Twitter Geotag Prediction

Bowen Zhang, Ziqi Li, Jiangtao Chen, Tianyu Zhang, Yuhong Jiang, and Jiapeng Wan

Dept. of Computer Science, University of Los Angeles, California

December 2020

Abstract

Under the background of the global epidemic of covid-19, the study of human flow become increasingly important. In the era of Internet, real-time tweets can provide useful information on evolving pandemic situations. Since we need to know where each patient had been to, Geo-tagged tweets are especially useful in helping with investigation and control of the spread of COVID-19. In this report, we adapt a state-of-the-art neural network model for city-level geolocation prediction. We further improved it by removing unused features and rebuilding the text feature processing part using different pretrained word2vec model. After being trained using nearly ten thousand geo-tagged tweets, our model achieved a training accuracy of 20.3%, outperforming the old model by an increase of 2% in training accuracy and an increase of 0.6% in validation accuracy.

1 Introduction

In the past months, a highly contagious respiratory and vascular disease, COVID-19, has spread throughout the world and became a pandemic. The COVID-19 pandemic has led to a dramatic loss of human life worldwide and presents an unprecedented challenge to public health, food systems and the world of work. To control this deadly disease, whenever a new confirmed COVID-19 case appears, a comprehensive Epidemiological Investigation needs to be cast. CDC Investigators need to know where the patient had been to and who were the potential close contacts. On the other hand, with the development of world-wide-web technology, internet became one of the most important information carriers in the world. People started sharing their lives and thoughts through platforms like facebook or twitter. Twitter, as one

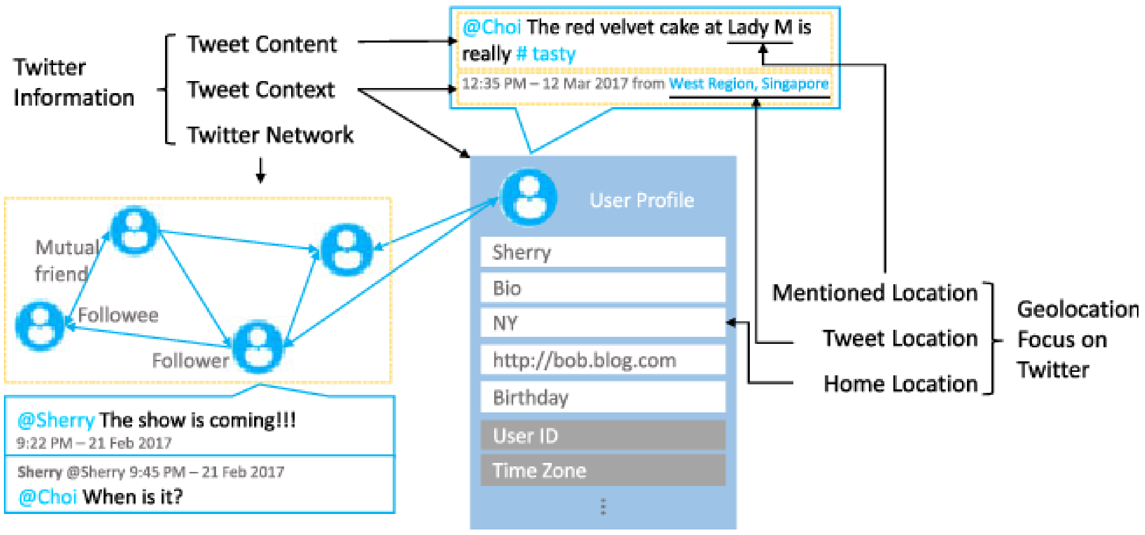


Figure 1: Twitter User information

of the largest social media in the world, could potentially help. Each day, more than hundreds of thousands of tweets are posted, and many of them may contain a geo tag or location-indicative words. If we could make use of these position information and build up a model to predict the geological information of a user, the Epidemiological Investigation process will be boosted, and thus will help to gain control over the pandemic. However, tweets are not geotagged by default, requiring Twitter users to manually activate geotagging. As a result, only 0.9% of all tweets are geotagged, considerably limiting their use for situational awareness.

To increase the amount of geotagged tweets, researchers have worked out many different models for predicting the location of a tweet, for example the deep learning classifiers [7] and gazetteer-based methods [10]. However, recently the Twitter's public feed has been changed. In that case, these changes may potentially affect these models. In this paper, we will report (i) adapt an existing geolocation prediction method based on the changes made by twitter since 2020. (ii) computationally improve its accuracy by utitlizing word2vec model in the text layer. we will use multiple pre-trained word2vec model. (iii) evaluate the effectiveness of the original version and modified version.

2 Problem Statement

Though twitter provides geotag for users to provide their locations, only around 1% of tweets are actually tagged with geo location. Since most of the tweets' location can't be retrieved directly. In 2012, people started to investigate on how to make predictions base on other features of twitter. As shown in figure 1, a tweet is able

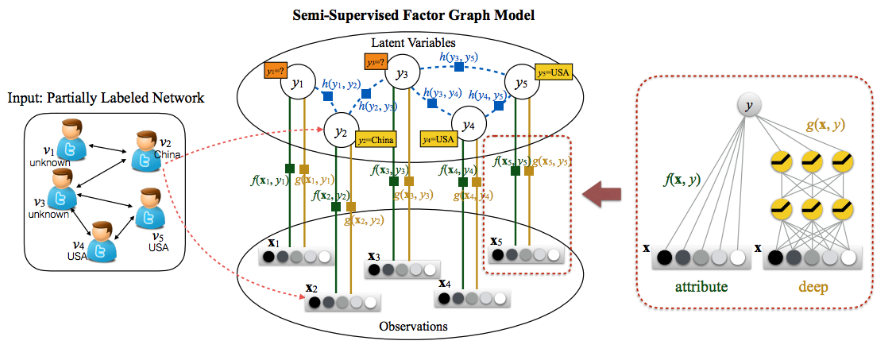


Figure 2: User level graphical model

to carry many different kinds of information. People started utilizing these features to predict the location. Researchers have developed various directions to estimate the geographic locations of both twitter users and tweets. There are three trends on tweet geolocation prediction: event level, user level and tweet level.

The model based on event level mainly focuses on the location of events mentioned in text. This model relies on identifying geolocations in text and distinguishing between different toponym references. The event level model also has been studied by many group. Recent studies integrated metadata and named entity recognition into geoparsing. This model reached pretty high accruracy percentages about 90% [1] but the event geolocation inference may not be able to tell us the actual location of individual tweet.

The model based on user level focuses on the users' tweet history and other useful metadata. For example, user location can be predicted by using toponym references within their tweets or friend networks or timezones. The majority of the algorithms for user level location prediction uses either statistical models or machine learning. Qian et al. [12] designed a probabilistic machine learning graph model, obtaining high accuracy for predicting users' geolocations at the country or state level as shown in Figure 2.

The model based on tweet level focuses on location of indicidual tweets. It is different from user-level prediction because a tweet might be posted in a different place from where user live, such as during vacation or at work place. Generally tweet level prediction is more difficult. The content of a tweet might be containing toponyms that do not convey the tweet's actual origin and tweets may not contain sufficient metadata or useful details. But researchers have continued to explore potential solutions and machine learning models. Duong-Trung et al [3] developed near real-time geolocation prediction at the tweet level with a matrix factorization-based statistical regression model. The performance is shown in figure3 Lau et al. [8] designed deepgeo, an end-to-end neural network that combines various recurrent and convolutional neural networks for inferring tweet locations at city level, achieving a

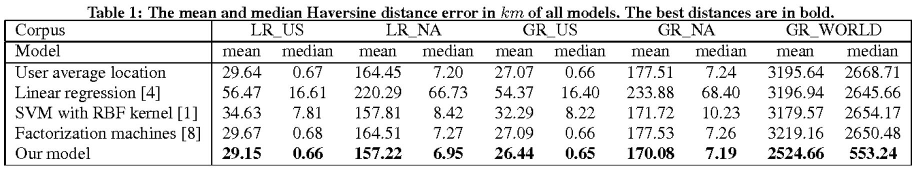


Table 1: The mean and median Haversine distance error in km of all models. The best distances are in bold.

Corpus	LR_US		LR_NA		GR_US		GR_NA		GR_WORLD	
Model	mean	median	mean	median	mean	median	mean	median	mean	median
User average location	29.64	0.67	164.45	7.20	27.07	0.66	177.51	7.24	3195.64	2668.71
Linear regression [4]	56.47	16.61	220.29	66.73	54.37	16.40	233.88	68.40	3196.94	2645.66
SVM with RBF kernel [1]	34.63	7.81	157.81	8.42	32.29	8.22	171.72	10.23	3179.57	2654.17
Factorization machines [8]	29.67	0.68	164.51	7.27	27.09	0.66	177.53	7.26	3219.16	2650.48
Our model	**29.15**	**0.66**	**157.22**	**6.95**	**26.44**	**0.65**	**170.08**	**7.19**	**2524.66**	**553.24**

Figure 3: tweet level model performance

state-of-the-art accuracy of approximately 40%.

Many of the models proposed very efficient algorithms. However, in 2020, many models' input are outdated. For example, twitter does not provide user timezone, user UTC offset anymore. Therefore, to predict the location of covid-19 related tweets, we need to modify the inputs of models improve with new techniques we have.

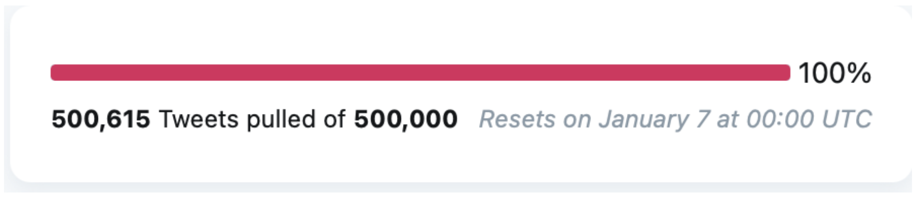
Our project focuses more on tweet level and user level: processing message/tweet along with some metadata in user profile to predict the location. We will be using the tweet content and tweet context in twitter information section and the Home location in user profile. We adapted an end-to-end neural network model and takes these features as input and output a geolocation tag for each tweet. To improve the previous model, we used word2vec model on text layer and compared the results between them.

3 Data Retrieval

3.1 Twitter API

At the very beginning, we planned to use the COVID-19 Twitter Steams API[15] which can directly provide the tweets related to COVID-19. However, since this API is deprecated, Twitter API called Filtered stream API with COVID-19 related tags is used instead. In order to get COVID-19 related tweets with filtered stream API, some keywords need to be specified. To cover more tweets related to COVID-19, two tags are specified as the parameters for our API request, "covid" and "corona", because most people mention COVID-19 as "covid-19", "corona virus", or "covid".

The data retrieval process is implemented by Python, which is provided by Twitter Development Platform[15]. Some changes and modification are done by us to store the returned record in a list data structure. Then, to manage the data extracted on different dates, the data of each time slot are stored in three pickle[11] files including all data, data with geo-tags, and data without geo-tags. In this project, we mainly deal with the data with geo-tags. Since each Twitter account can only make 500,000 requests to this API within one month, two accounts are utilized, which can make 1,000,000 requests in total. As the figure 4 shows, both accounts have achieved the request limitation.


100%

500,615 Tweets pulled of **500,000** *Resets on January 7 at 00:00 UTC*

Figure 4: Twitter Developer Platform API

3.2 Data Time Distribution

So, in total, we got roughly about 800,000 valid tweets in four days from Dec. 4th to Dec. 8th. However, we find that only 1% of these tweets have geo-tags. The total number of tweets with geo-tags is 9,693, which are used as the data set for our following model training and validation. As the filtered stream API is a real-time streaming API, all the tweets we got mainly cover 8 AM to 12 AM time slots. The time distribution of these tweets is shown in the figure 5. To minimize the effect of different hours on tweet texts, we cover as many time slots as possible. However, due to the network stability and bandwidth issue, as the left part of figure 5, which stands for Dec. 4th to Dec. 6th, shows, the number of tweets in each bin changes a lot. But from Dec. 7th to Dec. 8th, which stands for the right part of figure 5, tweets are extracted more evenly from the API.

We plan to extract the tweets from 8 AM to 12 AM because during these time slots, users are more likely leave their home and go somewhere else, which can provide more user location information for following model training process. Besides, as we mentioned in previous section, since this Twitter Stream API is a real-time API, if the data retrieval is performed during daytime, more tweets can be returned by the API increasing the size of our training data-set.

4 Data Pre-processing

After the data is extracted from the API and stored on local storage in pickle format, the data pre-processing is performed before sending the data to our model. Figure 6, a beautified version of the raw JSON data in the API response is shown. It mainly contains three fields, data, includes, and matching rules. The Data field contains the id, the text, and creation time of the tweet. The includes field mainly has user and tweet's metadata, which includes username and geo-location data. The matching rules field indicates which rule this tweet hits.

As figure 7 shows, some fields are extracted and form the input data, which is a part of data preprocessing task. To be more specific, the geo-location information within the places field in raw data is converted to the tweet city field in the processed data with 'city-country' format, which is the combination of the 'full_name' and 'country' field. This format is designed by [8]. Intuitively, this format can also avoid

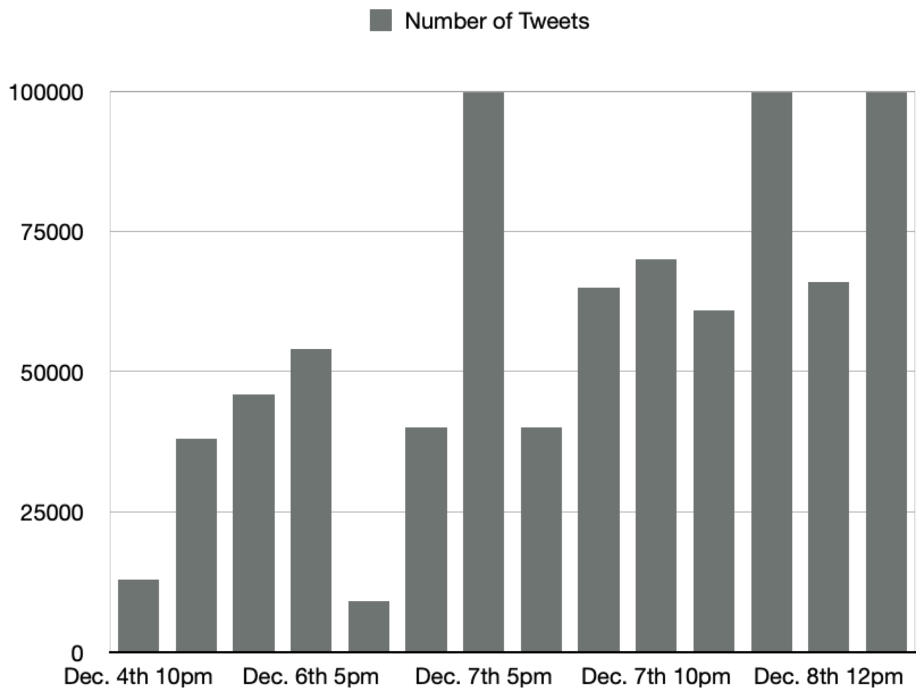
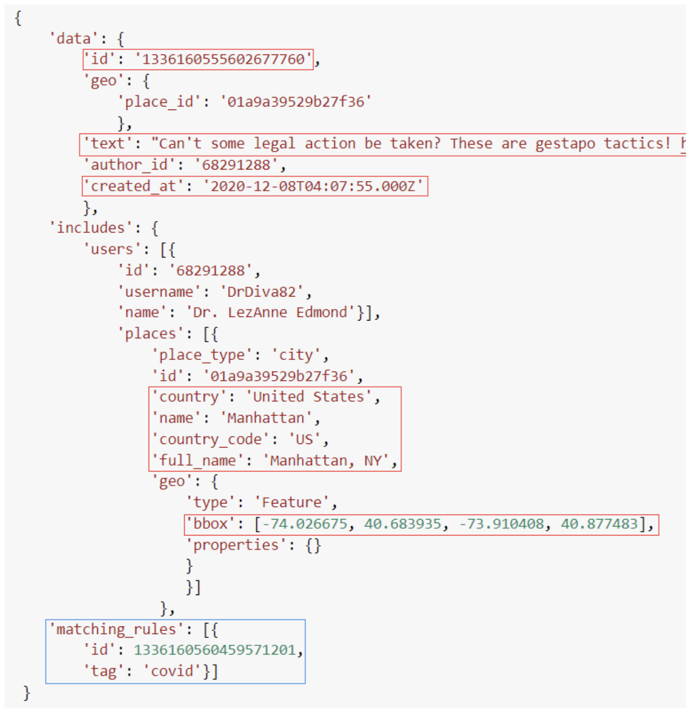


Figure 5: Tweets Data Retrieval Time Distribution combined with tweets number within each time slot (Time slots along X axis are not evenly distributed, mainly cover time from 8AM to 12 AM)



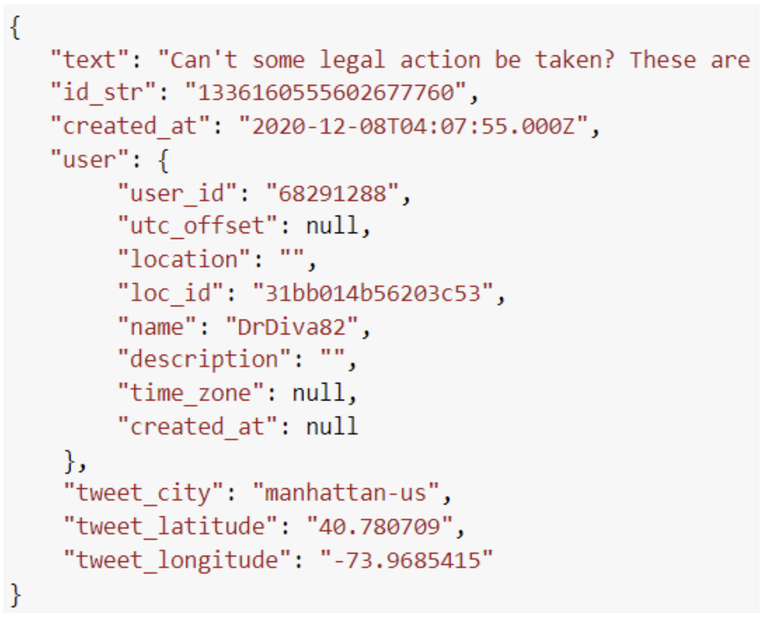
```
{
    'data': {
        'id': '1336160555602677760',
        'geo': {
            'place_id': '01a9a39529b27f36'
            },
        'text': "Can't some legal action be taken? These are gestapo tactics! h
        'author_id': '68291288',
        'created_at': '2020-12-08T04:07:55.000Z'
        },
    'includes': {
        'users': [{
            'id': '68291288',
            'username': 'DrDiva82',
            'name': 'Dr. LezAnne Edmond'}],
            'places': [{
                'place_type': 'city',
                'id': '01a9a39529b27f36',
                'country': 'United States',
                'name': 'Manhattan',
                'country_code': 'US',
                'full_name': 'Manhattan, NY',
                'geo': {
                    'type': 'Feature',
                    'bbox': [-74.026675, 40.683935, -73.910408, 40.877483],
                    'properties': {}
                    }
                    }]
                },
    'matching_rules': [{
        'id': 1336160560459571201,
        'tag': 'covid'}]
 }
```

Figure 6: Raw Tweet Data in JSON (Beautified version for better readability)

a conflict circumstance that two cities have the same name but different country. If just city names are provided, it can potentially weaken the model. The bounding box field which is ‘bbox’ in raw data, is averaged and output as tweet latitude and longitude in the processed data. The bounding box stands for the area of tweet’s geo-location. Some users may have location information in their user files, we also include that field if the user has this information. And this processed data works as the input of our model, which could be denoted as X.

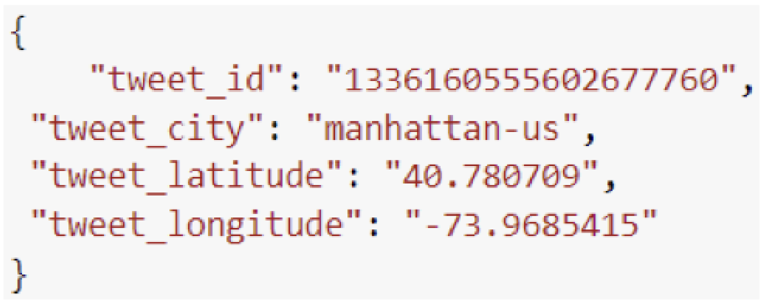
Besides the Input X, its corresponding label Y is also extracted from the raw data. In label Y, we not only include the exact city name but also combine the latitude and longitude for further reference. The high level idea of this input X and label Y is that each line in X corresponds to a line of label Y in the same position.

After retrieving and pre-processing the data, the raw JSON data is converted to a text file, in which each line is a JSON object with all the fields we need for training and validation. All the 9,693 tweets is processed using the same process. Also, to prepare for the evaluation section of our project, a data training validation split is performed with a 9 : 1 ratio. All the data is selected randomly and output as two sets.



```
{
   "text": "Can't some legal action be taken? These are
   "id_str": "1336160555602677760",
   "created_at": "2020-12-08T04:07:55.000Z",
   "user": {
        "user_id": "68291288",
        "utc_offset": null,
        "location": "",
        "loc_id": "31bb014b56203c53",
        "name": "DrDiva82",
        "description": "",
        "time_zone": null,
        "created_at": null
    },
    "tweet_city": "manhattan-us",
    "tweet_latitude": "40.780709",
    "tweet_longitude": "-73.9685415"
}
```

Figure 7: Processed Data in JSON (Input X)



```
{
    "tweet_id": "1336160555602677760",
 "tweet_city": "manhattan-us",
 "tweet_latitude": "40.780709",
 "tweet_longitude": "-73.9685415"
}
```

Figure 8: Processed Data in JSON (Label Y)

5 Approach and Implementation

5.1 Multiview Learning

There is a great deal of available information contained in a tweet. Other than the tweet message content itself, a number of raw Twitter metadata are also potentially valuable when it comes to predicting the geolocation of a tweet, including the tweet creation time, account creation time, location field specified by the account user in his/her profile, etc. What's more, network-based approaches for geotagging tweets also take users' social networks into account, which can be embedded and reconstructed by leveraging the mentions and following information among users. Faced with such rich features, an important question that needs to be answered is that how we are supposed to combine them and make the best use of them. Among many brilliant approaches in the literature, multiview learning[16] is an effective, powerful and generic paradigm that aims to encompass methods that learn different representations from different angles. This kind of method is thought to be able to utilize abundant knowledge of various aspects, and this turns out to be a perfect fit for our problem.

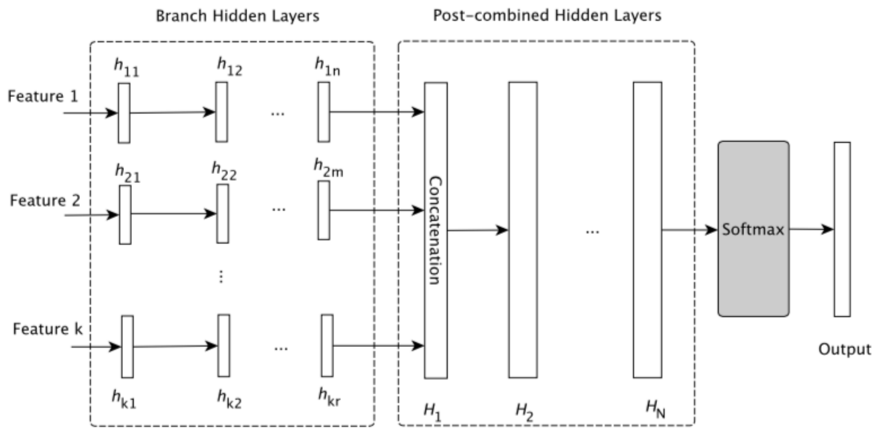


Figure 9: A Diagram of the Generic Architecture of MENET (from [2])

Do et al.[2] propose a multi-entry neural network(MENET) for Twitter geolocation at a user level, i.e., predicting the location of the users, instead of geolocating every single tweet. They combine the strengths of both content-based and network-based approaches by leveraging textual contents and metadata of tweets, and users' social connections with each other to geolocating every user. More specifically, their network incorporates four types of features from the pre-processed data: TF-IDF, doc2vec[9], node2vec[5], and timestamp of tweets, which is the time-related information of user behavior. The first two features respectively capture words and paragraph semantics of tweets from different points of view, while the node2vec represents the users' social ties. The fourth feature, which is 24-dimensional, records how many tweets a user posts every hour. Each type of feature constitutes one view of this multiview neural network. Each of the four features is fed to a branch of hidden layers(one hidden layer for each branch, according to the paper), after which their representations are concatenated and directly sent to the softmax layer.

They apply their model to the classification tasks on two datasets, GeoText[4] and UTGeo2011[13], which contain tweets from the U.S. only. The MENET does a pretty good job in both regional and state classification. They also address regression tasks of predicting the geographical coordinates of users from a classification point of view, by first partitioning the training set into small areas(each corresponding to a class), and estimating a new user's geographical coordinates using the centroid of data points in the class(small area) that it is assigned to. Their work outperforms state-of-the-art by a large margin.

5.2 Deepgeo

For the MENET based on multiview deep learning, abundant features from different views make it powerful enough for geolocation. The model is also quite generic and simple, because it does not need a specific architecture for each type of feature, and other than the four features mentioned in [2], it can also incorporate other features. However, the method takes the users' social networks into account by learning embeddings(node2vec) for nodes(users) in an undirected graph. Therefore, whenever a new user joins in the graph, the embeddings need to be learned again, which is quite inefficient. What's more, MENET is designed to deal with geolocation prediction of tweet users, but not every single tweet. For the limited data that we have retrieved from COVID-19 Twitter Streams, users' social connections information is unlikely to be extracted because almost all the tweets are posted by different users who do not know each other. MENET also takes too much feature engineering for each feature before feeding them to each branch of the multi-entry network. All these cons mean that MENET is not suitable for our project. But the key insight from MENET is that the multiview deep learning paradigm is very promising for the geolocation prediction problem.

Another end-to-end trainable neural network architecture for addressing the tweet-level geolocation prediction task in a hard classification setting is the deepgeo[8]. In our opinion, it also borrows the paradigm of multiview learning, so we think it may be a starting point of our project. It takes totally 6 features from raw Twitter metadata as inputs to the network, respectively tweet message, tweet creation time, user UTC offset, user timezone, user location and user account creation time. This network aims to predict the city tag of every tweet out of 3362 cities, and it outperforms several benchmark architectures. From the overall architecture of deepgeo shown in Figure 10, we can see that the 6 features are processed by separate networks that share no parameters to generate 6 embedding vectors, which are concatenated and become a long vector $\hat{f}$. The $\hat{f}$ is an overall embedding

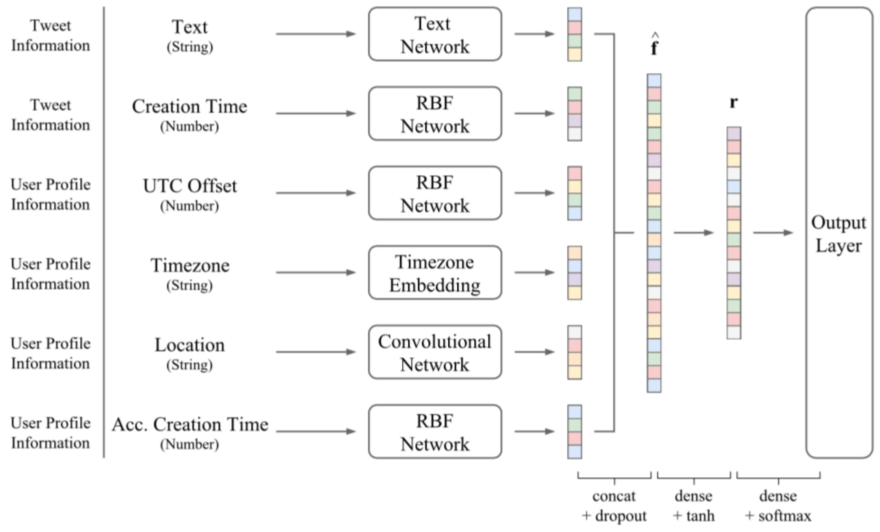


Figure 10: The Overall Architecture of deepgeo(from [8])

that this model learns for capturing the information in a tweet. There are totally 4 kinds of networks that are designed to respectively process 4 different kinds of features: Text Network, RBF Network, Convolutional Network and the Timezone Embedding. The Timezone Embedding network is simply a single-layer network that takes a one-hot timezone categorical feature as input and learns the embedding of timezone. The RBF Network is used to capture time information from the three time features, and aims to find out different activity patterns in different regions. The Convolutional Network is to extract the embedding from the user's self-declared location field, which is a free-text string. The Text Network is used to embed the text message. More details of the networks are as follows.

The RBF Network is responsible for processing the 3 time features in the meta-data of tweets, including tweet creation time, user account creation time and user UTC offset. They are all normalized to [0,1] and the aim of this network is to capture the time and activity patterns of tweets in different regions, since they believe regions across the world should have distinct activity patterns(which is more reasonable for regions with fairly different longitudes). The network splits time into B bins, and they use Radial Basis Functions(RBFs) to serve as the new features for embedding the time information. Specifically, each bin follows a Gaussian distribution with a mean μ_i and standard deviation σ_i, and the goal is to learn the μ_i and σ_i for each bin. A new input then can be embedded by a B-dimensional vector r, with each element r_i equal to:

$$\mathrm{r}_i = exp(\frac{-(\mu-\mu_i)^2}{2\sigma_i^2})$$

The embedded feature is believed to capture the preference difference of different

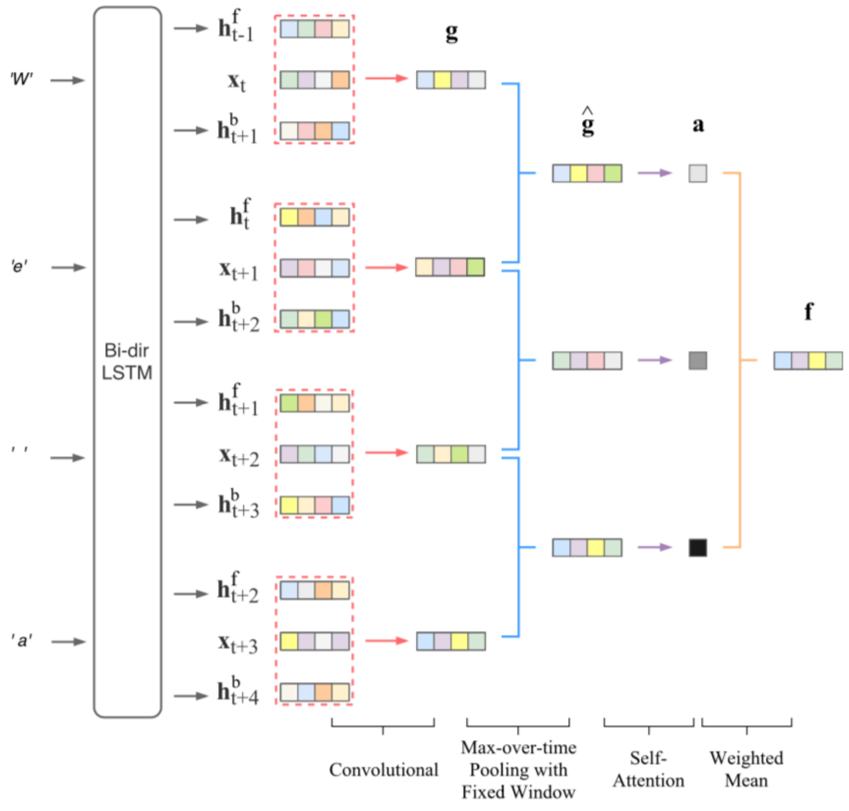


Figure 11: A Diagram of the Text Network(from [8])

Feature Set	**Accuracy**
All Features	0.428
−Text	0.342 (−0.086)
−Tweet Creation Time	0.419 (−0.009)
−UTC Offset	0.431 (+0.003)
−Timezone	0.422 (−0.006)
−Location	0.228 (−0.200)
−Account Creation Time	0.424 (−0.004)

Figure 12: Feature ablation results from [8]

regions towards these bins.

The Convolutional Network is for processing the location feature, which is a free-text string in a user's self-declared location field. It is a standard character-level convolutional neural network by Kim et al.[6] Briefly speaking, first, the character embeddings of all the characters in the string are concatenated, and O convolutional filters spanning Q characters generate an O-dimensional vector g_i for each span. The output vector of this network is the element-wise maximum of all the g_i's. More details can be found in [8] and [6].

The most important network in deepgeo is the Text Network as shown in Figure 11, which is a character-level recurrent convolutional network. For every character embedding x_t, it is fed to a bi-directional LSTM network. The forward hidden states, current character embeddings, backward hidden states are then concatenated, transformed by a fully connected layer and activated by ReLU to form g_t's. After that, they apply max-over-time pooling over P characters, i.e.,

$$\hat{g}_t = max(g_t, g_{t+1}, ..., g_{t+P-1})$$

Here, the max function is also an element-wise max function. The above vectors are further used to output the self-attention coefficients a_t's in front of themselves, that is, a_t captures the importance of $\hat{g}_t$. The final text embedding f is output by computing the weighted mean of $\hat{g}_t$'s over a_t's. This kind of self-attention mechanism is believed to "discover the saliency of a character span", which means "attending to location indicative words"[8] in a Twitter message.

5.3 Problems of Deepgeo

From the above sections, we can see that deepgeo is a powerful and generic architecture that takes a number of raw Twitter metadata as input and is able to predict the geolocation of every tweet. It is end-to-end trainable and outperforms several benchmark architectures. At the same time, the character level embedding technique makes it an language-independent model, allowing deepgeo to be extended to other languages easily. The approach does not need so much feature engineering as that in [2].

5.3.1 Twitter API Deprecation Problem

However, deepgeo was first proposed in 2015. As we are now approaching the end of 2020, some parts in this model become outdated and are no longer appropriate in geo-location prediction. As figure 12 shows, deepgeo takes six different features as input. But, as twitter recently updated their API, some of the features, including the user timezone and user UTC offset, are no longer provided by Twitter. What's more, according to the qualitative analyses in [8], tweet creation time and account creation time have a small and even negligible impact on the prediction performance of deepgeo. We believe that these are redundant features. Including them not only bring us no benefit but also burdensome, requiring more networks to be trained and more hyper-parameters to be tuned.

5.3.2 Random Embedding Initialization Problem

In deepgeo's Text Network, the tweet text embedding is randomly initialized. This potentially contributed to a lot of extra training to offset the randomness, which we think is quite inefficient. In addition, the character-level embedding may be too fine-grained for our project. The reason that they use character-level embedding was to build a language-independent model, but, in our case, the data we have retrieved from COVID-19 Twitter Streams is all in English. Therefore, it might be better for us to use some other language model to generate tweet embedding.

5.4 Deepgeo with Word2Vec

5.4.1 Brief Introduction to Word2Vec

Word2vec is a two-layer network that vectorizes words. The input is a text corpus and the output is a set of feature vectors corresponding to the words in the corpus.

The aim of Word2Vec is to create a vector space with contains all the vectors of similar words together by detecting similarities mathematically. The vector created by Word2Vec contains distributed numerical representations of word features such as the context of individual words. With enough data, Word2Vec can guess a word's meaning accurately only based on previous learning procedure. We can utilize the result to build an association network of the words or build different clusters of documents for classification by topic. The clusters are the basis of many applications such as searching, sentimental analysis etc. The output of the Word2Vec is a vocabulary where each word has a corresponding vector. The output can be fed into a deep-learning network for further training and analysis.

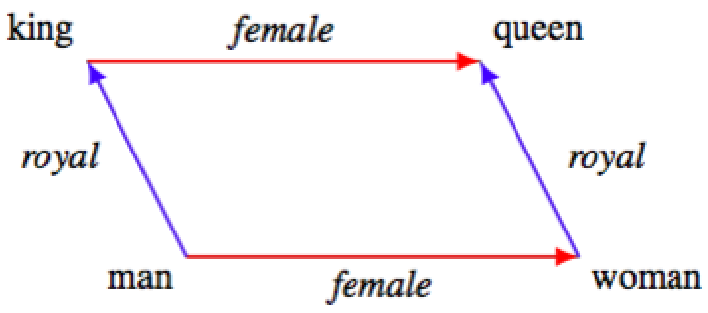


Figure 13: Vector Representation of Word2Vec

Generally, Word2vec can be applied to many other contents such as social media graphs and verbal series where some patterns can be discovered, not only limited to text corpus. This is because words are intrinsically discrete states like the other data in other kinds. Word2Vec embedding just describes the transitional probabilities between those states of words.

5.4.2 Integration of DeepGeo with Word2Vec

Originally, the character embeddings of deepgeo are initialized randomly with a uniform distribution and then put into a subsequent training process. This will negatively impact performance because the initial embedding weights are not learned and contain no information.

To solve the problem, we improved our model by replacing the weights with Google's Word2Vec embedding, which is inspired by [14]. Word2Vec is a pre-trained that provides embedding weights at word level. Word2Vec embedding represents similar words with similar vectors, and interrelated words are mapped to points close to each other in a high-dimensional space. It contains 3 million 300-dimensional word vectors pretrained on the Google News corpus.

Numerous machine learning algorithms have utilized Word2Vec and achieved great results in text classification because of many advantages. First, Word2Vec embeddings strongly capture semantic and syntactic relationships between words. Therefore, words with similar contexts share semantic meanings. Just as 13 shows here, the vector "King" - "Man" + "Woman" should be similar to the vector "Queen". Also, it's computationally efficient compared to other state-of-the-art methods.

Instead of original random uniform initializer, now we replace it with a pretrained word2vec embedding for the text layer. In our new model, each token (word) is sequentially represented with its vector embedding (Word2Vec) and concatenated with the forward and backward hidden states from a bi-directional LSTM network

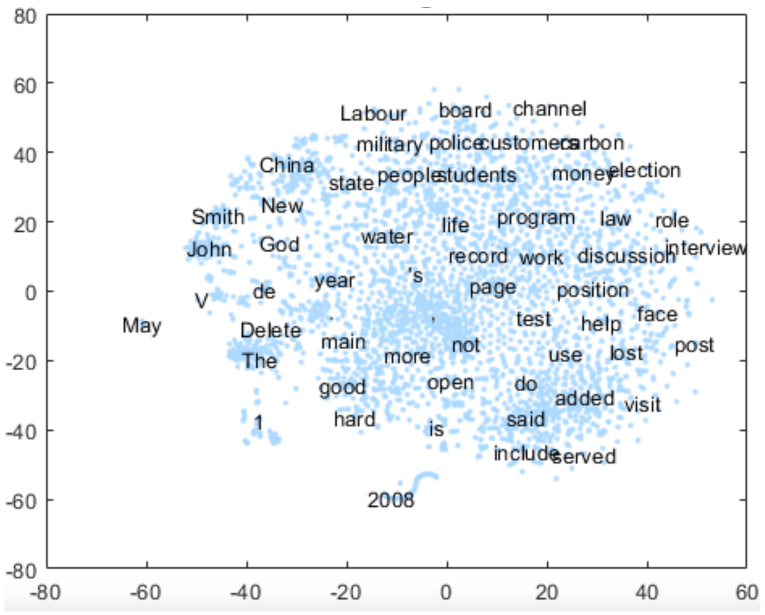


Figure 14: Vector Representation using Plots

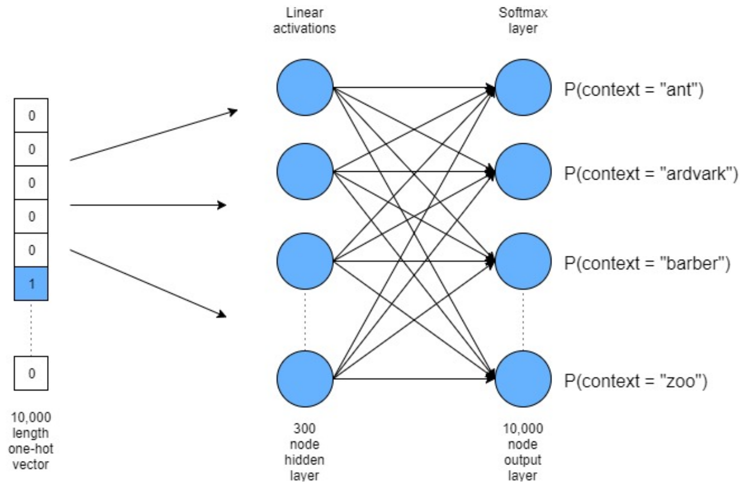
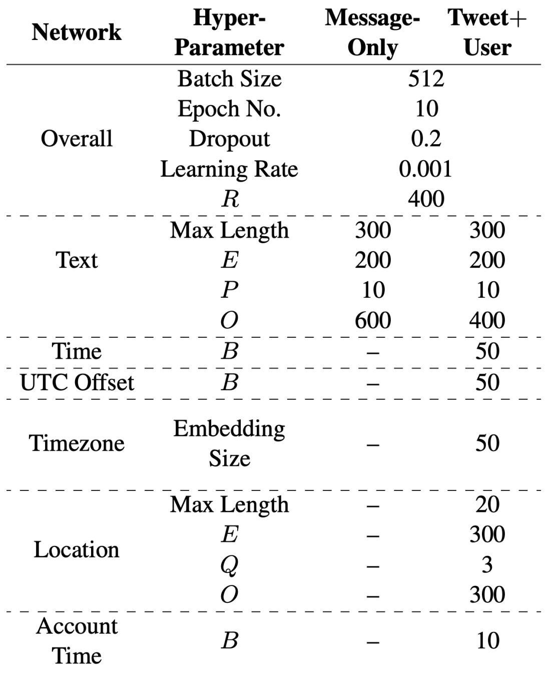


Figure 15: Architecture of Word2Vec Training Process



Network	Hyper-Parameter	Message-Only	Tweet+User
Overall	Batch Size	512	
	Epoch No.	10	
	Dropout	0.2	
	Learning Rate	0.001	
	R	400	
Text	Max Length	300	300
	E	200	200
	P	10	10
	O	600	400
Time	B	–	50
UTC Offset	B	–	50
Timezone	Embedding Size	–	50
Location	Max Length	–	20
	E	–	300
	Q	–	3
	O	–	300
Account Time	B	–	10

Figure 16: deepgeo hyper-parameters and values (from [8])

before applying max-overtime pooling, self-attention, and weighted mean which are all the existing architecture of deepgeo's text network. If a word's 300-dimensional vector is not present in the Word2Vec pre-trained model, we randomly initialize it. In this project, we experimented with different kinds of Word2Vec embeddings and compare their performance on our dataset. More analysis and evaluation of model performance are described in the following sections.

6 Evaluation and comparison

To evaluate the effectiveness of our improvement using Word2Vec embeddings, we first trained our adaptation of deepgeo twice: once with the original character-level embeddings and once with Word2Vec embeddings. The training processes were executed in the same manner and with the same optimized hyper-parameters that deepgeo originally used (figure 16) except for embedding dimensions. Due to Twitter terms of service, we can only retrieve a small number of Covid-related tweets using the API that twitter provides. So, our training data set includes only 8724 tweets. Also, because of time constrains for reporting our results and the limit number of training tweets, we used three different pre-trained word2vec models to train our deepgeo model.

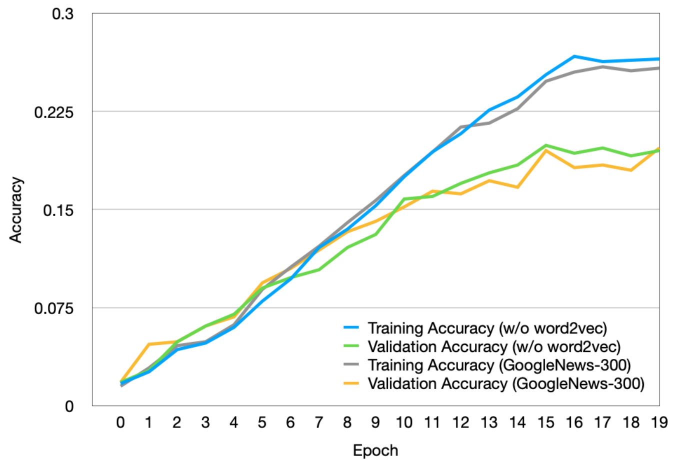


Figure 17: Training Accuracy and validation accuracy of model trained with original character embeddings and Word2Vec embeddings (GoogleNews-300)

The Word2Vec model we first use is GoogleNews-300 model which is pre-trained with Google News corpus (which has around 3 billion running words). As shown in figure 17, the original deepgeo model has a better performance than the model with Word2Vec embeddings. The highest validation accuracy is 19.9% achieved at the 15th epoch. Also, the training accuracy of the original model outperforms the Word2Vec model. This might because the Google-News-300 model which was published in 2015 is not suitable enough to extract major information in our Covid-related tweets.

To evaluate the effectiveness of different Wrod2Vec model, we used another two pre-trained word vector model glove-twitter-100 and glove-twitter-200. 100 and 200 represent the text embedding vector size. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. The Twitter corpus used to train this model has 2B tweets, 27B tokens, and 1.2M vocabularies. For every model in figure 18, the validation accuracy curve gets stable at the 15th epoch and the highest validation accuracy is 20.3% achieved with glove-twitter-200 model.

To further compare the performance of different models, the validation accuracy of each model is shown in figure 19. We can see that the Twitter-200 word embedding model has the highest validation accuracy which is 20.3%. Almost every model reaches the highest validation accuracy at the 15th epoch. The gap between these models is not significant, generally says within a gap of 2.6%.

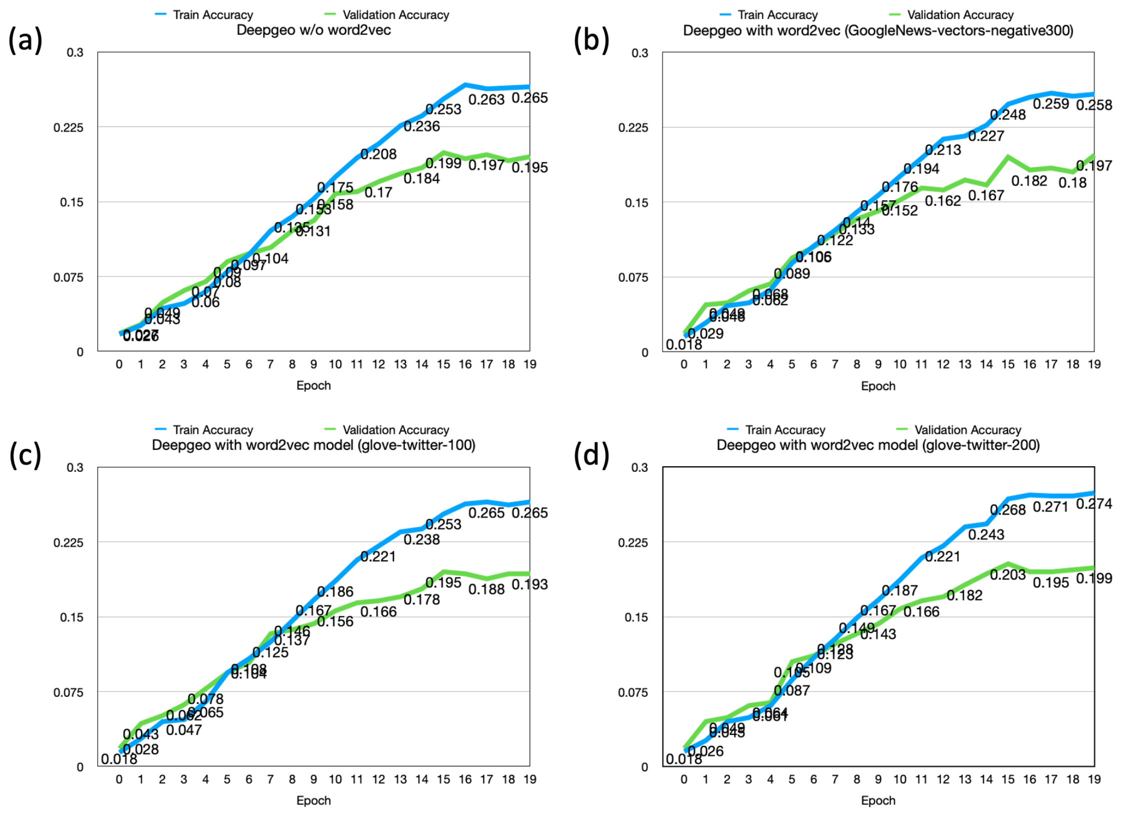


Figure 18: Training Accuracy and validation accuracy of different models

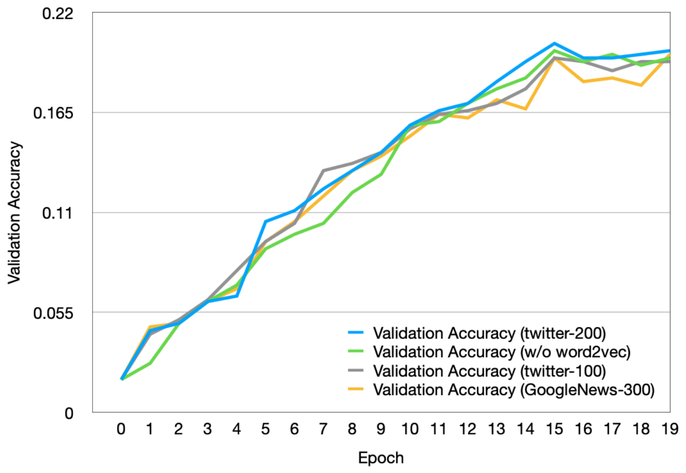


Figure 19: Validation accuracy of different models

7 Discussion and future work

Although we achieved a relatively good result, our project process and model still come with several defects. In general, the bias can be divided into three categories: data bias, feature bias and embedding bias. We are expecting a better model prediction accuracy if all of the following defects are resolved. In the following sections, we will talk in detail about each bias and its potential solution.

7.1 Data bias

7.1.1 Data Location & Time Period Bias

As we mentioned in previous section, the twitter api that we used to obtain tweets is a stream-like api. Even though it is possible to add some filters, such as filter by a specific tag, we still cannot control the time period and location of tweets we get. Whenever we call the api, we will always get the most up-to-date tweets from the stream. Therefore, tweets in our training dataset are mainly created between late November and early December. Given that covid-19 started to be a global health problem in January 2020, this definitely leads to data bias. For example, it might be possible that tweets created in early March, when the pandemic started its outbreak in the US, might carry more geo location information compare to tweets created in December. Besides, we also can't control the location of tweets. In our training data, we have tweets from different countries. Some countries has great amount of tweets while others only have a few. Intuitively, people from countries with more covid positive cases may tweet more and thus carry more geo location information.If we could get tweets that are equally distributed both from January to December and among different countries, this data bias can be resolved and we might observe an improvement in model performance.

7.1.2 Data size Bias

In this project, the total number of training data is quite limited. For now, although we obtained more than one million tweets that are tagged with covid, only about 1% of them, nearly ten thousand tweets, come with a geo location. That's because most of people don't like to include their location information in tweets. Obviously, training a model with only ten thousand piece of training data is not enough. This also explains why we only get about 20% of training accuracy. If more geo-tagged tweets could be obtained for training the model, we might be able to improve the prediction performance.

7.1.3 Data Independence Bias

In our model, we treat each tweet as an independent single object. We care only about its text data as well as some other meta information. But, in reality, tweets could also depends on the user who tweet it. In other words, tweets from the same user contains some underlying connections. We could group tweets by user id and analyze them as together. Since a user might not move around so often, if one of the tweets in its timeline contains some geo location information, it will be reasonable to increase the probability of other tweets being predicted to the same location. In this way, we might find more hidden clues about user location, which will result in a performance improvement.

7.2 Feature Bias

The input feature of model is biased. In our model, one of our input features is “user location”. This field is set by user. It is possible that some user randomly selects a city as their profile location. This will greatly impact the performance of our model. Currently, we still can’t solve this problem. But, maybe we can try taking geo location information from user’s timeline and compute the probability.

7.3 Embedding Bias

The text embedding process in our model could also be improved. As we all known, there are a lot of stop words, abbreviations and emojis appearing in tweets. Currently, the way we construct the embedding for each tweet is that we split the tweet by white space and apply word2vec model on each word to create a 300-dimension word embedding. We then add these embeddings together and average it by the size of text to obtain final tweet-sentence embedding. However, these stop words and emojis usually don’t contain too much useful semantic information. Mixing them with ”regular” text embedding will somehow contaminate the embedding. In the future, to better capture semantic and location information contained in tweets, we could develop a tokenizer or a filter that draws out these unrelated words and characters.

Besides the text embedding bias, the word2vec model that used to create text embedding could also be improved. Because of the limitation of time and computation resources, we are unable to train our own word2vec model using tweets we downloaded. Instead, we downloaded the pre-trained glove-300 model. This is a model pre-trained with two billion tweets. Since the original purpose of glove model is not for geo location prediction, directly using this for embedding may not offer the best vector representation that captures the geo information. In the future, if more data is available, we can fine tune the pre-trained word2vec glove model to make it better fit for geo location prediction.

8 Roles of team members

Ziqi Li: Data retrieval, original Deepgeo analysis, generate Deepgeo baseline model result, word2vec embedding preliminary investigation, write report
Bowen Zhang: Data processing, help with word2vec embedding preliminary investigation, model future improvement analysis, write report
Yuhong Jiang: Data retrieval, multiview learning literature investigation, original Deepgeo analysis, problems and improvement analysis, write report
Jiangtao Chen: helped with data retrieval, incorporated Wrod2Vec model into deepgeo model, Trained deepgeo model with pre-trained Word2Vec model and evaluated the results.
Tianyu Zhang: helped with original Deepgeo analysis and word2vec embedding, generate original Deepgeo baseline model result, write report
Jiapeng Wan: help with data retrieval, analysis of Deepgeo with Word2Vec, write report

References

[1] J. D. Bruijn et al. "TAGGS: Grouping Tweets to Improve Global Geotagging for Disaster Response". In: *Natural Hazards and Earth System Sciences* (2017), pp. 1–22.

[2] Tien Huu Do et al. "Multiview Deep Learning for Predicting Twitter Users' Location". In: *CoRR* abs/1712.08091 (2017). arXiv: `1712.08091`. URL: `http://arxiv.org/abs/1712.08091`.

[3] Nghia Duong-Trung, N. Schilling, and L. Schmidt-Thieme. "Near Real-time Geolocation Prediction in Twitter Streams via Matrix Factorization Based Regression". In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (2016).

[4] Jacob Eisenstein et al. "A latent variable model for geographic lexical variation". In: *Proceedings of the 2010 conference on empirical methods in natural language processing.* 2010, pp. 1277–1287.

[5] Aditya Grover and Jure Leskovec. "Node2vec: Scalable Feature Learning for Networks". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 855–864. ISBN: 9781450342322. DOI: `10.1145/2939672.2939754`. URL: `https://doi.org/10.1145/2939672.2939754`.

[6] Yoon Kim. "Convolutional neural networks for sentence classification". In: *arXiv preprint arXiv:1408.5882* (2014).

[7] Abhinav Kumar and Jyoti Prakash Singh. "Location reference identification from tweets during emergencies: A deep learning approach". In: *CoRR* abs/1901.08241 (2019). arXiv: `1901.08241`. URL: `http://arxiv.org/abs/1901.08241`.

[8] Jey Han Lau et al. "End-to-end Network for Twitter Geolocation Prediction and Hashing". In: *CoRR* abs/1710.04802 (2017). arXiv: `1710.04802`. URL: `http://arxiv.org/abs/1710.04802`.

[9] Quoc Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents". In: *Proceedings of the 31st International Conference on Machine Learning.* Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, 22–24 Jun 2014, pp. 1188–1196. URL: `http://proceedings.mlr.press/v32/le14.html`.

[10] S. E. Middleton, L. Middleton, and S. Modafferi. "Real-Time Crisis Mapping of Natural Disasters Using Social Media". In: *IEEE Intelligent Systems* 29.2 (2014), pp. 9–17. DOI: `10.1109/MIS.2013.126`.

[11] *Pickle Library, https://docs.python.org/3/library/pickle.html.* URL: `https://docs.python.org/3/library/pickle.html`.

[12] Yujie Qian et al. "A Probabilistic Framework for Location Inference from Social Media". In: *CoRR* abs/1702.07281 (2017). arXiv: 1702.07281. URL: http://arxiv.org/abs/1702.07281.

[13] Stephen Roller et al. "Supervised text-based geolocation using language models on an adaptive grid". In: *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*. 2012, pp. 1500–1510.

[14] Luke S. Snyder et al. "City-level Geolocation of Tweets for Real-time Visual Analytics". In: *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery* (2019).

[15] *Twitter Developer, https://developer.twitter.com/en/docs/labs/covid19-stream/overview.* URL: https://developer.twitter.com/en/docs/labs/covid19-stream/overview.

[16] Jing Zhao et al. "Multi-view learning overview: Recent progress and new challenges". In: *Information Fusion* 38 (2017), pp. 43–54. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2017.02.007. URL: http://www.sciencedirect.com/science/article/pii/S1566253516302032.