Tweet Popularity Detection

VISHAL SINGH YADAV (CS20MTECH01001) and DESU SURYA SAI TEJA (CS17BTECH11048)

Cascade outbreak is a common phenomenon observed across different social networking platforms. Cascade outbreak might have severe implications in different scenarios like a fake news/rumour can spread across a significant number of people, or a hate news can be propagated, which may incite violence etc. Early prediction of cascade outbreak would help in taking proper remedial action and hence is an important research direction.

1 INTRODUCTION

Popularity prediction is an important task for finding out the extent of exposure a particular social media post will get, i.e. how popular a particular tweet/post will get. This is fairly important in current world of information age where information can spread rapidly. This has lead to an important shift in how we, the public consumes information.

This is further intensified by the fact that we are witnessing a lot of fake information being on the rise and being spread through these platforms. This has led to a lot of social issues in recent times. Popularity prediction can help by finding posts/tweets that a have a high probability of getting popular and hence can be subjected to moderation before damage is done.

2 PROBLEM SETUP

The main objective of this project is to develop a method to predict the future popularity of a given online post.

We would like to describe a few terminologies which will be used in this report.
Observation Time: This time indicates the duration for which we observe the tweet before making a prediction for future number of retweets during training. We have divided the observation time into time slots of 600s each. In this study we are trying to find the number of retweets at Time_d at the initial Time_o.

We have m tweets

$$(tweet_1, tweet_2, tweet_3, ..., tweet_m)$$

and cascades corresponding to each tweet are

$$(cascade_1, cascade_2, ..., cascade_m)$$

.

Threshold: We define threshold as number of retweets necessary for a tweet to be popular. If by time_d a tweet is predicted to be of class 0, if the number of retweets by Time_d duration < *threshold*. If number of retweets by Time_d duration > *threshold* it is predicted to be of class 1 (tweet is popular).

Authors' address: Vishal Singh Yadav (CS20MTECH01001); Desu Surya Sai Teja (CS17BTECH11048).

3 DATA COLLECTION AND STATISTICS

We have used existing SEISMIC [**Seismic**] database. Many popularity prediction works have used this publicly available database. The database contains over 3.2 billion tweets and retweets. Out of these, we filtered out non-English tweets and tweets with less than 50 total retweets. This resulted in a data-set containing 1.6K+ tweets. We divide these tweets into two parts keeping half for training and half for testing. This data-set does not contain tweet information which we downloaded from Twitter [1] using twitter data downloader [2].

4 METHODOLOGY

Our model is created using 3 parts as described here:

1. Textual feature extractor: In this sub-module, tweets are first pre-processed by eliminating stop words, punctuation, URLs, hashtags, mentions, reserved words (RT, FAV). After pre-processing, individual words of each tweet are converted to embedding vectors by utilizing keras embedding layer. Hence, we have a sequence of vectors for a sequence of words corresponding to each tweet. This sequence of embedding vectors is then fed through LSTM units to extract latent textual features. We can say that LSTM sub-module gives

$$F_{LSTM} = LSTM(T)$$

Here the function LSTM(.) takes embedding vector and previous hidden state as input and produces latent feature F$_{LSTM}$.

2. Retweet count feature extractor:we can represent a cascade as a sequence of time windows after monitoring each tweet up to it's maximum observation time. Each of these time windows has zero or more number of retweet counts. Then we use one-hot encoder to represent the total number of retweets in each time window to a vector. Thus, we have a sequence of vectors for a sequence of time windows corresponding to each tweet. Each of these embedding vectors is then fed through single GRU units to extract latent features. We can summarize this sub-module as

$$F_{GRU} = GRU_{attention(R)}$$

3. Final module: In this sub-module, we first concatenate two latent feature vectors inferred from the first two sub-modules.

$$F_{concatenate1} = F_{LSTM}F_{GRU}$$

where · represents vector concatenation. Output of the first level concatenation Fconcatenate1 is then passed to two fully connected Layers. Output of the second fully connected layer is denoted as Fdense . Fdense is again concatenated with Fexplicit.

$$F_{concatenate2} = F_{dense}F_{explicit}$$

The output of this second level concatenation Fconcatenate2 is forwarded to a fully connected Layer followed by a softmax activation function. The softmax function returns the probabilities of each class and the class with the higher probability is finally tagged.

5 EXPERIMENTS

The hyperparameters are fine-tuned with the train dataset. As the class size is only 2, there is higher probability of model overfitting the data. To prevent this, l2 activity regularizer is added to the layers, combination of linear and non-linear activation functions(relu, sigmoid) are considered and softmax for the final dense layer. The dropout for lstm and gru are considered to 0.4 and 0.1 respectively. BinaryCrossEntropy loss, adam optimizer(with learning rate=0.01) are used for training.

For the explicit features, text length, url_count are considerd for tweet information features and number of followers of the tweet person is considered for social information features. These features for each of the dataset is collected and min-max normalization is performed. (Due to less resources , these are first calculated and stored as csv(explicit_features.csv) instead of calculating on the fly. However, we provide necessary functions to do it in the code.) For experiments, time window is considered to be 10 min and the obeservation time to be 1 hour, similar to the RNN-HMFC.

Threshold	Precision	Recall	F1-score
500	0.96359	0.9645	0.964
600	0.9726	0.97339	0.97296
700	**0.9759**	**0.9769**	**0.9763**
800	0.97203	0.9712	0.9588

6 CONCLUSION

This model captures latent features of tweet text and retweet frequency using deep learning models. Also RNN - HMFC incorporates a set of handcrafted features like user social information and tweet information which play an important role in spreading. Compared to Seie

7 REFERENCES

[1] Twitter. *Twitter.* May 2001. URL: www.twitter.com.

[2] Quan Nguyen Van. *Download Tweet using ID.* 2018. URL: https://github.com/quanap5/DownloadTweetsUsingID?files=1.