COMP90051 Project1 Report

Group 68; Kaggle Team Name: ChickenDinner

Team Member: Rongxiao Liu; Xudong Ma; Dong Jia

1. Introduction

Authorship attribute (AA) is to detect the author's unconscious writing habits implicit in the text, which can be expressed via certain quantifiable features, thereby determining the author of the anonymous text. With the advent of social media, people tend to interact using brief text and sometimes use emojis to express their feelings rather than using long text. AA has been widely used in literary works, commodity comments, author identification of spam email and supervision of public sentiment and public opinion in recent years. Koppel and Winter has discovered that it is difficult for AA systems to achieve the same performance for short text compared with long text. Tweets are one of the most popular short text format which has short, restricted lengths and more casual writing requirements. In this work, we focus on short text tweet and try to solve the authorship attribution task on Twitter. In this project, the data set consists of 328932 tweets from 9297 users.

2. Methods

2.1 Feature Selection

When observing the content of tweets in training set, we notice that users follow similar writing "patterns" while they may not always express the same meaning. Overall, the writing style we noticed have two categories as follows:

(1) structured components in twitter: Considering that there are several components with fixed format that may appear in every tweet, we treat them as the most important features. They are: 1) retweet label: "RT". We notice that the same retweeted tweet appears not only once; 2) At Someone: "@handle"; 3) hashtag: "#xxx". It represents the topics one user may focus on; 4) URL link: "http://xxx.xxx.xxx/". Furthermore, we find that one user may post URL links with the same domain name and different path. Thus, domain name is the most important "pattern" in URL.

(2) personal writing "patterns": Apart from these fixed structured components, some other "patterns" can also represent users' writing habits show their personality: 1) repeated letters: "whatttttt", "heeeeello"; 2) repeated punctuations: "??????", "....."; 3) emoji characters: ":-)", ":D"; 4) language (English tweet or not)

Text Cleaning: To ensure the features we mentioned above be extracted, we first clean the original tweets so that the "patterns" we mention above can be standardized as tokens in sentence: e.g. 1) lowercase text, 2) attach label ("$REPEAT") for repeated characters (if there are more than three continuous identical characters "xxx!!!???..."), filter url link into its domain name ("http://doc.google.com/123" => "doc.google.com"). We also tried lemmatization and to filter stopwords & punctuation but they do not contribute to the accuracy.

Feature Extraction: After cleaning original tweets, we apply bag of word (BOW) and tf-idf to transform the texts to vectors based on CountVectorizer and TfidfVectorizer in sklearn. Then we get a sparse matrix where term frequency or tf-idf of each term represents a feature. Comparing BOW and tf-idf, we find that tf-idf performs better. The reason may be that tf-idf emphasises the contribution of term rarity while BOW only calculates the number of vocabulary words in each tweet.

Furthermore, we also perform character level ngram transformation in tfidf vectorization step so that each term is not a word but a character level ngram substring. It significantly boots the accuracy.

Why not using word/sentence embedding: First, the goal of this task is to predict authorship according to the these writing patterns rather than finding the sentence with the most similar meaning

(e.g. “I am happy~” and “So delighted I am!!!” are probably from different users). Second, on semantic level, considering that vocabulary in tweets is not formal as well and even unique between users (e.g. “hot” and “hoooooot”, “2morrow” and “tomorrow”), it is likely to be inefficient by using word embedding or sentence embedding as feature. We only care about the appearance of the components and “patterns” we defined above so that bag of word and Tfidf are more suitable. Ngram can also effectively capture these “patterns”. Besides, most of the tweets do not strictly follow english grammar so that POS is not suitable to be the feature as well.

Model Selection

In this project, we aim to implement both traditional machine learning model and deep learning model and compare them with each other.

In traditional machine learning part, we use models in Sklearn that can perform multi-class classification, including LogisticRegression and LinearSVC corresponding to logistic regression and SVM with linear kernel respectively. Both of these two models can perform one-vs-rest classification strategy which is suitable to the problem in this project. To test the performance of these two models, we first choose the tweets from the first 100 users and split them as training data and test data (9:1). After training the models using tweets from the first 100 users (around 1/100 data) to and testing their accuracy, we find that linear SVM performs remarkably better than logistic regression (51.10% versus 36.59%). We also tried other models such as Perceptron and RandomForestClassifier. However, they are far from better than linear SVM. As a result, we decide to train the whole data set and make predictions using LinearSVC. Parameter Tuning: After deciding what models to use and get the test results, we also performed Grid Search to find out the best parameters according to test accuracies.

As for deep learning part, We apply pytorch to develop several fully connected neural networks on this problem. “relu” and “softmax” are applied as the activation functions for the hidden and output layer respectively. Basically, we use only one hidden layer including 1000 units and the training dataset is got from the BOW model without data cleaning. Furthermore, we make some changes on the structure of the network and the preprocessing of the training dataset, as well as the parameters of the neural networks. The results are shown at the “Results” part of the report.

Results

SVM: The result is just like what we assumed. Character level Ngram is more efficient to capture the “patterns” we defines (e.g. continuously repeated characters “ ahhhhh!!!!!????? ” and emoji “ :-) ”) while word level unigram treats them as a unique term.

feature selection	**word level tf-idf**	**character level 4-gram + tf-idf**
test accuracy	**31.906%**	**34.305%**

Tab.1 test accuracy of linear SVM

NN: For all the models of this part, the training batch size is set as 1000. Firstly, we train the basic 1 hidden layer fully connected network on the BOW data without data cleaning. The results for training 1,2,3,10,20 and 30 epochs are shown respectively in table 2.

Epochs	1	2	3	10	20	30
test accuracy	17.35%	17.79%	17.03%	16.43%	15.90%	15.03%

Tab.2 Accuracy on the test set for training different epochs of BOW data without cleaning on the 1 layer + 1000 units fully connected network

It is clear from table 2 that the biggest accuracy is 17.79% with training only 2 epochs. Furthermore, it can be concluded that dozens of training epochs are too big for this problem. Therefore, we set the training epoch as 2 for the following experiments.

On the other hand, we try to change the number of hidden layers and the units for each hidden layer respectively. Also, we apply data cleaning and tf-idf model on the training data before training the models. The results are shown in table 3.

Model	1000 units + 2 layers + no data cleaning + BOW	500 units + 1 layer + no data cleaning + BOW	1000 units + 1 layer + data cleaning + BOW	1000 units + 1 layer + data cleaning + tf-idf
test accuracy	13.12%	13.4%	14.96%	17.04%

Tab.3 Accuracy of different models on the test set

The data from table 3 shows that all these changes perform worse than the best case of our first try. To our surprise, even data cleaning or tf-idf, which results in big improvements on the traditional model, cannot make the deep learning model performs better.

Analysis: We can notice that the performance of deep learning model is not as good as traditional machine learning model SVM. We think there are two reasons for this. Firstly, there are too many classes (9297) and this makes the distribution of the deep learning classifier easier to be uniform, which makes it harder to observe that the probability of the right label is much bigger than the other labels. Secondly, we only have 35 training instances for each class on average and this is far away from being sufficient for training a good deep learning model. On the other hand, SVM that implements “One-vs-the-Rest” (OvR) multi-class strategy applies classifier for each class. If there are n classes, there will be n-1 classifiers in LinearSVC, which guaranteed the accuracy. Another problem that affects the accuracy is that the data is not much balanced. The number of tweets from each user ranges from less than 10 to more than 100.

Future Works

In future work, we need to find a way to utilize new features besides tfidf, to increase the accuracy. Moreover, the tf-idf feature matrix is a sparse matrix so that it demands much memory to train and limits our feature selection (4gram instead of 5gram because we do not have enough memory to run matrix from 5gram tfidf). A solution to this problem is dimensionality reduction. Dimensionality reduction algorithms, such as PCA or LDA, can be performed to reduce the dimensionality and guarantee the accuracy. In this condition, more features can be applied in the matrix. As for the deep learning part, we think some advanced networks can be tried in the future, such as convolutional neural network (CNN) and recurrent neural network (RNN).

Conclusion

In this project, we cleaned the tweet in dataset and extracted the features of “patterns” we defined by several vectorizers, among which character level 4-gram tfidf performs best. After that, we compared performance of different models, especially SVM and NN, and we find that “ovr” linear SVM performs better. The reason is about the limitation of the size of dataset.

References

Koppel, M., & Winter, Y. (2014). Determining if two documents are written by the same author. *Journal of the Association for Information Science and Technology*, *65*(1), 178-187.