Tweet Popularity Prediction

JINDE SAGAR, AI19MTECH11006
VINAY PRAKASH, AI19MTECH11003
KUNAL KADIAN, AI19MTECH11001

We visit the tweet popularity prediction problem by considering: tweet language semantics, author's social relationships, and the diffusion process of tweets. To model the content of tweets, we use a neural network that combines textual, and social cues together. This is used for pre-conditioning a 'dynamics RNN', which models the tweets propagation process. A Poisson regression loss is optimized to train the network. This work is a variation of "Retweet Wars: Tweet Popularity Prediction via Dynamic Multimodal Regression" Paper

Additional Key Words and Phrases: NLP, retweet count prediction, tweet popularity

ACM Reference Format:
JINDE SAGAR, VINAY PRAKASH, and KUNAL KADIAN. 2020. Tweet Popularity Prediction. 1, 1 (September 2020), 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The world is better connected than ever before. On social networks, users are connected to large number of other users. Short communication distance and ease of access make online social media an increasingly popular venue for information sharing. Both individuals and organizations can be easily overwhelmed by the sheer volume of online posts or misled by wide-spread rumors. Therefore, the ability to predict which post has a high popularity potential in its early stage can help individuals improve their communication efficiency and also allow organizations sufficient time for remedial actions. Reliable forecasting of online content popularity is thus a vital need. Here, we study the role of content for the popularity prediction task in a setting where the early retweeting process is also known. Subsequently combining content features with social cues and, we show that, in addition to who you are, what you say is also important indicator of the breadth of message's reach.

We present the prediction problem on the Twitter domain. We adopt the LSTM-RNNs to model the textual features. We use in-domain word embeddings specifically trained on tweet-style language as input to the LSTM-RNNs. Deeply learned features together with user-specific social features, are then used to learn a Poisson regression model which predicts the potential influence of the given tweet. We propose to use recur- rent neural networks to predict the final retweet count for a given sequence of retweeting actions. The content based features are used as pre-conditioning for the recurrent network. To evaluate our proposed model, We use our own collected dataset, which is mentioned in detail in Dataset section.

Authors' addresses: JINDE SAGAR, AI19MTECH11006; VINAY PRAKASH, AI19MTECH11003; KUNAL KADIAN, AI19MTECH11001.

Table 1. Tweet text pre-processing rules

Category	Before	After
Hashtag	covid	<hashtag> covid
Numbers	3.1415926	<number>
Username	@narendra	<username>
Long words	greeeeat	great <elong>
Retweet Tag	RT:	Removed
Capitalization	SAD	<allcaps> sad

2 PROBLEM SETUP

For a tweet, we predict how much retweets it will get after some time (say 3 days). We predict this from textual contents, user information, and initial response to this tweet (number of retweets in first hour, second hour, third hour, . . . , 24 hours). We consider initial response only in training data while test data considers only textual contents, and user information. We have restricted the tweets to Covid-19 topic. We use: 1. A neural network model that harnesses all available Twitter data modalities: textual and social cues; 2. A Poisson regression model for predicting retweet count based on all available data modalities

3 DATA COLLECTION AND STATISTICS

We have used our own dataset. We collected around 12000 tweets along with their user information and their retweet counts for first 24 hours. All tweets are on Covid topic. We used 100 keywords to collect tweets. Some of them are mentioned below

Coronavirus, Koronavirus, Corona, CDC, Wuhancoronavirus, Wuhanlockdown, Ncov, Wuhan, N95, Epidemic, outbreak, covid-19, corona virus, covid, covid19, sars-cov-2, COVID19, COVD, pandemic, coronapocalypse, Coronials, Social Distancing, panic buy, 14DayQuarantine, DuringMy14DayQuarantine,panic shop, InMyQuarantineSurvivalKit, panic-buy, coronakindness, quarantinelife, stayhomechallenge, DontBeASpreader, lockdown, staysafestayhome, flattenthecurve, PPEshortage, saferathome, GetMePPE

We collect following social features i.e user information from the author's profile U : account age, friend count, follower count, total tweet count, favorited tweet count.

A tweet having language description L is first issued by its author U . At time ti the tweet of interest accumulates ri retweets. Such dismantling process is recorded as D = (t0 , r0), (t1 , r1), . . . , (tN , rN). Note that D only records the early stage of the dismantling. The maximum retweet count during the data collection period is used as the ground-truth retweet count rgt.

Messages on Twitter usually contain informal language. Accordingly, we preprocess the text to reduce irregularities to lessen the burden on the later LSTM network. We first reduce the irrelevant information in tweet text by simplifying hashtags, numbers, usernames, etc. Please refer to Table 1 for detailed pre-processing rules. Instead of using a single symbolic word <URL>, we expanded and parsed the hashed/shortened URL within tweets. Only domain names are recorded as words.

4 METHODOLOGY

We consider the problem of predicting tweet popularity, that is the number of times a tweet will be retweeted after a certain period of time. A tweet T containing language description L is tweeted by its author U. At time t_i a certain tweet has r_i retweets. Such process is recorded as $D = \{(t_0, r_0), (t_1, r_1), \ldots, (t_N, r_N)\}$. The retweet count after each hour (for N hours]) during the data collection period is used as the ground-truth count (r_0, r_1, ..., r_N). For simplicity, we will use r_{gt} to represent the ground-truth for any particular hour. As a discrete probability model, the Poisson distribution characterizes the probability of a given number of events occurring during some time period. Therefore, the retweet count r of a tweet $T(L, U, D)$ follows a Poisson distribution:

$$P(R = r|\lambda) = \frac{e^{-\lambda}\lambda^r}{r!} \tag{1}$$

where the latent variable $\lambda \in \mathbb{R}^+$ defines the mean the variance of the underlying Poisson distribution. While testing our model, we use only the tweet's language description L and user-profile U for the retweet prediction and will not use the propagation information D.

We propose a network that combines the multi-modal information from the unseen tweet $\tilde{T}$ to predict the Poisson parameter $\tilde{\lambda}$ for its latent Poisson distribution $P(R)$. The retweet count prediction $\tilde{r}$ for $\tilde{t}$ can then be easily inferred by maximizing $P(R; \tilde{\lambda})$:

$$\tilde{r} = max\left(\left\lceil\tilde{\lambda}\right\rceil - 1, \left\lfloor\tilde{\lambda}\right\rfloor\right) \tag{2}$$

4.1 Feature Network

Figure 1 gives an abstract view of our overall network architecture. Our proposed model consists of two stages: a feature extraction network and a dynamics RNN network. The feature extraction network processes language L and user-profile U. The output features are used to pre-condition the dynamics RNN. The precondition dynamics RNN then processes the propagation process data D to estimate the hidden Poisson parameter λ.

In feature network, a long short-term memory recurrent neural network together with tweet-trained word embedding encodes the variable length tweet language L into a fixed dimensional feature vector $f_{LSTM}(L)$. The character limitation and the lightweight retweet operation noticeably differentiate the Twitter language to be very different from daily languages. Users tend to use abbreviations, Internet slang, emojis, and hashtags to emphasize their emotions. Thus, we need a powerful language model to characterize and understand the semantics of the tweeted text. We use LSTM-RNN to model variable length sequences such as tweet text. Individual words are first mapped to an embedding layer. The sequence of embedding vectors are then fed through LSTM to extract textual features. We used 100-dimensional word embedding from http://nlp.stanford.edu/data/glove.twitter.27B.zip. These word embeddings are created using 2 billion tweets. We also trained a GloVe model on the tweets that we collected ourselves to learn the word embedding, specifically for the words which are related to "corona" pandemic. We take the final output from the LSTM-RNN as the textual feature $f_{LSTM}(L)$ for tweet T.

4.2 Dynamics RNN

We use a simple but effective recurrent neural network to learn the temporal propagation pattern. Given a tweet $T(L, U, D)$, at each time step i, the dynamics RNN updates its hidden state h_i and computes an output prediction $\tilde{\lambda}_i$ by

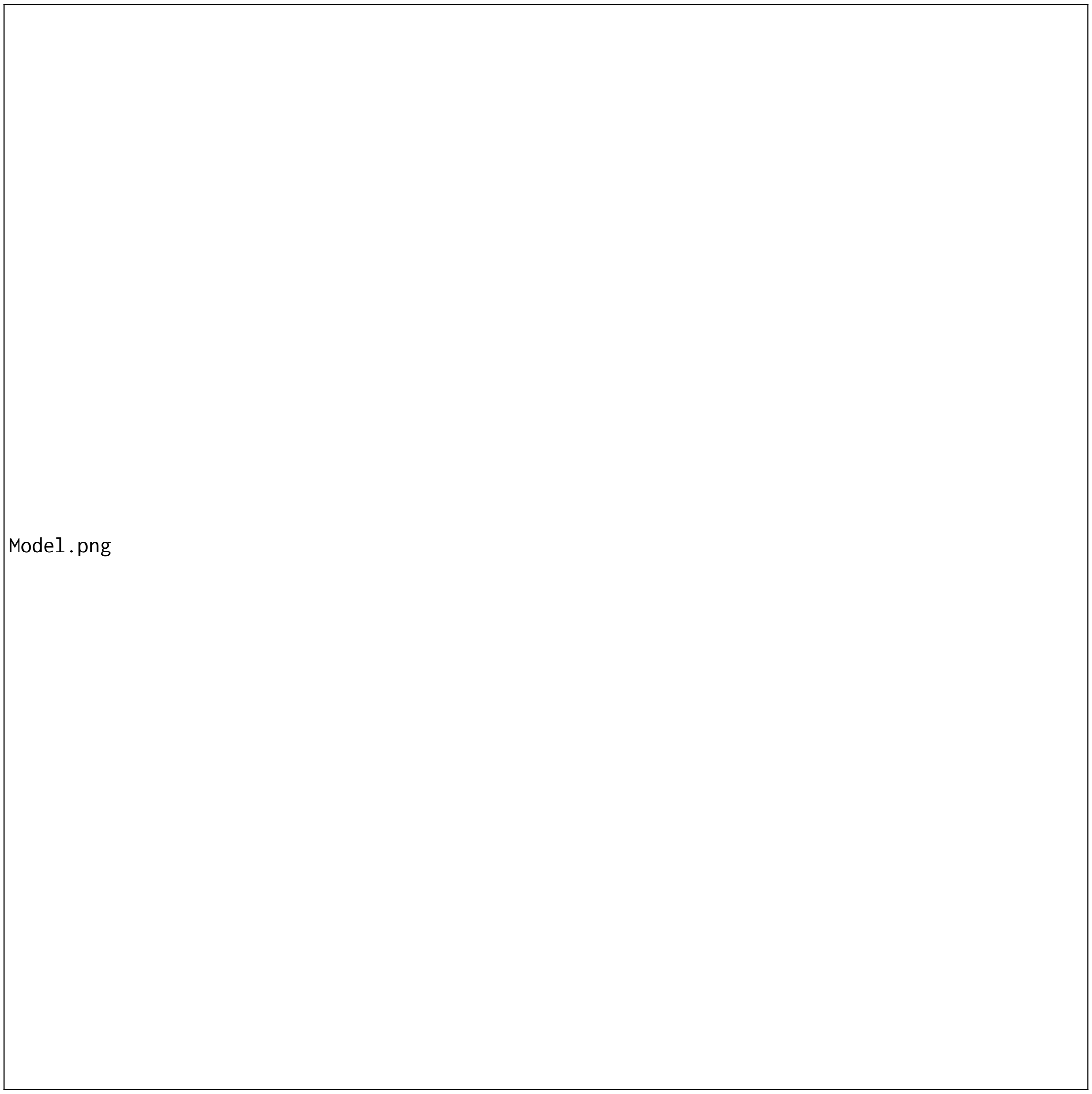

Fig. 1. Our proposed model to predict tweet popularity. LSTM is used to extract textual features from tweets.

iterating the following relations:

$$\begin{aligned} c &= W_{hc}[F_c(L), F_s(U)] \\ h_i &= tanh(W_{hr} r_{i-1} + W_{hh} h_{i-1} + b_h + c \odot \mathbb{I}[i = 0] \\ \ln(\lambda_i) &= W_{oh} h_i + b_o \end{aligned} \tag{3}$$

Weights W_{hc},W_{hr},W_{hh}, W_{oh} and biases b_h, b_o are learnable parameters. $\mathbb{I}$ is an indicator function.

Table 2. Detailed configurations of our proposed network

Component	Layer	Dimensions/Units
LSTM	Hidden	512
Social Features	-	5
FC Layer	FC Layer	128
Dynamics RNN	Hidden	128

4.3 Social Features

Tweets are spread over Twitter by the retweeting them by the users of the twitter. The content quality of a tweet and the characteristics of its author can significantly affect its potential reach. Famous and active users can spread the information faster and broader on the network than less well-connected users. Thus, it's natural to consider the author's characteristics when prediciting the popularity of a new tweet. We can directly extract social features from the author's profile U: $account\_age$, $friend\_count$, $follower\_count$, $total\_tweet\_count$, $favourited\_tweet\_count$.

Together with the cross-product transformation features $\phi(U) = \{u_i \cdot u_j | u_i \in U, u_j \in U, i < j\}$, we have the following social feature:

$$F_s(U) = [U; \phi(U)] \tag{4}$$

4.4 Poisson Regression

We train our model to maximize the Poisson likelihood (at each time step) given a collection of N training tuples of tweets T_i and their retweet counts $r_{gt,i}$:

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \{\lambda(T_i) - r_{gt,i} \ln \lambda(T_i)\} \tag{5}$$

where θ contains all parameters of our proposed model. The loss function can be denoted as:

$$L_{Poisson} = \frac{1}{N} \sum_{i=1}^{N} \{\lambda(T_i) - r_{gt,i} \ln \lambda(T_i)\} \tag{6}$$

5 EXPERIMENTS

5.1 Network Architecture and Training

Network configuration We detail the architecture and configuration of our proposed model in Table 3. Our network contains multiple components. We train each individual component separately. Thereafter, we train the entire model jointly.

Warm-up language model We start by training the twitter language model, i.e, the LSTM network. We are not training the embedding layer because we are using pre-trained word embedding which are trained on 2 billion tweets. Further, we trained the word embedding on the 12K tweets that we collected to get the word embedding for the words related to "corona" pandemic. We used GloVe for this. A generalized linear model is used to predict the Poisson parameter λ from the LSTM output vector.

$$\ln(\lambda(T)) = \mathbf{w} \cdot f_{LSTM}(L) + b \tag{7}$$

where linear weight **w** and bias b are learnable parameters. We then train the LSTM network based on Equation 7 with the loss function in Equation 6. We train the LSTM with Poisson loss for 2000 iterations.

Warm-up dynamics RNN We randomly initialize the dynamics RNN. The dynamics RNN is designed to predict λ from past observations. Thus we train the dynamics RNN using the Poisson loss Equation 6. We fix the language LSTM component of the network during the warm-up phase of the dynamics RNN and train it for 2000 iterations.

Preventing overfitting Since the data can be very noisy, we adopted different techniques to prevent over-fitting. We used dropout layers in the LSTM network with dropout probability of 0.3. Moreover, we used L_2 regularization during training.

Optimization We initialize and warm up each component of our network separately as discussed above. Then, we combine the negative log-likelihood $L_{Poisson}$ the weight θ regularization as the joint loss function:

$$L = L_{Poisson} + k_2||\theta||_2 \tag{8}$$

Our model is implemented in Keras and is optimized using Adam. L_2 regularization is achieved by initializing the regularization term at each layer. The value of k_2 used in the above equation is 0.05 and was found out by cross-validation.

We perform different modifications to the model while training to get better results. We tried adding batch normalization after the dense layer. However, it was not helpful so we didn't use it in our final model. We also tried replacing ReLU with Leaky-ReLU. However, there were not significant improvements. We also tried initializing the weights. This resulted in different loss values at the start but eventually it settled for the same losses as we were getting with random initialization. We also tried using different batch sizes. We finally used 512 as batch size.

6 CONCLUSION

We were not able to product desirable results using complete dataset. But while testing on a smaller dataset which were rich in non-zero retweet-count, we were able to product results which were better then the one using complete dataset. Lack of desired results might be due to many all zero rows in the training dataset.

Results produced by partial dataset on 500 iterations is shown in Table 2.

Table 3. Ground Truth VS Predicted with Partial Data

Component	Layer	Dimensions/Units
Hours	Ground Truth	Predicted
1	14	1033
2	16	792
3	26	995
4	39	1748
5	45	1739
6	45	2420
7	48	2947
8	49	3813
9	49	4633
10	49	5370
11	49	5956
12	49	6326
13	49	6695
14	50	6793
15	50	6904
16	50	6960
17	50	6989
18	50	7015
19	50	7018
20	50	7030
21	50	7034
22	50	7037
23	50	7037
24	50	7039
25	50	7039
26	50	7039
27	50	7039
28	50	7040
29	50	7039