COMP30027 Project2 Report

Anonymous

1 Introduction

Nowadays, social media becomes one of the most popular approaches to share and acquire information. Tons of messages are posted with hidden information related to location, events and so on. Automatic geolocation is a common analytic method that predicts possible posted location and particularly useful in fields like disaster detecting [1] and stock market monitoring [2].

In this paper, we compare several different feature engineering strategies and classifiers on the effectiveness of text-based tweets geolocation prediction.

2 Related Work

Text Approach uses text to predict location. Some combined traditional machine learning models with feature extraction strategies, for instance, extracting local entities [4] from text using clustering algorithms like DBSCAN, selecting words from probabilities prospective [5], or using word embedding [6, 7, 8, 9]. Other common models include multilayer model [3] and Neural Network based model [10, 11].

3 Text Data

The data were directly collected from Twitter API with equal distribution over 4 cities (Melbourne, Sydney, Perth, and Brisbane). The dataset is consisted of 248,828 tweets and divided into train (103,364), development (37,316) and test (108,148) based on time collected, where only train and development have actual label that could use for evaluation.

4 Experiment Approach

4.1 Data Preprocess

The main purpose of preprocessing was to filter out obvious infrequent/location-irrelevant words, like URL, emojis, special characters (@, #), numbers, punctuation symbols and stop words. The rest were tokenized, transformed to lowercase and preserved irrespective their length since Roller [12] mentioned short words contribution to geolocation prediction, more exploration about word length was done later.

We also tried other approaches like stemming and check spell, but eventually discard them since the over/under stemming and time-consuming problems.

After preprocessing, the size of unique words of train, development and test datasets were reduced to 72819, 40808 and 74319 separately.

4.2 Feature Engineering

Although preprocessing eliminated some unhelpful words, the data still contained lots of noises, like nan, like. Also, plenty of words appear in development but not in training due to the characteristic of tweets, people are likely to misspell and concatenate words together, like dandedong (188 times), perthfest (38 times). Therefore, we implemented and tested several feature engineering strategies to help address these problems.

The first one was provided from COMP30027, top words related to each city based on the Mutual Information and Chi-Square were selected.

Like MI, Konstantinos [5] used probability-based approach Word Locality Heuristic (WLH) to determine words that are more useful, we used it as our second strategy. After WLH or MI feature construction was done by one hot encoding, noise words were filtered out and unseen words were simply ignored.

Lastly, we combined TF-IDF feature with following methods. To eliminate noises, we simply assumed the train was general enough to obtain the probability of one label given a word ($P(c|w)$), words with maximum $P(c|w)$ higher than threshold t were kept, also we eliminated words with length smaller than l or total count less than c. As for unseen word, one was replacing it with most similar existing words calcu-

lated from Word2Vec. Another one was breaking all words into substring of length s, since most unseen words could be found in training with few alter/insert/delete operations or concatenate them together. Finally, we consider all tweets from the same city as one document and merge them together to better obtain TF-IDF.

4.3 Classifier

Besides chose Majority-class and stratified Dummy classifiers as baseline. we ran several comparative tests on easy-interpret classifiers, like linear Support Vector Machine [5], Bernoulli Nave Bayes, Multinomial Nave Bayes [3, 5, 8] and Logistic Regression [5], also the combination (voting and stacking ensemble) of them, where voting integrated MNB, SVM and LR with hard voting, stacking put MNB and BNB as lower layer and use LR as higher layer.

5 Experiment Results

5.1 Evaluation Metrics

To measure the effectiveness of different approaches, we used execution time, accuracy (ACC), macro-averaging precision, recall, and fscore since the classes were distributed equally, also included results distribution, cities with most correct/incorrect predictions, and the most common error, which were all obtained from the confusion matrix.

5.2 Results

Although we try various combination of features engineering hyper-parameters, table 1 and 2 summarize only part of the results due to page limitaion, bold means the highest value of each column. The best classifier was determined by first ACC and then Time.

5.3 Interpretation and Error Analysis

As presented in table 5.1 and 5.2, MNB and Voting Ensemble classifiers performed more effective with accuracy around 35.5%, so the interpretation and error analysis mainly focused on these classifiers.

Probability-based model like MNB, BNB performed better after filtering out unrelated words, since the appearances of them largely decreased the posteriors probabilities and resulted in lower prediction authenticity. After leaving out them, NB predicted each instance based on the number of informative words inside and how pervasive were these words. For instance, 84.19% of melbourne were posted from Melbourne which made it a strong indicator of Melbourne. Also, we chose Laplace smoothing with alpha 0.01, means that we considered new values less important than the existing values since most of them are just interferences, like pleaseee.

As for classifiers assigned parameters to attributes, like SVM, LR, the impact of unrelated words was reduced by low associated parameter, so that even without filtering, they could still perform automatic feature selection and produce nearly equal distributed results combing with MNB. LinearSVM created hyperplane dividing informative words (with high-value parameter) into groups and didnt care about the distribution of irrelevant words (with low-value parameter), texts with informative words were more likely to fall into the same side of hyperplane and thus being classified as the same label. Here we adopted penalty C as 0.95 to ignore special instances, like Melbourne tweets w sydney, and generate more generalized model. Similarly, LR would assign high-value to label-correlated words and the appearance of these words will produce more confident prediction that closer to 1 (if 1 is the label associated). We used lbfgs solver with l2 penalty that was suitable for multi-class problems and handling multinomial loss, also changed the inverse of regularization strength to 0.95 allowing few mistakes.

However, irrespective types of feature engineering methods and classifiers, the main error was that the distribution of results was bias towards guessing one label due to the lack of information, as some tweets were just lyric or advertisement that was inherently unpredictable. 15 20% instances were empty after filtering (even more when using MI or WLH), and some others contained only unseen words that partially caused by sample bias from time-based data dividing. And if we didnt do that or did inappropriately, common words with little bias towards specific city, like nan prefer Perth in train but Brisbane in development, would also lead to a higher chance of error.

Another unavoidable error was the wrong indicator, like melbourne in Sydney tweets, NB would always predict towards Melbourne based on the posterior probabilities it learned, LinearSVM and Logistic Regression would give Melbourne higher output due to high-value associated parameter, unless there were more other strong indicators of other cities in this tweet.

Classifier	Best Classifier	ACC	Precision	Recall	Fscore	Time
Baseline	Dummy(maj)	0.25	0.0625	0.25	0.1	0.126616
Baseline	Dummy(strat)	0.250509	0.252651	0.252653	0.252651	0.374720
MI (top10)	MNB	0.294914	**0.662155**	0.294914	0.194591	0.754906
MI (top50)	MNB	0.301345	0.533528	0.301345	0.216524	1.011281
MI (top100)	MNB	0.307830	0.491527	0.307830	0.236774	1.516725
WLH (top 10%)	MNB	0.258575	0.571608	0.258575	0.119207	0.245079
WLH (top 30%)	Voting Ensemble	0.299710	0.652912	0.299710	0.202381	20.24341
WLH (top 50%)	MNB	0.329108	0.573644	0.329108	0.263804	0.217443
TF-IDF 1	Stack	0.343873	0.346076	0.343873	0.342224	2.298197
TF-IDF 2	MNB	0.339452	0.384120	0.339452	0.317352	0.111144
TF-IDF 3	Voting Ensemble	0.353012	0.354618	0.353012	**0.352226**	58.89453
TF-IDF 4	MNB	**0.356120**	0.371116	**0.356120**	0.350733	0.153969

1 TF-IDF t=0 + l=0 + c=0

2 TF-IDF t=0.5 + s=6 + l=3 + c=10 + merge tweets from same city + replace unseen words

3 TF-IDF 3: t=0 + s=6 + l=0 + c=0

4 TF-IDF 4: t=0.5 + s=6 + l=3 + c=0 + merge tweets from same city

Table 1: accuracy and time related measurement

Classifier	Best Classifier	Distribution	Most correct	Least correct	Common error
Dummy(majority)	Dummy(maj)	100% Bris	Bris	Other	$Other \rightarrow Bris$
Dummy(stratified)	Dummy(strat)	25.55% Perth	Melb	Syd	$Bris \rightarrow Perth$
MI (top10)	MNB	93.34% Bris	Bris	Syd	$Syd \rightarrow Bris$
MI (top50)	MNB	89.49% Bris	Bris	Syd	$Syd \rightarrow Bris$
MI (top100)	MNB	98.55% Bris	Bris	Perth	$Melb \rightarrow Bris$
WLH (top 10%)	MNB	85.56% Bris	Bris	Syd	$Syd \rightarrow Bris$
WLH (top 30%)	Voting Ensem	93.22% Syd	Syd	Melb	$Melb \rightarrow Syd$
WLH (top 50%)	MNB	86.04% Bris	Bris	Syd	$Syd \rightarrow Bris$
TF-IDF 1	Stack	32.99% Syd	Syd	Bris	$Bris \rightarrow Syd$
TF-IDF 2	MNB	60.45% Bris	Bris	Syd	$Syd \rightarrow Bris$
TF-IDF 3	Voting Ensem	28.55% Bris	Bris	Perth	$Syd \rightarrow Bris$
TF-IDF 4	MNB	43.32% Bris	Bris	Perth	$Perth \rightarrow Bris$

Table 2: Results distribution

6 Conclusions

In this report, we examined the performance of several feature engineering methods and classifiers that were commonly used in text geolocation tasks, using tweets data collected from 4 cities.

Firstly, we experimented words selection using MI and WLH, and use one-hot encoding to transform to features, resulted in around 29% - 33% accuracy using MNB and hard voting ensemble classifier combing MNB, LinearSVM, and LR.

Secondly, we combined TF-IDF features with other approaches together, like breaking words into substrings, selecting words based on P(c—w), length and overall count, merging all tweets from the same city into one document, replacing new words with most similar existing words from Word2Vec. Then accuracy was increased to around 35.5% with the same classifiers.

Finally, we summarized and interpreted the behaviors of classifiers above according to their hyperparameters, parameters, and theory be-

hind. Also, we analyzed the potential reason of errors, like tweets only contained irrelevant words or new words were inherently unpredictable. The appearance of strong indicator of wrong city, like sydney in Melbourne tweet, would unavoidably make both probability-based and parameter-assignment-based models predict incorrectly.

References

[1]: Ashktorab, Zahra, Christopher Brown, Manojit Nandi, and Aron Culotta. "Tweedr: Mining twitter to inform disaster response." In ISCRAM. 2014.

[2]: Mittal, Anshul, and Arpit Goel. "Stock prediction using twitter sentiment analysis." Standford University, CS229 (2011 http://cs229. stanford.edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentiment-Analysis.pdf) 15 (2012).

[3]: Tang, Haina, Xiangpeng Zhao, and Yongmao Ren. "A multilayer recognition model for twitter user geolocation." Wireless Networks (2019): 1-6.

[4]: Cheng, Zhiyuan, James Caverlee, and Kyumin Lee. "You are where you tweet: a content-based approach to geo-locating twitter users." In Proceedings of the 19th ACM international conference on Information and knowledge management, pp. 759-768. ACM, 2010.

[5]: Pappas, Konstantinos, Mahmoud Azab, and Rada Mihalcea. "A Comparative Analysis of Content-based Geolocation in Blogs and Tweets." arXiv preprint arXiv:1811.07497 (2018).

[6]: Lim, Kwan Hui, Shanika Karunasekera, Aaron Harwood, and Yasmeen George. "Geotagging tweets to landmarks using convolutional neural networks with text and posting time." In Proceedings of the 24th International Conference on Intelligent User Interfaces Companion (IUI19). 2019.

[7]: Ajao, Oluwaseun, Deepayan Bhowmik, and Shahrzad Zargari. "Content-aware tweet location inference using quadtree spatial partitioning and jaccard-cosine word embedding." In 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 1116-1123. IEEE, 2018.

[8]: Do, Tien Huu, Duc Minh Nguyen, Evaggelia Tsiligianni, Bruno Cornelis, and Nikos Deligiannis. "Twitter user geolocation using deep multiview learning." In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6304-6308. IEEE, 2018.

[9]: Thomas, Philippe, and Leonhard Hennig. "Twitter geolocation prediction using neural networks." In International Conference of the German Society for Computational Linguistics and Language Technology, pp. 248-255. Springer, Cham, 2017.

[10]: Rahimi, Afshin, Trevor Cohn, and Timothy Baldwin. "A neural model for user geolocation and lexical dialectology." arXiv preprint arXiv:1704.04008 (2017).

[11]: Rahimi, Afshin, Trevor Cohn, and Tim Baldwin. "Semi-supervised user geolocation via graph convolutional networks." arXiv preprint arXiv:1804.08049 (2018).

[12]: Roller, Stephen, Michael Speriosu, Sarat Rallapalli, Benjamin Wing, and Jason Baldridge. "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, pp. 1500-1510. Association for Computational Linguistics, 2012.