Natural Language Processing and Machine Learning for Stylometry
Problem (9)

Mohamed Shawky Zaky AbdelAal Sabae
Section:2,BN:15
mohamed.sabae99@eng-st.cu.edu.eg

***Abstract*— Stylometry is the application of the study of linguistic style, usually to written language. In this work, we discuss the proper methods of using *natural language processing* and *machine learning* for binary classification of authors' writings.**

I. INTRODUCTION

The report contains the discussion of stylometry problem. We discuss the means of dataset collection, feature extraction and language modeling. Moreover, we mention the advantage of the submitted solutions over the other possible approaches. Finally, we review the code structure and usage. The provided code mainly contains two solutions. **First,** *Naive Bayes* classifier with *n-grams* features. **Second,** a simple *neural network* with *word embeddings*.

II. APPROACH

This section contains the discussion of *dataset collection*, *feature extraction* and *language modeling*.

A. Dataset Collection

Dataset is collected from *Kaggle's* **Spooky author identification problem**, as it contains authors of the same era and genre. Other authors are not considered mainly because no available data for multiple authors of the same era. The dataset contains three authors :

- **Edgar Allan Poe** *(EAP)* : 7900 phrases.
- **HP Lovecraft** *(HPL)* : 5635 phrases.
- **Mary Wollstonecraft Shelley** *(MWS)* : 6044 phrases.

B. Feature Extraction

Before feature extraction stage, some text processing is performed on the input phrases. Basically, **tokenization**, **stemming** and **lemmatization** are performed. **stemming** performs better than **lemmatization** in our case.
For feature extraction, five methods are considered :

- **Bag of Words** *(BoW)* : yields the worst performance.
- **n-grams** : offers moderate to decent performance based on the classifier.
- **TF-IDF vectorization** : *n-grams + term frequency - inverse document frequency*, offers decent performance with most classifiers.
- **PCA / Truncated SVD on previous features** : using *principal component analysis* or *truncated singular value decomposition* on our features does not seem to perform well.
- **Word Embeddings** : basically using *GloVe* pretrained embeddings for *neural networks* training.

C. Language Modeling

The submitted solution only offers two methods *Naive Bayes* classifier as a classical language model and a *deep neural network* as a deep learning language model.

- **Classical language modeling : Naive Bayes** is used with *n-grams (count vectorization)* features. This is basically because it yields better results than all the other considered classical approaches. The other classical approaches, considered in this work, are **support vector machines** *(SVM)*, **logistic regression** and **gradient boosting**.
- **Deep learning-based language modeling :** a simple *neural network* is used with *word embeddings*, in order to test the ability of neural networks on our dataset. The network consists of one **bidirectional GRU** layer followed by two **fully-connected** layer with **dropouts**.

III. CODE STRUCTURE

A. Code Files

The submission contains 4 main code files :

- **text_dataset.py :** contains the dataset class that contains the code for building and processing the text dataset. *NLTK* and *sklearn* libraries are used in the implementation.
- **model.py :** contains one class for *Naive Bayes* as linear classifier (*sklearn MultinomialBN* is used) and one class for *the simple neural network* (*PyTorch* is used for network implementation).
- **train.py :** contains the main code for training both *Naive Bayes* and *neural network*.
- **evaluate.py :** contains the main code for evaluating both *Naive Bayes* and *neural network*.

B. Dataset

The folder named *data* has 6 files for train and test data :

- **EAP_train.txt :** *Edgar Allan Poe* train data.
- **EAP_test.txt :** *Edgar Allan Poe* test data.
- **HPL_train.txt :** *HP Lovecraft* train data.
- **HPL_test.txt :** *HP Lovecraft* test data.
- **MWS_train.txt :** *Mary Wollstonecraft Shelley* train data.
- **MWS_test.txt :** *Mary Wollstonecraft Shelley* test data.

C. Miscellaneous

- **config** folder : contains *JSON* config files.
- **models** folder : contains neural network model file *(deep_model.pt)* and linear classifier model file *(nb_model.sav)*.
- **report** folder : contains submission report source and *PDF*.
- **README.md** : contains code installation and usage.
- **requirements.txt** : contains required dependencies.
- **w2v_models** folder : *[REQUIRED]* needs to be created *(follow code usage details)*.

IV. CODE USAGE

The details, mentioned in this section, are also mentioned in *README.md*

A. Dependencies

The following *python* packages and libraries are required for code usage :

- *pytorch*
- *sklearn*
- *nltk*
- *numpy*
- *tqdm*
- *argparse*
- *pickle*

B. Usage

Follow the instructions to run code functionalities :

- **For training linear classifier model :**
  - Create *models* folder.
  - Edit *configs/lc_config.json*.
  - Run *train.py* :
    - *python train.py train-lc*
- **For training neural network model :**
  - Create *models* and *w2v_models* folders.
  - Download *GloVe* https://nlp.stanford.edu/projects/glove/ embeddings into *w2v_models* folder.
  - Edit *configs/nn_config.json*.
  - Run *train.py* :
    - *python train.py train-nn*
- **For inference on linear classifier model :**
  - *python evaluate.py eval-lc –author1 /path/to/author1/text –author2 /path/to/author2/text –model /path/to/model/file*
- **For inference on neural network model :**
  - *python evaluate.py eval-nn –author1 /path/to/author1/text –author2 /path/to/author2/text –model /path/to/model/file –w2v_path /path/to/w2v/model*

V. EXPERIMENTAL RESULTS

VI. CONCLUSION

REFERENCES

[1] Machine Learning Methods for Stylometry Book.

[2] *Kaggle's* **Spooky author identification problem** : https://www.kaggle.com/c/spooky-author-identification