Machine Learning (Natural Language Processing)
Predicting Location of Tweets

Akira Wang and Callum Holmes
(Dated: May 24, 2019)

Twitter is an open online social network platform where users have access to media through a type of message denoted as tweets.

The aim of this project is to successfully identify the location from which a tweet message was tweeted, as one of four Australian cities (Brisbane, Perth, Melbourne and Sydney). This problem is a subset of **geo-tagging**, a wider problem set relevant to spatial and social analytics that asks whether some form of geographic label can be predicted for data (such as tweets). The usefulness of such predictions can range from natural disaster response to targeted advertising.

Our investigations have shown that an optimal feature space balancing the total number of features and minimising the number of redundant features, yields an increase in accuracy that suggests a plausible ability for classifying tweet locations. This was seen to be credible after our baseline classifier was able to make the top percentile for Kaggle submissions.

I. PREDICTIVE MODELLING

The multi-class classification problem prevented the use of typical binary classifiers such as Logistic Regression or Support Vectors Machine (SVM) without extending to multi-dimensional approaches. Thus, any used classification algorithms needed extension to multi-dimensional counterparts (as has been successfully implemented in our research).

With this in mind, we also need take into account the possible time and space complexities we may face due to the algorithmic complexities of the above.

As such, simple classifiers such as Naive Bayes were our primary benchmark, due to its robustness under the conditional independence assumption and its moderate scalability (grows linearly with number of instances).

For the remaining classifiers, we will experiment with Support Vector Machines (SVM), Logistic Regression (LR), and a Stacked Ensemble Learner (utilising a meta-classifier approach). Further experimentation suggested that there was plausible benefit in using the NB model in our meta-classifier.

A. Generative Multinomial Naive Bayes (NB)

Generative models are based on the estimation of joint probability distribution of observing a word vector in the presence of a class. This means that our features should correspond to integer frequency counts, but may still work with fractional values.

We will use be using a Generative Multinomial Naive Bayes since it incorporates a class prior, allowing it to classify an instance in the absence of any features shared with the training data. This makes it ideal for problems such as text classification.

B. One vs Rest Support Vector Machine (SVM)

Support vector machines work by maximizing the margin which separates examples from the hyperplane. Although the base SVM classifier only supports binary classes, we can use a multi-class by transforming the data using a linear kernel with a One vs Rest approach.

Our feature space and instance count is notably large; however this coupled with the binary values that each feature should allow the SVM to differentiate the classes quite well. On this basis, SVM should suit our tweet classification problem.

C. Multi-Class Logistic Regression (LR)

Logistic regression uses a logistic function to estimate the probabilities of instances associating with binary variables. For this task, we will use a multinomial variant of logistic regression to allow for multi-class predictions.

As explained in the "Author Classification" research preferred by this paper, the multinomial variant of logistic regression should perform well in text classification problems.

D. Stacked Ensemble Learner (Ensemble)

Stacked Ensemble Learners work by training a meta-classifier to combine the predictions of several base models. This works by training each base model using the train data set, from which the meta-classifier operates on to make a final prediction.

Our implementation of the Stacked Ensemble System will use three base models: Multinomial Naive Bayes, One vs Rest Support Vector Machine, and Random Forest (using gini as criterion). This will hopefully give a wide variety of varying biases and variances, which our meta-classifier can use to predict new instances, based on the probability of each base classifier's predictions.

II. INITIAL RESULTS

A. Data

The data sets provided have a feature space derived from the top-$n, n \in \{10, 50, 100\}$ number of words according to Mutual Information and Chi-Square. Each data set has a uniform distribution of labels corresponding to the classes (Sydney, Melbourne, Brisbane, Perth), and originate from different Twitter at different timepoints.

Our classifiers will train on the train set, and use the development set to estimate classifier accuracy - the equivalent of a 75-25 holdout strategy.

B. Results and Evaluation

Classifier	Top-10	Top-50	Top-100
NB	29.49%	30.14%	30.79%
SVM	29.49%	29.79%	30.39%
LR	29.49%	30.05%	30.77%
Ensemble	29.51%	29.83%	30.42%

TABLE I. Initial Classification Accuracy

Our benchmark classifier NB gives us an accuracy of about 30.14% across the provided data sets (Table 1), which indicate this task at hand is quite difficult.

Although Randomised Search for hyper-parameter tuning was executed, the performance was unjustifiable given the time consuming process. In addition, the hyper-parameters resulted in an over-fit model, where we saw negligible improvements on the dev set, but a decrease in performance on the test set on Kaggle.

As expected from real world text based predictions for geological location, an accuracy as high as 80% based off these tweets seems unrealistic, or even impossible.

Consider this very possible example tweet:

"Melbourne was a fantastic place! I absolutely loved it :)".

But, the tweet was tweeted by a tourist *from* Brisbane **after** they had returned.

With our current bag-of-words feature space, only the feature *"melbourne"* would be recognised in feature set, and the classifier would classify this instance as a tweet originating from Melbourne. Yet from a human standpoint, the phrases *"was a fantastic place!"* and *"absolutely loved it"*, would have implied that someone had indeed just visited Melbourne. On the contrary, even with the added context, a human would still never know **where** the tweet was tweeted from. The end result is a random guess, albeit a slightly higher probability of a correct location since Melbourne could be ruled out.

In essence, even a human classifying this tweet would find it troublesome. In the context of a learning approach, only classifiers that employ negative sampling approaches (such as Neural Networks) could possibly extract underlying information from this, without hurting accuracy.

It is because of several tweets like this, that a performance of 30% now looks much more reasonable.

C. Initial Error Analysis

If we take a closer look at top-10 data set, it becomes apparent that several development instances are filled with zeros, implying that these instances do not contain any of our features.

Consider these three features of the top-10 data set:

$$\{\text{“}uu\text{”}, \text{“}uuu\text{”}, \text{“}uuuu\text{”}\}$$

Although they may have high chi-square or mutual information, the features may only exist in one tweet within the train set but will *certainly* identify a location. Yet,it becomes apparent that the feature becomes redundant when predicting new instances, rendering it useless as a feature.

As a result, our classifiers are forced to use a 'tie-breaker' for these instances, performing the same as a uniform dummy classifier (i.e. classifier that makes predictions at random). This is hypothesised to be the main cause of such low performance when predicting.

Compare this to the top-50 and top-100 data sets. Our classifier performances increase uniformly at a small rate. The results from the initial data set give us an intuitive explanation - the more features we have in our training model, the better performance we can expect. This is because there should be less tweets containing no features, giving more predictive power to our classifiers. Taking the data sets into account (since they originate from different time periods), by combining both development and train instances, we should be able to find a verbose feature space.

III. FEATURE SELECTION

Rather than purely relying on a numerical metric for feature selection, we decided to also take into account the nature of this task, hence NLP techniques such as Stemming and Lemmatization was considered (refer to the example in the Initial Error Analysis). In the final experimental design, lemmatization was deemed to be the more preferable method.

A. Feature Selection and Preprocessing Steps

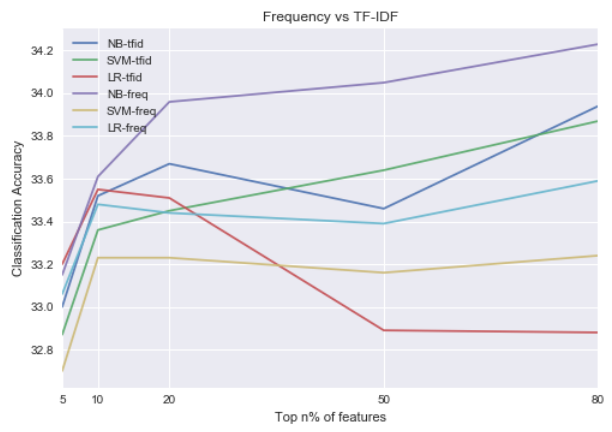
1. The features were stripped of all punctuation, Unicode emoticons (in the form "\\utXXX"), non-English characters, numbers, and English stopwords (provided by the 'nltk' library).
2. The cleaned body text was tokenized (using 'nltk.TweetTokenizer').
3. Tokens were then *lemmatized* (using 'nltk.lemmatize').
4. Words **with at least** 2 occurrences were vectorized in a binary format (all non-zero counts become zero).
    - This removes words that are rare in the data set and improves robustness. Singletons cannot be reliably used for classification as the conditional class distribution of instances with singletons will be completely biased towards a single class - and so any occurrences in test sets will yield biased classifications on these rare words.
    - Allows us to perform proper feature engineering and selection for words *in the presence* of a class.
    - Also significantly reduces our feature space (184,674 unique words, and after lemmatization to about 39,000 - a substantial 79% decrease of nuisance features).
5. Using chi-square and mutual information, features were engineered and selected.

- For chi-square, we conducted a False-Positive Rate (FPR) test and took values at a significance level of $\alpha = 0.2$. This was conducted through a experimental design that aimed to minimize the occurrences of False-Positive.
- For mutual information, we took the top 50% of features. We determined that this was the most ideal percentile after several iterations.
- Finally, the unique set of both features was taken to be our final feature space. This ensured that we were able to retain all useful features.

6. The data was re-vectorized using both frequency and TF-IDF for the new feature space.

B. Performances of Vectorization Methods

Following feature selection, the vectorization method had to be accounted for. The techniques considered were Frequency Counts and Term Frequency-Inverse Document Frequency (TF-IDF).



- NB performs quite well with TF-IDF, but arguably performs better with frequency counts (due to the Multinomial distribution). It may be the case that Naive Bayes works well with frequency counts due to its conditional independence assumption - whereas TF-DF is a normalization and thus may undervalue the higher frequency words than a raw frequency count.
- SVM performs significantly better with TF-IDF, most likely since TF-IDF normalized rarer words which may play a role in distinguishing specific tweets.
- Although LR performs well with a small n% of features, it seems to become inconsistent as more features are added. Therefore, it was opted to use frequency counts since it was dependent on the Multinomial distribution like NB.

After examining the intermediate performances of each vectorization technique, the final data set will utilise:

- frequency counts for NB and LR, an intuitive choice that complies with the properties of the Multinomial distribution.
- TF-IDF for SVM, which allows for higher weightings for rarer words.
- TF-IDF for stacked learners due to the probabilistic counts, which is amplified when taking the TF-IDF of a document.

IV. FINAL RESULTS AND EVALUATION

Classifier	Accuracy	Precision	Recall
Naive Bayes	34.00%	34%	35%
Support Vector Machine	33.40%	33%	33%
Logistic Regression	33.35%	33%	33%
Stacked Ensemble	32.87%	33%	33%

TABLE II. Final Results (Micro Averaging)

From the final results, we see that all our classifiers benefited from our feature and engineering and selection on the raw data set. Our initial feature space used a much smaller range of features and still saw a minimum increase of 2%. With our final feature selection, we can see that our initial proposal that performing feature selection at a greater level would yield much better performance.

Although we had expected our stacked ensemble to perform the best, it was surprising to witness the overall performance fail to exceed the simple benchmark classifier - Multinomial Naive Bayes. Hypothetically, it is expected that meta-classifiers, including the stacked ensemble should yield reduced variance (and consequently, more precise predictions) when the base classifiers are independent of each other.

Naive Bayes relies on the conditional independence assumption that features are independent and have no interaction - the consequence being that a feature will only help influence the classification if a value occurs to a high frequency, with respect to a class.

It may be the case that given the vectorization method used was raw Frequency counts, the Naive Bayes classifier was overfitting words with high frequencies; while it did yield stronger results, greater attempts to regularize the classifier may be necessary in future investigations to rule out such possibilities.

Furthermore, the data used had a significantly large feature space; this is where the multinomial distribution excels. If time and space complexity is also taken into consideration, the benchmark Naive Bayes appears stronger still, with a theoretical time complexity (with equal number of instances) of $\mathcal{O}((CD+C))$ compared to SVM $\mathcal{O}(C^2D+C^2)$ (C number of classes and D number of features).

Comparing to the results of the Kaggle competition, our over 35% accuracy rate placed our classification process in the top percentiles. This suggests that despite the high error rate encountered by our approach, other groups' effectively independent processes in the worst case yielded no better a result - suggesting the errors encountered were largely systemic and attributable to the high variance inherent in the data itself (as explored in **Initial Error Analysis**).

V. CONCLUSION

It was found that after additional pre-processing, lemmatization and feature selection using chi-square scoring, and Frequency Count vectorization, up to 4% gain in overall accuracy was observed on our classifiers. Our results indicated that a Multinomial Naive Bayes classifier still performed better than the other classifiers discussed (which is also discussed in this study by Rennie), albeit by a margin of 1%.

Since the train / dev set were an example of a 75-25 holdout strategy, future considerations include performing a k-fold cross-validation of a re-sampled train set to yield more promising results. This would be due to the data sets originating from different years, and re-sampling the train set would help increase the verbosity of the feature space. Due to the time constraints presented for this project, we were unable to complete this and remained with the holdout strategy.

A pre-trained neural network created by Stanford University researchers dubbed "glove" is specifically designed for classification on tweets (roughly 7 billion different tweets were used to train it). Future attempts at this problem should take a look into a deep learning, or even a network-based methods for improved results.

VI. REFERENCES AND ACKNOWLEDGEMENTS

Bird, Steven, Ewan Klein and Edward Loper. *Natural Language Processing with Python.* Sebastopol: O'Reilly Media, 2009.

Geron, Aurelien *Hands-on Machine Learning with Scikit-Learn and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems.* O'Reilly Media, 2017.

Lane, Hobson. *Natural Language Processing in Action.* Manning Publications Company, 2018.

Bengfort, Benjamin, Rebecca Bilbro, and Tony Ojeda. *Applied Text Analysis with Python: Enabling Language-aware Data Products with Machine Learning.* Sebastopol, CA: O'Reilly Media, 2018.

Madigan, David. *"Bayesian Multinomial Logistic Regression for Author Identification."* AIP Conference Proceedings, 2005. doi:10.1063/1.2149832.

Joachims, Thorsten. *"A Statistical Learning Learning Model of Text Classification for Support Vector Machines."* Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR 01, 2001. doi:10.1145/383952.383974.

Yoo, Jong-Yeol, and Dongmin Yang. *"Classification Scheme of Unstructured Text Document Using TF-IDF and Naive Bayes Classifier."* 2015. doi:10.14257/astl.2015.111.50.

Manikandan, J., and B. Venkataramani. *"Diminishing Learning Based SVM Classifier with Non-linear Kernels."* 2008 International Conference on Electronic Design, 2008. doi:10.1109/iced.2008.4786724.

Professor Tim Baldwin's Publication List.