**1. Introduction**

Nowadays, social media such as Twitter, becomes one of the most popular approaches for people to share and acquire information. There are tons of messages being posted online with potential hidden information related to the user's location, events around and so on. Automatic geolocation is one of the common analytic methods that analyzes possible posted location of online texts, and is particularly useful in fields like disaster detecting [1] and stock market monitoring [2].

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

**2 Related Work**

Text Approach uses text information and several different approaches to predict location. Some of them combining traditional machine learning classifiers with different 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 approaches include multilayer model [3] and Neural Network based method [10, 11].

**3 Text Data**

The text data is directly collected from Twitter using the Twitter API, used only for research purpose that follows the Twitter Terms of Service. We choose same number of tweets from 4 cities in Australia (Melbourne, Sydney, Perth and Brisbane). The text dataset consists of 248,828 tweets and is being randomly divided into train (103,364), development (37,316) and test (108,148) set, where only train and development sets have correct location label that could be used for evaluation.

**4 Experiment Approach**

**4.1 Data Preprocess**

The main purpose of preprocess is to filter out obvious infrequent/geolocation-irrelevant words in text, such as URL, emojis, special characters (like @ and #), numbers, punctuation symbols and stop words. The rest words are tokenized and transformed to lowercase, and all being preserved irrespective their length since Roller [12] mentioned short words could also contribute on geolocation prediction, more exploration about word length will be done at feature engineering part.

We also try other preprocess approaches like word stemming and check spell, but eventually don’t use them since we couldn’t handle the problem of over/under stemming, and check spell is time consuming.

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

**4.2 Feature Engineering**

Although preprocess help eliminate some unhelpful words, the datasets still contain lots of noises words, like “nan”, “like”. Also, there are plenty of words frequently show up in development data but never appear in the training due to the characteristic of tweets that people are more likely to misspell and concatenate words together, like “dandedong” (188 times), “perthfest” (38 times). Therefore, we implement and test few feature engineering strategies to help better address these problems.

The first one is provided from COMP30027, we select top (10, 50, 100) related words of each city based on the Mutual Information and Chi-Square, so that noise words could be filtered out and we just simply ignore unseen words.

Like MI, Konstantinos [5] used a heuristic probability base approach Word Locality Heuristic (WLH) to determine words that are more likely to be “useful” in prediction, we also use this as our second strategy.

Lastly, we adopt the typical TF-IDF text feature extraction method and combine it with other feature engineering methods. To eliminate noises words, we simply assume the train dataset is general enough to obtain the probability of a class given a word (P(c|w)), the words have maximum P(c|w) higher than threshold t is kept, also we eliminated words with length smaller than l or total count is less than c. To deal with unseen words, one method is replacing with most similar words in training set calculated from Word2Vec. Another one is breaking all words down into substring of length s, since most unseen words could be found in training set if we just slightly alter/insert/delete few characters or concatenate words together. Finally, we consider all tweets from the same city as one document and merge all tweets together to better use TF-IDF features.

公式1：P(C|W) =

**4.3 Classifier**

We consider majority-class and stratified Dummy classifier as our baseline. According to previous work, we run several comparative experiment on variant classifiers that are easy to interpret, such as linear Support Vector Machine [5], Bernoulli Naïve Bayes, Multinomial Naïve Bayes [3, 5, 8] and Logistic Regression [5], and the combination (voting and stacking ensemble) of them, where voting ensemble integrate MNB, SVM and Logistic Regression and use hard voting, stacking ensemble put MNB and BNB as lower layer and use Logistic Regression as higher layer.

**5. Experiment Results**

**5.1 Evaluation Metrics**

Here in order to measure the effectiveness of different feature engineering strategies and classifiers, we use Accuracy (ACC), macro-averaging precision, recall and f1 score since the classes are distributed equally, also include predict label distribution, cities are more likely to predict correct/incorrect, and city has the biggest number of instances being classified as another city, which are all obtained from confusion matrix, and finally, the execution time.

**5.2 Results**

Table 5.1 and 5.2 show the results of different feature engineering methods and classifiers, the highest value of each column is represented with bold font. Although we try various combination of hyper-parameters of TF-IDF features engineering, due to page limitation we only list those are representative here. The best classifier is determined by their ACC and Time together, that is the classifier the has the highest accuracy with as low execution time as possible.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Classifier | Best Classifier | ACC | Precision | Recall | Fscore | Time |
| Baseline | Dummy(maj) | 0.25 | 0.0625 | 0.25 | 0.1 | 0.126616 |
| 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  t=0 + s=not + l=0 + c=0 | Stack | 0.343873 | 0.346076 | 0.343873 | 0.342224 | 2.298197 |
| TF-IDF  t=0.5 + s=6 + l=3+ c=10  merge tweets from same city into one document + replace unseen with word2sec | MNB | 0.339452 | 0.384120 | 0.339452 | 0.317352 | 0.111144 |
| TF-IDF  t=0 + s=6 + l=0 + c=0 | Voting Ensemble | 0.353012 | 0.354618 | 0.353012 | **0.352226** | 58.89453 |
| TF-IDF  t=0.5 + s=6 + l=3 + c=0  merge tweets from same city into one document | MNB | **0.356120** | 0.371116 | **0.356120** | 0.350733 | 0.153969 |

Table 5.1 evaluation on the score

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classifier | Best Classifier | Class Distribution | City has most correct prediction | City has least correct prediction | Most frequent error |
| Dummy(majority) | Dummy(maj) | 100% Bris | Bris | Melb, Syd, Perth | Mel/Syd/Perth -> Bris |
| Dummy(stratified) | Dummy(strat) | 24.88% Melb  24.59% Syd  24.97% Bris  25.55% Perth | Melb | Syd | Bris -> Perth |
| MI (top10) | MNB | 3.23% Melb  1.77% Syd  93.34% Bris  1.66% Perth | Bris | Syd | Syd -> Bris |
| MI (top50) | MNB | 4.10% Melb  2.84% Syd  89.49% Bris  3.56% Perth | Bris | Syd | Syd -> Bris |
| MI (top100) | MNB | 0.69% Melb  0.21% Syd  98.55% Bris  0.54% Perth | Bris | Perth | Melb -> Bris |
| WLH (top 10%) | MNB | 4.40% Melb  6.33% Syd  85.56% Bris  3.71% Perth | Bris | Syd | Syd -> Bris |
| WLH (top 30%) | Voting Ensemble | 1.83% Melb  93.22% Syd  2.20% Bris  2.75% Perth | Syd | Melb | Melb -> Syd |
| WLH (top 50%) | MNB | 4.90% Melb  3.39% Syd  86.04% Bris  5.12% Perth | Bris | Syd | Syd -> Bris |
| TF-IDF  t=0 + s=not + l=0 + c=0 | Stack | 23.57% Melb  32.99% Syd  20.87% Bris  22.57% Perth | Syd | Bris | Bris -> Syd |
| TF-IDF  t=0.5 + s=6 + l=3+ c=10  merge tweets from same city into one document + replace unseen with word2sec | MNB | 12.49% Melb  13.01% Syd  60.45% Bris  14.05% Perth | Bris | Syd | Syd -> Bris |
| TF-IDF  t=0 + s=6 + l=0 + c=0 | Voting  Ensemble | 24.60% Melb  26.91% Syd  28.55% Bris  19.95% Perth | Bris | Perth | Syd -> Bris |
| TF-IDF  t=0.5 + s=6 + l=3 + c=0  merge tweets from same city into one document | MNB | 17.92% Melb  20.33% Syd  43.32% Bris  18.41% Perth | Bris | Perth | Perth -> Bris |

Table 5.2 evaluation on the result data

**5.4 Interpretation and Error Analysis**

As presented in table 5.1 and 5.2, MNB and Voting Ensemble classifier combing MNB, LinearSVM and Logistic Regression perform more effectively in terms of the evaluation criteria.

Probability-based model like MNB, BNB performs better when unrelated words are filtered out, since the appearances of irrelevant words could largely decrease the posteriors probabilities and results in a low prediction score. After leaving out these words, NB predict an instance to a label simply based on how many informative words show in it and how pervasive are these words. For instance, 84.19% of “melbourne” are from Melbourne and this word is strong indicator of Melbourne for NB. Also, the smoothing method being chosen here is Laplace smoothing with alpha 0.01, means that we consider new values less important than the existing value since most of them are just interferences, like “pleaseee”.

As for classifiers with parameter assigned to each attribute, like SVM, Logistic Regression, the influence of unrelated words is being reduced by lower associated parameter, so that even without filtering, they could still produce nearly equal distributed good results combing with MNB. LinearSVM will create hyper plane could divide as more informative words (attributes with high-value parameter) into groups as it could and less focus on how irrelevant words (words with low-value parameter) is being divided, . Here we adopted penalty C as 0.95 to ignore outlier instances and get more general model, like Melbourne tweets contains “sydney”. Similarly, logistic regression. We used lbfgs solver with l2 penalty that suitable for multi-class problems and handles multinomial loss and change the inverse of regularization strength to 0.95 to allow few mistakes.

However, irrespective what feature engineering methods being chosen, the main error that all classifiers encountered is that the predict results distribution is highly bias towards one label due to the lack of information, as some tweets are just lyric or advertisement that are inherently not predictable. 15 – 20% instances are empty after filtering (even more when using MI or WLH), and some others contains only unseen words, therefore, classifier couldn’t do anything more than guessing majority. And if we don’t eliminate those words, common words with little bias towards specific city in the training set, like “nan” prefer Melbourne, “good” prefer Perth, will also lead to a higher probability of misclassifying. Parameter

Another unavoidable error is the wrong indicator, like “melbourne” in tweets from Sydney, NB would always predict towards Melbourne based on the posterior probabilities it learned unless there are more other strong indicators of other cities in this tweet.

**6. Conclusions**

Table 5.1 and

Reference:

[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-StockMarketPredictionUsingTwitterSentimentAnalysis. 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 (IUI’19)*. 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.