Thanaboon Muangwong 1049393
Saket Khandelwal 1041999
Aali Alqarni 1002741

1 Abstract

This report tries to cover our experiments for Tweet authorship attribution system using different feature representation schemes and Machine Learning models to identify tweets' authors. Also, we illustrate the data preprocessing and feature engineering, experiment results, challenges and finally, critical analysis.

2 Introduction

Tweet authorship attribution systems face obvious challenges that lead to less accuracy and misclassification. Firstly, the corpus of tweets is short texts which are considered much smaller than the famous author's publications. In addition, short texts may contain informal, slang and icons which make them noisy. Another challenge, tweets of a single author may represent several thoughts rather than provide single patterns of opinions. As a result, it makes it difficult to classify microblogs based on authors' ideas. In this report we describe our approaches in feature engineering, Model selection and results analysis on our pipeline for tweet authorship attribution.

3 Data Preprocessing and Feature Engineering

3.1 Dataset and EDA

We were provided with a training dataset consisting of roughly 328k tweets by 9k users. We were also provided an unlabeled dataset with over 35k tweets limiting to the same users from the training set. Upon further analysis we found that for the 9k the distribution for number of tweets per user was highly imbalanced as you can observe in Figure 1 below. We also observed that a lot of text were short.

```
count    9293.000000
mean       32.131820
std        25.313505
min         1.000000
25%        17.000000
50%        26.000000
75%        37.000000
max       262.000000
Name: User, dtype: float64
```

Figure 1: Tweets per user distribution

3.2 Data Preprocessing

We first processed the dataset by converting the text file to a dataframe consisting of user-id and the corresponding tweet as separate columns. We filtered out all data which where retweets and duplicates. We also cleaned the tweets by removing links, decoding HTML tags and substituting accents. We removed all words starting with "@". The processed text was converted to lowercase followed by tokenization using NLTK's *TweetTokenizer* function, which preserves the hashtags as we expect these to be relevant information in identifying users. Stop words were removed by comparing to the NLTK's *stopwords* set. Lemmatization was then performed on the processed text using NLTK's *WordNetLemmatizer* to reduce each token to its base lemma. In order to lemmatize we need to provide the token's part-of-speech, first we used its verb part-of-speech if it doesn't exist then the noun was used.

3.3 Feature Selection and Extraction

Utilizing the tokenized and processed tweet data, we extracted syntactic, lexical, and semantic features to feed our classification models.

3.3.1 TF-IDF method

We used the term frequency-inverse document frequency (TF-IDF) method. This weight is a statistical measure used to evaluate how relevant a word is to a document (tweet). It measures how frequently a term occurs in a tweet and how relevant it is by weighing down too frequent terms and scaling up rare ones.

To produce this feature set we used the Sk-learn's *TfidfVectorizer* function. We created three tf-idf feature sets. Two containing character and word n-grams ranging from 1-grams to 4-grams, the third capturing characters unique to tweets language, for example, hashtags, emoticons etc. For each of the feature sets, we limited the most frequent words to 60000 words for each set because considering the whole corpus can be computationally extensive. We chose to stick with 4-grams for the upper bound range because of the length of tweets mostly being short.

3.2 Part of Speech and Sentimental analysis

We used the part-of-speech (POS) tags as our features to further enhance the feature set by syntactical analysis. We used NLTK's part-of-speech tagger to tag each token with its respective tag. Our POS feature set consisted of the frequency of each POS tag within the tweet. We also further added sentiment analysis in which the tool used was VADER (Valence Aware Dictionary and sEntiment Reasoner) which is a lexicon and rule-based sentiment analysis tool which has been specifically attuned to sentiments expressed in social media [1]. The tool provides the intensity of the tweet being positive, negative, and neutral with scores ranging from 0 to 1 and a composite score ranging from -1 to 1.

These scores were used to create the sentiment feature set and we think it can help us further distinguish between users. Both of these feature sets were constructed on the text before the stopword removal and lemmatization process.

3.3.3 Sampling

Due to our dataset being imbalanced which reflects unequal distribution of classes within dataset. We chose to downsample the majority class or in this case users with tweets higher than a certain threshold. So we randomly selected all tweets and limiting to a maximum of a certain threshold tweets or take their max for each user not reaching the threshold.

3.3.4 Union and Feature sets combination

We created a union of all feature sets explained above and normalized by their unit norm because of the differences in the ranges of the features. We also tried different union sets of features depending on the model which is discussed below.

4 Learners and Model Selection

4.1 Naive Bayes

Due to its simplicity and its method of learning a priori and conditional probabilities results in fast training and is often used as the baseline for many classification tasks. For this task, we choose two types of naive Bayes classifiers, the Multinomial Naive Bayes (MNB) and the Complement Naive Bayes (CNB). Naive Bayes makes strong conditional independence assumptions between features. MNB is just a specific instance of the Naive Bayes classifier which uses a multinomial distribution for each of the features and is used for multiclass classification. CNB is also another instance of the Naive Bayes but the estimation of probabilities for each class are based on the complement, CNB is known to work well with an imbalanced dataset (which is the case with our dataset) and tackles the severe assumptions made by MNB [2]. We used scikit-learn to implement both of these models.

4.2 Support Vector Machines

Support Vector Machines (SVM) can perform multiclass classification using a one-vs-all approach, by using the best fit hyperplane which maximized the “margin”, or the largest separation between the classes. In a multiclass problem, the optimization is solved for all the classes, most of the differences between the different variations of SVM is the choices of the kernel. For this task, we choose Linear SVC (Support Vector Classification) because it has more flexibility in the choice of penalties, loss function and scales better to a large amount of dataset and we expect it to perform better on sparse data. Linear SVC basically uses a linear kernel for the optimization with a squared hinge loss function and we only have to adjust the regularization parameter. Another key difference is that Linear SVC penalizes the intercept as well. Due to our data and number of classes being large we chose Linear SVC as it is optimized for a linear case which converges quicker on large datasets. We implemented the model using scikit-learn.

4.3 Logistic Regression

Logistic regression is one of the most fundamental and widely used ML algorithms. Logistic regression is not a regression algorithm but a probabilistic classification model. It is an advanced Linear regression technique used for classifying both linear and non-linear data. It is being used widely to model data with binary response variables. In this case, we have multiple classes which we are required to use multinomial logistic regression classifier since we have around 9000 classes. Multinomial Logistic Regression uses one vs all technique which is used to fit one classifier per class. For every class, the class is fitting against all the other classes. We used scikit-learn to implement Multinomial Logistic Regression model.

4.4 Model Selection

We tried various combinations of features that we generated and feed it to our model. For both the Naive Bayes models we chose to use the normalized TF-IDF features and POS tags combinations, as sentiments contained negative scores we discarded them because the models expect positive values. We tried combinations of all feature sets for Linear SVC as all of them are compatible, TF-IDF and POS frequency counts are linearly separable which make them a suitable choice. The choice of using a linear kernel is also because the data has a higher dimension because of the size. For Multinomial Logistic regression, we chose to use TF-IDF features except for the special characters feature set of TF-IDF.

5 Critical Analysis

5.1 Results and Critical Analysis

We discuss the comparison between the models from section 4 in terms of classification accuracy on a validation set and as well as the test set and provide a thorough analysis. We also discuss how the model we preferred performs. The Validation set was a split of 25% from the training set and the test set was provided for the kaggle submission.

(1)POS - Part of Speech (2)TF-IDF(word and character) (3)TF-IDF(special characters) (4)Sentiment Score (5) Sampling

		Multinomial NB		Complement NB		Linear SVC		Logistic Regression Except (3) feature	
Feature	# Features	Val Acc %	Test Acc %	Val Acc %	Test Acc %	Val Acc %	Test Acc %	Val Acc %	Test Acc %
2+3	130,871	11.2	10.3	18.6	17.4	31.9	**32.2**	28.95	**27.403**
2+3+5	130,871	20.2	**15.3**	20.1	**18.3**	29.3	29.6	-	-
1+2+3	130,916	11.1	10.7	18.1	17.1	30.9	31.31	-	-
1+2+3+5	130,916	-	-	-	-	31.2	31.04	-	
1+2+3+4	130,920	-	-	-	-	32.5	32.14	-	-
All features	130,920	-	-	-	-	31.6	31.5	-	-

Table 1: Accuracy results of models with different feature combinations tested on validation and test set.

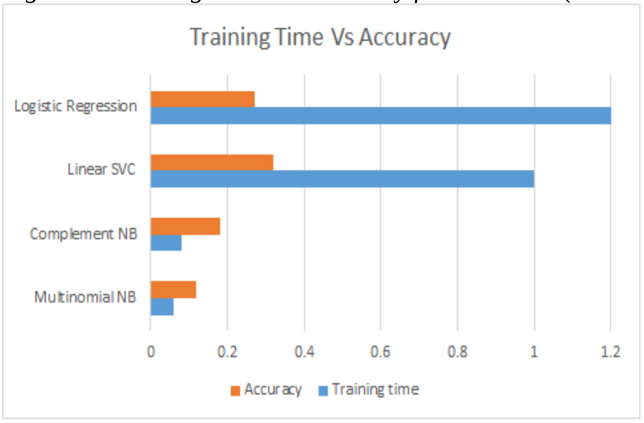
From above table, Linear SVC outperform both Naive Bayes model and Logistic Regression. The best accuracy that we achieved is using the combination of feature sets (2) and (3) in Linear SVC. Due to computational time limitations of SVC the ngram-range we came down to explained in our feature selection section is by using Multinomial Naive Bayes as the baseline and we then used that range for all our models. By observing the accuracy results for the Naive Bayes we can see that both MNB and Complement NB work better when sampling is performed on data as expected. We can also observe that MNB does not generalize for many users as it is bias towards denser classes. Complement NB as suspected does perform better on imbalanced dataset compared to that of MNB because of the reasons explained in section 3. The addition of POS tags feature set did not significantly change the accuracy as this may be because of the vast difference in values and size between the features. In contrast, Logistic Regression was performing better than Naive Bayes models but not better than Linear SVC.

Training the Linear SVC classifier on identical features from the Naive Bayes models yields better accuracy for all the feature combinations. A regularization parameter of 1 and a max iteration value of 10000 was empirically set for the model. It is also noticeable that sampling does not actually help as this maybe because the sampling method may remove important features which could be a problem. Adding both sentiments and POS tags results in the similar accuracy which means that the features are not really significant as the TF-IDF features are highly sparse and thus dominate. We can also see that due to the regularization and penalization the linear SVC classifier generalizes better than the other models. That’s the reason we chose Linear SVC as our final model.

5.2 Problems

The problems we encounter in this project is time complexity of training our model and imbalanced class. Naive Bayes took least time in training but in logistic regression, the training time took 3 days to finish and the accuracy wasn’t better than Linear SVC which took about 3 to 4 hours to finish. Imbalance class is also a problem in our classification because some users had only 1 or 2 tweets while some users tweet 200 or more. This makes our model bias toward the users that have more tweets. The training time vs accuracy can be observed in the figure below.

Figure 2: Training Time VS Accuracy for the model (Training time normalized by 3 hours)



6 Reference

[1] Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

[2] Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. 2003. Tackling the poor assumptions of naive bayes text classifiers. In Proceedings of the Twentieth International Conference on International Conference on Machine Learning (ICML'03), Tom Fawcett and Nina Mishra (Eds.). AAAI Press 616-623.