Introduction

With the increasing growth of information on social medias such as twitter, facebook, wechat, it is tempting that if a larger proportion of data can be classified. To deal with the problem that information at this amount is nearly impossible to be processed as a whole, twitter released "hashtag" function which allows users to summarize the post with one word, and use the word as a tag for the post. However, a large proportion of tweets do not contain any effective hashtags (Krokos et al., 2014). In this case, auto-generating hashtags for twitter has become an effective solution. This field caused much attention among scholars. Krokos et al. proposed using a variety of classifiers to generate hashtags for twitters (Krokos et al., 2014). Wang et al. treats hashtags as a sequence of words in order to summarize conversation context to generate hashtags (Wang et al., 2019b). We find that though there is progress in each popular existing method, there lacks a model that can summerize a tweet and tag it. Hence, we propose a model which summarizes tweets into keyword lists based on word embedding models. Then based on hashtags users generated, we can better manage the tweet contents and provide better search results for tweets. We can also improve the quality of tweet recommendations for users. We believe this model will demonstrate a good overall twitter hashtag generator for normal twitter users, and might give some insight to scholars.

For this project, our tasks will be divided into 2 parts, Tweet2Vec and Hashtag Generation.

1.1 Tweet2Vec

Tweet2Vec is a modification of Word2Vector model. We train a Word2Vector model which automatically generates dense vectors for each word in the collections of tweets. Then we define the average of all the words in a tweet as the vector representation of a tweet.

1.2 Hashtag Generation

This task defines a model which automatically generates hashtags for non-tagged tweets. This model analyzes the content of the text and relevant information of the tweet, and generates concise and easily distinguishable hashtags for the tweet.The model takes in a dense vector representing the tweet, and outputs any related hashtags for the tweet. The output of the model is a list of hashtags, where each hashtag is: either one keyword; or a sequence of keywords in certain order, with no spaces between them (ie. samplehashtag).

2 Data

In this section, we introduce some basics about our source data, including where the data come from, how we obtain the data and what kind of prepossessing steps we have done. Additionally, we may expand the source data to larger one for the better performance on future model training, which is included in the "Future Works and Modifications" part.

2.1 Data Source

We retrieve our source data from the Twitter with Twitter API, where we can access the recent tweets and obtain a concise and easy-to-use data format. We randomly retrieve the tweets from different users given a specific condition to form our source data. More details will be discuss later.

2.2 Data Collection

To retrieve the data, we create a project called `SI_630Project` on the Develop Portal of Develop Platform with an Essential access. SO that we can only access the Twitter API v2 (see Twitter API access levels and versions documentation), which is enough for our project.

According to our proposal, we plan to obtain the content, hashtags and user of the tweets for future training. We adopt the open source library Tweepy in python to obtain the source data.

Here are the detail steps to obtain the source data:

1 Randomly generate a series of common words to form our search query poll.

2 Randomly pick a query from the poll and search the tweets with method `Client.search_recent_tweets`. By specifying some useful arguments like `tweet_fields`, we can get the raw data from the tweets. Repeat this step multiple times until we have the enough data.

3 Perform data cleaning and pre-processing to the raw data and save the data to a csv file following a clean format (see below).

2.3 Data Pre-processing

We pre-process the data simple by performing following operations to the raw data:

- Keep the id for tweets and user as `id` and `author_id`
- Store the string format of tweet contents into `text` field with only url available for any non-text content like video.
- Filter the hashtags from the entities class in the `Response` and store it into a list (keep an empty list for tweets without any hashtags)

2.4 Statistics and Sample Data Entries

By performing these data pre-processing to the tweet data, we obtain the source data with following variables:

- `id`: ID for the tweet
- `author_id`: ID for the tweet's author (user)
- `text`: content of the tweet in string format
- `hashtags`: a list of hashtags in the tweet (empty list for non-tagged tweet)

The sample entries of our source data are should in Figure 1.

And Here are some basic statistics of our source data:

- Number of queries: 114
- Number of tweet data entries: 11397
- Number of hashtags: 2353

2.5 Future Works and Modifications

Plus, our source data can be easily re-create since it is a simple searching engine. Once executing, it will return required number of random selected recent tweets to form a new data set.

In future work, we may expand our source data for our training usage to obtain more flexibility. For example, we may include the relationship of a user, including his followers and who he is following. Additionally, we may consider design some targeted queries to get search result for a better training performance.

3 Related Work

3.1 Word Embedding

Word Embedding a method to transform each word to a vector with learned semantic and syntactic meanings (Wang et al., 2019a). It proves to be useful in finding the synonyms and understanding the meaning of words. Word Embedding based algorithms have been applied to find users' interest (Alekseev and Nikolenko, 2017). Word Embedding has advantages of ranking the texts and calculating the similarities. It is also easy to be evaluated using information retrieval metrics such as Normalized Discounted Cumulative Gain (NDCG). Based on the idea of Word2Vec,

	id	text	author_id	hashtags
0	1502482635268009984	@johnrich I see several each day. Sorry to bur...	1204036111946993664	[]
1	1502482634655649793	candle light bubble bath ¿ https://t.co/wpi1tq...	1475967654825496583	[]
2	1502482633866956800	RT @NUESTNEWS: NU'EST The Best Album 'Needle &...	784151839	[NUEST_JR_아론_백호_민현_렌, 뉴이스트]
3	1502482631002316801	@ellehcar_O2 nag bubble tea ka ba?	2875325584	[]
4	1502482624522043393	#Hyunlix 💙 ♡ saying goodnight on bubble at same...	1488856944186040325	[Hyunlix]
...	...	...	...	...

Figure 1. Sample Data Entries

other variations of word embeddings show satisfying performance in analyzing tweets. Soroush and Prashanth constructed character-level CNN-LSTM encoder-decoder model to generate tweet embeddings which can be applied to different languages (Vosoughi et al., 2016).

3.2 Hashtag Generation

Hashtag generation is helpful for managing the contents. For example, it can enhance the exposure of the content to search engines (Camargo et al., 2021). Researchers proposed a novel sequence generation framework via viewing the hashtag as a short sequence of words, which outperformed other models in MAP and F1 scores (Wang et al., 2019b). This reveals that sequences and contexts can be an important feature in the work of hashtag generation.

4 Methods

4.1 Subsampling

The pre-processed data mentioned in the Data section have lists of tags for each tweet. To simplify the task, we only preserve the first tag as our label. We plot the label distribution in this dataset, part of which is shown in Figure 2. We can see it roughly follows exponential distribution. It causes the risk of overfitting because some of the labels are far from enough and some labels occupy too much proportion. Our solution is to adjust the distribution of labels using subsampling method in homework 2. In this case, we made some modifications compared to the subsampling formula for Word2Vec and we sample the data according to the probability p:

$$p = ((\frac{z}{0.001})^{0.5} + 1) * (\frac{z}{0.001})^3$$

where

$$z = \frac{f_{label}}{N^{0.75}}$$

In the formula, f_{label} denotes the number of times the label occurs in the dataset and N denotes number of labels the dataset has. Using this formula, we can reduce the probability of the labels that frequently exist and elevate the probability of the labels that rarely exist. Also, we dropped data that have labels existing below twice.

After that, we split 80% of the data as training set and 20% data as validation set. At the beginning, we intended to use 10% of the data as valiation set, however, due to the limitation of cuda memory, we can not use that amount of data for validation. We only use 1000 data for reference of validation. Our validation metric is macro-f1 score.

4.2 Model

Our assumption is that the each tweet should belong to the tag that has existed in the corpus, since most of the tweets are posted to discuss an on-going topic or a hot topic which has existed. Therefore, we propose a BERT classification model to do the tweet tagging. This model is supposed to classify any tweet to a tag in our vocabulary in the corpus. Since the tweet has many emojis and abbreviations that are much different from normal texts. We tried to apply vinai/bertweet-base, which is a pretrained model on a large tweet corpus. The advantage is that it can understand the meaning of abbreviations and emojis (Nguyen et al., 2020). We also used vinai/bertweet-base as our tokenizer.

4.3 Hashtag Prediction

In this section, we aim to automatically add hashtags to each tweet using a classifier. To generate the training data, we split the tweet contents and the hashtags in them. Then we use the embeddings we learned from the last section to transform tweets and hashtags into vectors so that we can feed them into a model. The labels are the vectors of hashtags that each tweet belongs to.

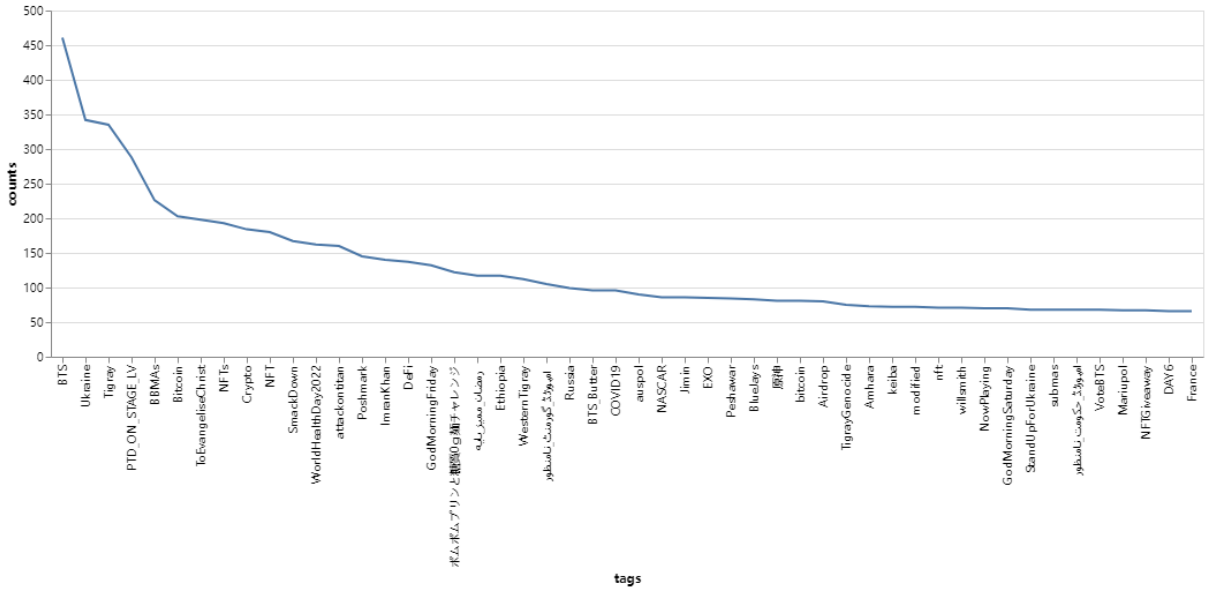

Figure 2. Distribution of Tweet Tags

Param Name	Value
Min_token_freq	8
Neg_samples_per_word	2
window_size	2

5 Evaluation and Results

We trained a Word2Vec model for evaluation. We trained our Word2Vec model with data from wiki-bios.med.txt in Assignment 2 and our collected data. The hyperparameters are set as follows:

We will be evaluating the accuracy of our model in predicting hashtags for the tweets using cosine similarity, or described in the following formula:

$Accuracy = mean(cosineSimilarity(p_i, a_i))$

where p_i is the ith predicted hashtag's word-to-vector, a_i is the ith actual hashtag's word-to-vector, and cosineSimilarity() is a function which returns the cosine similarity of two vectors.

For baseline measurements, we developed two levels of baselines: random generation and using the most popular hashtag.

The former method predicts the hashtag to be one random hashtag found in hashtags we see in the data set. And the latter predicts the hashtag to be the most common hashtag in the data set.

From the data set we found the most common hashtag to be 'BTS'.

The results for the accuracy of the two baseline evaluation methods are shown as follows:

$Accuracy_{random} = 0.45\%$

$Accuracy_{most\_common} = 3.5\%$

The result of our prediction is:

$Accuracy_{prediction} = 0.2\%$

6 Discussion

6.1 Model

Before we finally chose the model, we did an extensive search on hugging face about existing models that can handle the task of tweet tagging. We found that hugging face models can only do text-classification, text-summerisation, question-answering, language modeling and speech recognition. We picked 2 tasks that are most relevant to our task to have a try. The first is language modeling. To fit this task, we transform our data to the form that input and output has the same length and almost same text, except that the input has the masked tag. However, when we use the roberta model to do the language modeling, we found that the model only predicts '#'. This reveals that the tokenizer and the model only learns the rough function of tweet tags instead of learning the actual meaning of tags. Therefore, it only outputs '#' to denote that this position should be a tag, but it never determines what kind of tag should be placed there. This means that the language modeling task on hugging face can not finish our specified task, or we should come up with a customized tokenizer that can fit into the roberta model. Due to the time limit, this is impossible to complete in this course.

Then we move to the task of text-classification.

We interpret our problem into the task of classifying each tweet to a given tag. This makes the problem feasible to handle using hugging face. Then we tried both prediction and classification using BertForClassification on hugging face. Using our evaluation, the results are similar and the performances are less than our baseline. This may be attributed to several reasons. First, the task itself is challenging for existing bert models. The corpus from twitter is different from normal text and some tags even don't have enough context to train the embeddings. The exponential distribution of labels we mentioned before also add the challenges of training the model without bias. Second,

6.2 Evaluation

Since we are generating hashtags, which is a brief summary of the tweet, there is a great chance that the generated hashtag, though is not accurately the same as the actual hashtag, still has similar meaning to the actual one. For instance, if the original hashtag is "PCTG", then generating a hashtag "PokemonCards" might also be appropriate. Therefore, instead of using F1 score as accuracy measurement, the mean of cosine similarity between hashtag vectors are chosen as accuracy. The benefit of this method is that it will add some "smoothing" to hashtags that had not appeared in the training data set.

However, this approach also has some disadvantages. First and most important of all, we lack data, or ground truth word pairs to evaluate our Word2Vec model. Currently we can only evaluate the model on how the model performs in finding similar words. For example, the top most frequent words for $January$ should be $February$, $March$ etc. and we should be able to obtain $King - Man + Woman = Queen$. Despite the qualitative and subjective measurements, we do not have a quantitative and objective measuring method.

7 Conclusion

In this project, we proposed a BERT model that can generate a tag for a given tweet. We first collected data using Twitter API, then we applied BERT, RoBERTa and BerTweet to train our model, and finally trained a Word2Vec ourselves to compute the relevance of our generated hashtags. Our codes are uploaded to github[1].here. Unfortunately, our work can not beat the baseline and cannot complete the task successfully. Future research and improvement is required to make the model better. In the future, we need to select better datasets and further balance the distribution of labels to avoid the poor performance of our model. We can also try other models other than BERT so that it can be easier for us to customize the tokenizer and learn the meaning of tags.

8 Other Things We Tried

In this project, we have tried different methods to explore possible model for a better performance. There are some setup we tried:

- **Label Choices**: In this project, we have a hard to figure out how to define the label for our dataset. What have tried many kinds of label, including but not limited to:
  - The list of all hashtags
  - The string of first hashtag
  - Whole hashtags uncovered text
  - The integer of tokenized hashtag in customized vocabulary

  However, we still find it hard to have an acceptable performance since there are tons of labels in our task. So we finally adopt the last one as our label for training.

- **Prediction Model**: We first adopted a prediction model instead of the classification model. Mapping from each hashtag to a integer as the label, we tried to predict the result of hashtags for a test tweet. But the problems come:
  - We can hardly figure out how to deal the float outcomes: which hashtag should a float prediction belongs to?
  - The result shows that most of the predictions are the same words with a occurrence frequency. We did not regard it as an acceptable result.

  As a result, we turn to the classification model later.

- **Masked Model**: Besides, we also tried a masked model like what we did in the Assignment 4. We masked up the <UNKTAG>

[1] http://acl-org.github.io/ACLPUB/formatting.html

and tried to predict the hashtag at the position, comparing which with the true hashtag.

- **Different Pre-trained Model Choice**: We introduced different kinds of pre-trained model in our project, including but not limited to:
  - BERTweet (Nguyen et al., 2020): a pre-trained language model for English tweets
  - RoBERTa (Liu et al., 2019): Robustly Optimized BERT Pretraining Approach

  However, we finally adopt the bert-uncased model in the end due to the distinguishing performance it has on our dataset.

9 Project Reflection and Future Investigation

In our Project Proposal, we planned to implement a Tweet Recommendation Engine after the hashtag prediction task. However, we did not explore further on this topic due to the limit of time and unexpected result from the prediction task. In this part, some enlightening ideas about what we would have done differently or next in this project will be discussed:

9.1 What Would Have Done Differently

In this project, we have a hard to figure out which model should we use to achieve the purpose of predicting hashtags. We have tried different kinds of model but most of them did not have a good performance. We should re-consider some different models like a text summarizing, or classification task with annotated labels. To be more specifics,

- **Text Summarizing**: Given a tweet text, summarize the meaning at extract the keywords as hashtags prediction.
- **Classification with Annotated Labels**: To reduce the number of class we have due to tons of labels, we have to first perform an annotation to shrink the class number. For example, classify some tweets to class "Daily" and some to “Politics”. With customized classes, we can greatly reduce the training time and have a better prediction result.

9.2 What Would Have Done Next

Given the predicted hashtags, we can explore further on the recommendation task or user image construction. Here are some initial ideas for future investigation:

- **User Images**: Based on the hashtags and predicted ones in the tweet, we can construct a user image record to depict his digital record. For example, we can have
  - Background information for the user if permitted
  - Topic that the user follows recently
  - Relationship between users

  All these information included in the user image will contribute to future task like tweet or ads recommendation.
- **Tweet Recommendation**: Based on the user image we have, we can analyze the We the most concerned topics and recommend relevant content to him. Such a smart recommendation is not only useful for improve the user experience on browsing the Twitter, but also increases the potential profits.

10 Group Efforts

The following table are the general part of contributions for each group member:

Group Member	Tasks
Junqi Chen	Data Collection and Pre-processing
Yangqin Yan	Model Selection and Training
Gengchen Yang	Evaluation and Result

Table 1: Contribution of Each Group Member

And these are detailed job descriptions:

- **Junqi Chen**:
  - Collect the source dataset from Twitter
  - Cover up the hashtags in tweets and construct the training data
- **Yangqin Yan**:
  - Model Selection and parameter tuning
  - Train the model and predict the hashtag
- **Gengchen Yang**:
  - Define the evaluation metrics
  - Evaluate the prediction results

References

Anton Alekseev and Sergey Nikolenko. 2017. Word embeddings for user profiling in online social networks. *Computación y Sistemas*, 21(2):203–226.

Augusto Camargo, Wesley Carvalho, Felipe Peressim, Alan Barzilay, and Marcelo Finger. 2021. Text-to-hashtag generation using seq2seq learning. *arXiv preprint arXiv:2102.00904*.

Eric Krokos, Hanan Samet, and Jagan Sankaranarayanan. 2014. A look into twitter hashtag discovery and generation. In *Proceedings of the 7th ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, LBSN '14, page 49–56, New York, NY, USA. Association for Computing Machinery.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. Bertweet: A pre-trained language model for english tweets.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Soroush Vosoughi, Prashanth Vijayaraghavan, and Deb Roy. 2016. Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, page 1041–1044, New York, NY, USA. Association for Computing Machinery.

Bin Wang, Angela Wang, Fenxiao Chen, Yuncheng Wang, and C-C Jay Kuo. 2019a. Evaluating word embedding models: Methods and experimental results. *APSIPA transactions on signal and information processing*, 8.

Yue Wang, Jing Li, Irwin King, Michael R. Lyu, and Shuming Shi. 2019b. Microblog hashtag generation via encoding conversation contexts.

A Appendix

The source code of our evaluation and our implementation of baseline can be found in this link.

The data we scraped from twitter can be found on this link.