Authorship Attribution

1. Introduction

In this assignment we used various algorithms for text classification, performing an authorship attribution task on Donald Trump's tweets. Also, we introduced preprocessing approaches and features extractions. In our comprehensive evaluation, we highlighted the contribution of the word embeddings along with handcrafted features such as capital letters, hash-tags, etc. We also demonstrated the performance of the traditional compressed TF-IDF, i.e., perform SVD on TF-IDF vectors to cope with the sparsity.

2. Installation and Requirements

The program runs on Python 3.6. All the models, the functions and the preprocess data need the required following packages. Some packages are for pre-training models and visualizing thus are not mandatory therefore their code will be under comments in the program:

1) Torch – neural network package
2) scikit-learn – sklearn, machine learning packages
3) numpy - scientific computing package
4) nltk – natural language toolkit
5) re – regex package
6) Matplotlib – visualization – comment
7) Seaborn – visualization – comment
8) Genism - natural language toolkit - comment (used for train word embedding on dataset)
9) Json – save file dictionary data as json file – comment

3. Dataset

The dataset contain two files represent train and test. The train file contain 3528 tweets records and the test file contain 390 records. Both files have metadata. The train file record contains the following features: tweet id, user handle – usernames account that was used, tweet text, timestamp and the device which the tweet was sent from. And the test file contain user handle, tweet text and timestamp.

3.1. Data Assumptions

The dataset is unlabeled data therefore we needed to assume the labels by using the instructions and by previous works (Mcgill , 2016). First, because we got a test data with only three features (columns), we can only use those columns for the classification model. Second, tweet id is unnecessary for the classification task because its unique values will not separate between authors, therefore, it will be deleted. Third, creating 'labeled' data are by the assumption that Donald Trump will only use android device (Mcgill , 2016) but also use only his account official 'realDonaldTrump'. Fourth, Writers have different ways to express their writing. Hence, by creating features to recognize those ways we can divide between authors. And finally, a timestamp is used to separate the daypart and weekday.

4. Data Preprocessing and Exploration

The following section described the data preprocess supported by the exploration.

4.1. Exploration Raw Data:

First, the user account not used as a feature because most of the accounts are of 'realDonaldTrump' and the labeled was created with the aid of this column. Second, after creating the labels that separate between Donald Trump and his stuff we see that there is a small imbalance data 63:37 separating.

4.2. Exploration Text Length

The following histograms plot display the different distribution of word (Figure 1) and characters (Figure 2) counts between different label. Label 0 is Trump and label 1 is his stuff.

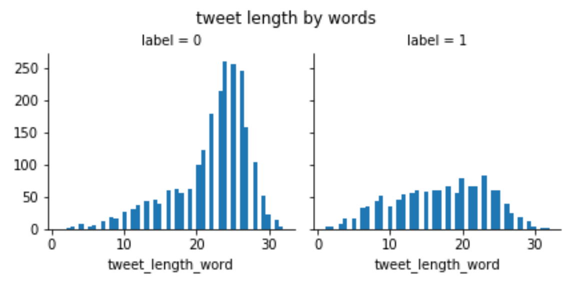

Figure 1: Tweets distribution word length

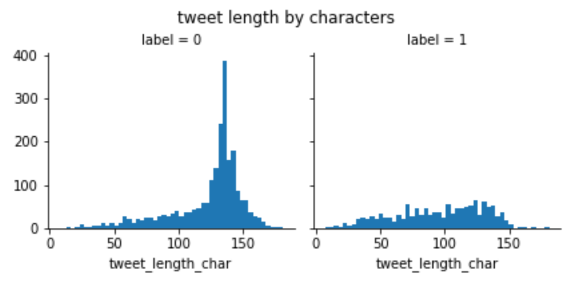

Figure 2: Tweets distribution character length

4.3. Clean and Normalize Text:

We performed a normalization on text for standardize it. This includes the following:

1) Change website url string to str 'website'
2) Multi dots in a row switch to 'dots'
3) Change hashtag sign(#) to str 'hash' and separate text by upper case in hashtags words. example '#ObamacareFail'→ '#Obamacare Fail' → 'hash Obamacare Fail'
4) Lower case of all characters
5) Change at sign(@) sign to str 'atsign'
6) Remove all string punctuation characters
7) Remove extra spaces in a row and leave only one space between tokens
8) Remove single characters if chars is False except 'a' and 'i'
9) Change all numbers that separated by space to 'num'

5. Features Extraction

We created features that helped to separate the authors' styles.

5.1. Timestamp Features:

From the timestamp, we created two features. First, we saw the importance of the hour of the tweet in the day (Figure 3 - left) therefore, we created a category part of the day by the hour – the day was cut to six different pisses from 0:00 to 23:00 every four hours. Second, we can see in, Figure 3 - right, difference amount of tweets between a day in the week so weekday and weekend features were created.

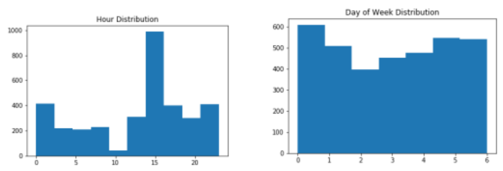

Figure 3: Hour and day of week distribution

5.2. Handcrafted Features:

We created text features for the specific task by using the methods described in the articles (Holmes et al., 1994),(Bozkurt et al., 2007) and also creating our features. The features were created in two steps. In the first step, we used the raw text without any cleaning and normalizing. For capturing person style we counted unique characters. Features were:

Uppercase characters(A-Z), uppercase word, quotation mark(“), At sign(‘@’), hashtags(‘#’), numbers characters, question mark(‘?’), exclamation mark(‘!’), retweet ('RT').

In step two, we normalized and cleaned the text for capturing the words and the characters' styles(Holmes et al., 1994).

1) Counting length of tweet by number of characters
2) Counting length of tweet by number of words
3) Calculating char word ratio - length of average word
4) Counting the use of stop words

5.3. Tf-idf svd features:

As mention in the article (Bozkurt et al., 2007). TF-IDF can create good features for this type of task. But because of its sparsity structure, we used SVD to compress its words features to 100 dimensions and also saving word abstract representation.

5.4. Word2Vec and Tweet Representation:

We trained word2vec (by using gensim package) with 100 dimensions vectors for a word to learn the context between words in the tweets. For each tweet, we average all vectors representations of the words in the tweet. Therefore, for each tweet, we produce 100 dimension average embedding.

5.5. Standardize Features:

Part of the algorithms was needed normalize features for better convergence. Therefore, we use ‘StandardScaler’ from sklearn package for Standardize the data of each feature.

Data	Metrics	Log_Reg	SVM_rbf	SVM_linear	Rand_Forest	Torch
Emb	**F1**	0.771 (± 0.023)	0.74 (± 0.026)	0.754 (± 0.019)	**0.77 (± 0.017)**	0.719 (± 0.021)
	Accuracy	0.852 (± 0.014)	0.841 (± 0.014)	0.846 (± 0.009)	**0.853 (± 0.015)**	0.829 (± 0.016)
Tf-idf-svd	**F1**	0.787 (± 0.047)	0.785 (± 0.034)	0.782 (± 0.035)	**0.793 (± 0.037)**	0.791 (± 0.04)
	Accuracy	0.858 (± 0.033)	0.865 (± 0.02)	0.86 (± 0.024)	**0.872 (± 0.021)**	0.86 (± 0.019)
Emb, Tf-idf-svd	**F1**	0.786 (± 0.054)	0.789 (± 0.022)	0.777 (± 0.048)	**0.791 (± 0.02)**	0.789 (± 0.023)
	Accuracy	0.856 (± 0.037)	0.868 (± 0.014)	0.853 (± 0.033)	**0.868 (± 0.013)**	0.864 (± 0.025)

Table 1: results by features and models

6. Methods

In this section, we present the methods of the input data for the training process and the models used for the classification task.

6.1. Models:

The models uses for the classification task was from the packages sklearn and from pytorch.

Logistic Regression: Use default hyperparameter (sklearn).

Support Vector Classifier: Create two models with different hyperparameters. One with kernel 'rbf' and the other with 'linear' kernel and max iterations 100K - bigger from the default (sklearn).

Feed Forward Neural Network: The network contain 4 layers in size [100,50,10,1] respectively. between the first and the second layers there is a dropout layer with p=0.2. After each linear layers compute a relu function except the final layer that contain sigmoid function(pytorch).

Random Forest: Use default hyperparameters (sklearn).

7. Experiments

In this part we describe the pipeline of the train experiment.

Train all models: We decided that the train data mostly balance therefore we used Accuracy and F1 metrics. Also, we evaluated each model by using cross-validation and compute the mean and the standard deviation of the metrics. Additionally, we created three sets of features for feature selection: (1) handcrafted (2) TF-IDF-SVD (3) embedding. Each model trained on three sets of data combinations: (1) concat with (2), (1) concat (3) and concat all the sets together.

Train the selected model: The selected model 'Random Forest' trained in cross-validation for parameters tuning with three combinations of features. The parameters that tune was (sklearn package random forest): n_estimators, criterion, max_features, max_depth, min_samples_split, min_samples_leaf, Bootstrap. After 50 tunes iterations for each feature selection, we chose the best model with the best features.

8. Results

In this section, we describe the result of the selected models. In every model we used handcrafted features. Therefore, the name of the model is by his other components (Table 1). We can see from the results that in all the experiments Random Forest gets the highest score in all metrics during maintain a small standard deviation compared to the other models. The best model was tune parameters and the selected parameters were - n_estimators: 150, min_samples_split:2,min_samples_leaf:2, max_features: None, max_depth:12, criterion:entropy, bootstrap: True

9. Conclusions

In this study, we demonstrate our ability to recognize authorship by utilizing different models, such as logistic regression, SVM, etc. Additionally, we show that combining handcrafted features with embedding and compress TF-IDF results with better performance. Our results indicate that random forest outperforms other classifiers based on all features.

10. References

Bozkurt, Ilker Nadi, Ozgur Baglioglu, and Erkan Uyar. "Authorship attribution." 2007 22nd international symposium on computer and information sciences. IEEE, 2007.

Holmes, David I. "Authorship attribution." Computers and the Humanities 28.2 (1994): 87-106.

Mcgill, Andrew. "https://www.theatlantic.com/politics/archive/2016/08/donald-trump-twitter-iphone-android/495239/", 2016