COMP30027 Report

anonymous

1 Introduction

This project is about to pick suitable machine learning strategy to predict the geotag based on the text generated by user. I will discuss my work from the following perspective: data preprecessing, Feature engineering, Models that I explored.

2 Related Work

There are a lot of impressive works has already been done in predicting geo-tag based on the information generate by social media users. Most of the related work does not use tweets alone, they always combine it with content in other media (like Blog) or user meta-data. For initial effort, a lot of sophisticated supervised, unsupervised model was used. For example, Wing and Baldridge attempt geodesic grids classifications using supervised models. Hecht et al. try to predict users' geo-info by using CALGARI algorithm. Pappas et al. (2018). In the following year, researchers are focusing more on feature selection techniques. As this is a era for deep learning, a lot of work are also based on deep learning method, for example GCN. Rahimi et al. (2018).

3 Data Preprocessing

It is really important to have a good approach to transfer the plain text to useful new features to feed into the model. Better data preprocessing strategy always produce more usable information and more choice for our later feature engineering.

3.1 Why Not *-top* Files

Up to 78% of the instances contain only 0 for all the attributes in this data representation. Even if I could correctly predict over 70% of the non-zero labels data and use 0-R to classify the others, 34.9% accuracy could be reached overall. However, it's even impossible to achieve 40% on the non-zero part based on my discovery. Also, the real world data wouldn't be evenly distributed. In this case, the data could not be very useful.

3.2 Raw File Text Precessing

Plain text can be hardly used but words are easier. Therefore, the basic step is to split the text by space. Next, there are still a lot of noise and redundancy in the feature space. The primary thing to do is to remove all the punctuation, unicode and stop words which are widely used but uninformative. Use too many of these things may cause overfitting. After that, by sampling some text from the tweets, I notice that the word after # and might be useful as they may contains information hight correlated to one of the class. For example, "BrisbaneNews9". Those words are split into smaller words by capital letter. Notice that some text may like this "#LouisWhyAreYouAnEGG", the EGG at the end won't be split. After this process, all words are transfer to lowercase to prevent mismatching.

4 Feature Engineering

Feature engineering is another very important part. Based on the performance of my preprocessing strategy, same set of data gain over 3-5% [1] increase in accuracy when comparing to using train-top100 and simple space-split feature.

4.1 Frequency Based

Since the feature space are still very large, but number of instance are relative small. To avoid overfitting and large variance of our model, top 20000 most frequent words among all tweets are selected. Two difference representations are used. They are one-hot ,and number of ranking in frequency (start from 1 and then use

[1] figure obtained by 5-fold CV on train-raw + dev-raw

0 padding to extend to the same length) which is preferred by word embedding.

4.2 TF-IDF

TF-IDF is a commonly used feature selection tools in NLP. The TF part measure how frequent we could see a word in a document, IDF represent the incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely. I process the text both in word level and char level (with n-gram range 2-6 from word to char). The feature only using word level features we call it **WORD**, and the feature combine both word level and char level features we call it **CHAR** However, using char level features does not give us a better results but require a lot more precessing time, therefore, it was discard eventually. By using grid search on the tfidfVectorizer, setting n-gram range to (1,2) and min_df to 0.3 would results in a highest train - dev accuracy. However, when combine dev and train set and implement a 5 fold cross validation, the default setting has slightly higher accuracy and low variance. Therefore, we decided to keep the hyper-parameters in default, and it has proved the performance on the test set.

4.3 Location Mapping

As those tweets information are generated by social media user, the information almost always represented in an unexpected way. In this scenario, some word mapping rules are created by look at the training data. For example, use regular expression to match word starting or ending with ”melb”, and turn it into Melbourne. The MNB classifier gain a huge boost after doing this. It is believed that the mapping could give the more frequent word like ”melbourne” more weight. Predictions could also be guarded by this mapping by using the idea of ensemble learning. We loop through the raw text and only assign the class label to the text which only contains the features from one location. If it contains more than one, for instance, the text is like ”traveling from Brisbane to Melbourne”, we will leave it alone. By combining those two results, the accuracy increase from 35.9 to 36.4 on public leaderboard.

5 Models

In this project, a wild range of different models was considered. Table below contains part of the model I used trained on train-raw and tested on dev-raw.

Model	WORD	CHAR
MNB	34.3%	34.7%
Logistic Regression	33.5%	33.4%
Random Forest	33.2%	33.1%
adaBoost	34.6%	34.0%

5.1 Multinomial Naive Bayes (sklearn)

- NB is normally the first approach people want to try when doing a NLP task. Also, It could handle multi-class problems naturally and yield some decent result. Since we know this is a multi-class prediction and the data distribution follows a multinomial distribution, we chose the MNB classifier.
- By looking at the confusion matrix, it works well on Brisbane (38%) and Sydney (36%) , but not very well on Perth(28%) with default hyper-parameters as they fit the problem quit well. The f1 score was quite close to the prediction accuracy when using cross validation(46%). [2] That means the model balanced the precision and recall quite well, which also means that it will be robust against new data.
- It is the best model among all the model that I tried. Due to the way we processed the raw data and the premise I made which is that the relation between word and word in a sentence is weak. It is very unlike a combination of two or few more very close words could help us predicting which proved by using 2-gram does not produce better results in word level. Normally, we could found that one single important word could weighted the change the prediction quite a lot. Thus, making independent assumption would not be a big problem here and could dramatically simplified the problem.

5.2 Word Embedding (Tensorflow)

- The previous mentioned frequent based method was used to transfer the text to embedding-layer-usable data to feed in a random initialized embedding layer. A

[2]mean accuracy for 5-fold cross validation on train+dev is 47%

Bi-direction LSTM layer was used directly blow the embedding layer for text analysis. [3] A bi-direction is a common tool for NLP task, it will look at the text both forward and backward which give the model extra chances to look at the text and against forgetting effect. However, overfitting is a big problem. Therefore, multiple dropout layers are applied.

- The result are not quite impressive, it only achieved 34.3% accuracy on dev set and relatively low F1 score. A great accuracy boost on Perth has been noticed, which gives me an idea that word counting features might benefit Perth. I'm afraid that the model may not works well on unbalanced new data. thus, it does not becomes our final choice.

6 Conclusions

In conclusion, I found out that the most important part of this project is feature engineering. It is true that some model could have a very good pattern recognition ability, nevertheless, bad preprocessing may mislead the model and make the problem non-learnable. By using the same processed data, the difference between best model and worst model are pretty small. Intuitively, better pre-processing method could extract more useful information from the raw data. Consequently, models could gain huge advantages by using these "extra" information.

References

Alexandre KOWALCZYK. 2019. GUR's and LSTM's. `https://towardsdatascience.com/grus-and-lstm-s-741709a9b9b1`. [Online; accessed 19-May-2019].

Konstantinos Pappas, Mahmoud Azab, and Rada Mihalcea. 2018. A Comparative Analysis of Content-based Geolocation in Blogs and Tweets. *arXiv e-prints*, page arXiv:1811.07497, Nov.

Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. 2018. Semi-supervised User Geolocation via Graph Convolutional Networks. *arXiv e-prints*, page arXiv:1804.08049, Apr.

[3] high level understanding of LSTM Alexandre KOWALCZYK (2019)