**Important Links**

Dataset URL: <http://mib.projects.iit.cnr.it/dataset.html>

Github: <https://github.com/Twitter-Fake/twitter-fake>

Intermediate Data Pre-Processing Output Link+ BERT Model Link: <https://drive.google.com/open?id=1rmlam_0MLmDGiJb6myjbgec2gQ4Jci7pI>

**How To Run**

1. Unzip the data files from the above google drive in the data folder

2. Run the Python Scripts using *python xxx.py*

3. The python compiler expected is 3.7

4. ***The Entire Details on how to Run is present in the code details section of the github readme file ( README.md)***

**Data pre-processing**

*Files: EDA\_train\_data.ipynb , load\_baseline\_train\_data.py, load\_train\_data.py, online\_features.py*

The initial dataset was raw and organized from a collection of datasets (TWT, TFP, E13, etc.). A part of them some were tweeted by human users while the others were fake. The corresponding tweets were also available in these datasets.

First, the data was converted into a single data frame (using *load\_train\_data.py*) with users, profile features, and their corresponding tweets (tweets were concatenated). Based on the source dataset (fake/human), we added a label column to differentiate a fake user from a real user. For example, the TWT\_Fake\_Users dataset contained only fake users, while the TFP dataset contained real users.

Secondly, this dataframe was stored in a csv file (*user\_training\_tweet.csv*). Additionally, the stop words, URLs, and special characters were also removed using regex matching and simple filtering techniques (*online\_features.py*).

Finally, we created a tweet downloader script (*tweet\_downloader.py*) which uses the external Twitter API to fetch genuine tweets. This could be used for testing and for data augmentation.

**Exploratory Data Analysis**

*Files: EDA\_train\_data.ipynb*

To understand the user features and the nature of tweets we explored the training data. We first sampled genuine/fake user tweets and visualized the words using a word cloud representation. We found that the language used was pretty “strong” in the fake tweets.

At this point, we used a simple TF-IDF representation of the tweets for performing K-Means clustering. We found the presence of pure clusters (where all tweets were fake/genuine) and the top-10 words in those clusters gave us valuable insight about the data.

We concluded that we can build a better classifier by using the semantic information in the tweets along with the user’s profile features.

**Feature Engineering**

*Files: online\_features.py*

To extract the additional features from the tweet we attempted the following:

1. Contextual Features (Tweet Replies, Retweets, etc.)
2. Sentiment analysis of tweets with the contextual features (*senti\_strength\_features.ipynb*)
3. Similarity based feature, more profile features (*similarity\_timediff.ipynb*)
4. TF-IDF Representation of the tweets (2000 dictionary words)
5. LDA Topic Modelling (20 Topics to group the tweets)
6. Glove Pre-Trained Encoding (200 Dimensions): These are word embeddings; embeddings of words contained in a sentence were averaged to create a sentence representation
7. BERT Sentence Encoding (768 Dimensions) (*BERT\_Notebook.ipynb*)

**Models**

*Directory: src/model*

Using the above representations, we trained various models. As a baseline, we used a Logistic Regression classifier trained only on user’s profile information (without tweets).

We then created several classifiers, trained them with the 1. user data and 2. with user data and tweet data representations.

The Models (+ Representation + Classifiers) we used are listed below:

|  |  |  |
| --- | --- | --- |
| **Model** | **Classification Accuracy(%)** | **F-1 Score (%)** |
| LR(Baseline) | 81.34 | 86.73 |
| TF-IDF+SVM+WT | 79.77 | 78.96 |
| Profile features + Senti+Sim (XGBoost) | 90.23 | 91.67 |
| XGBoost | 90.8 | 91.6 |
| XGB + LDA + WT | 92.8 | 93.6 |
| DNN | 90.3 | 91.8 |
| **DNN+BERT+WT** | **93.8** | **94.8** |

**WT - With Tweet Data**