Gender Classfication from Blog Text

Hoang Vo
Department of Computer Science
University of Houston
Houston Texas USA
hoangvod@gmail.com

May 2021

Abstract

Blogs are popular due to the availability of internet and devices. This project will build a model that confirms whether the blog writer is a male or a female. The model is built by using convolution neural network.

1 Introduction

Many people write blogs or articles on websites to express their knowledge, experiences, and feeling. Blog writers could be working people who write blogs and reviews as a way of earning. The wide spread of global internet and modern life lead to the popularity of written blogs, not being constrained on papers and physical forms. There may be many blogs, but we cannot easily depend on names, pretty letters, or pictures to know whether the blogs were written by men or women.

The goal of this project is that with a given set of labelled English blogs written by males and females, I will build a convolution neuron network (CNN) model which can predict the gender of the author of a new blog. Due to the chosen available data set and limited tools, I narrow the current objective within only English language. This model can easily be expanded to the scale of different written languages.

2 Data Set

The chosen data set is from the paper named "Improving Gender Classification of Blog Authors" authored by professor Mukherjee and Liu (Mukherjee and Liu). The data set has two features: blog content and gender of author. It has totally 3232 blogs, and each blog is labeled with the gender of its author. The data set is almost balanced between male blogs and female blogs. Totally 1677 blogs (51.2%) are written by male authors while other 1547 blogs were written by female author. The average length of each blog content is 250 words for male authors and 330 words for female authors.

During testing phrase, I might add in some new data from Kaggle website to solve issue of over-fitting (Schler et al.), but the additional data did not help solve the issue. The additional data could decrease the testing accuracy. In my guess, the data set could contain noises that affects the performance of the model. I did confirm the potential issues of the additional data set, and I will discuss over this issue in later sections. Hence, I returned to using only the first chosen data set for this project.

3 Challenges

The topic of gender classification has many possible difficulties. The first challenge would be the variety of different languages. Blog writers can be American, Vietnamese, Chinese, and so on. Blogs are not always written in English language. If I tried to classify gender of blog writers in any possible written language, then it would require more resources, man power, and time. In this project, I would do Gender Classification in only English language. It also suits the feature of my chosen data set.

The second challenge is the free writing style in blogs. Unlike professional journals and formal documents, blogs do not follow normal rules of grammars and syntax. Sentences and texts in blogs could have odd terms, such as "AAAAAAAA", "btw", "asap", and many other strange words. Blog writers may know English, but they may not native English speakers. Blog writers have different cultures and experiences. A businessman will not speak and write as a doctor does. Those factors create noises in the data set.

The third challenge is the limit of resources. I will apply word2vec and Spacy as pre-trained models to generate weight embeddings during training phrase. The pre-trained models do not know all words in blogs. Moreover, my available resources do not allow me to conduct training on vast data set over 10000 blogs. The training on vast amount of data could lead to bad performance while the CPU usage is limited.

The fourth issue is the usage of suitable data set. During training and testing phrase, I encountered

overfitting issue. I tried to add more data from an available public data set which contained blogs with gender of authors. However, overfitting was not solved, and my model produced lower testing accuracy. Later, I was curious and run the additional data on Naive Bayes classifier("1.9. Naive Bayes"), Support Vector Machine classifier("1.4. Support Vector Machines"), and Logistic Regression classifier("sklearn.linear_model.LogisticRegression"). The result was a surprise.

	Naive Bayes	SVM	Logistic Regression
dataset_1	0.6976	0.6945	0.703
dataset_2	0.6242	0.5797	0.5939

Figure 1: Testing Accuracy Results of data sets on many classifiers

In table 1, data set 1 is the original data set from paper "Improving Gender Classification of Blog Authors" (Mukherjee and Liu). Data set 2 is the new data set that I found online (Schler et al.). The new data set has lower testing accuracy than the original data set on many available classifiers in python language. Due to this feature of the source of new data, the result in my project could be affected when I added new data for training. In my assumption, the new data set may contain error in gender labels. The new data set may have a large amount of data and require better hardware tools to deliver true good result. To solve this issue, I quit using additional data, and I used only the original data set for this project.

The fifth issue is building a suitable convolution Neuron Network classifier for the classification task. It requires a suitable design of CNN model for the chosen data set. The suitable CNN model needs a good parameter selection for acceptable performance. If the CNN model is too complex, it can cause overfitting in training. In my opinion, this issue has been the most time-consuming because I needed to adjust parameter selection, confirmed suitable embedding weights, and observing training results before making changes in design of CNN model.

4 Methods

I used Python programming language for this project. Python is a popular tool for machine learning. Python offers data scientists many useful packages for data processing, training, visualizing, and evaluating, such as scikit-learn, numpy, pandas, keras, and so on. For convenience, I wrote python code on Jupyter Notebook.

4.1 Data Exploration

The data set is loaded into Jupyter notebook via Pandas. The original data set has 5 rows without gender label and 7 rows without text blog. The labels in gender column are not in the same format. For example, in male category, the labels have 3 different forms: "f", " F", and "F ". Hence, there is a need of pre-processing the data set before performing Natural Language Processing (NLP) techniques. The Pandas package helps remove empty rows from the original data set. At the same time, Pandas replaces the gender labels into only 2 format "F" and "M". Before training the data set on CNN network, I use Pandas to change gender labels into numeric values: 0 for male, 1 for female.

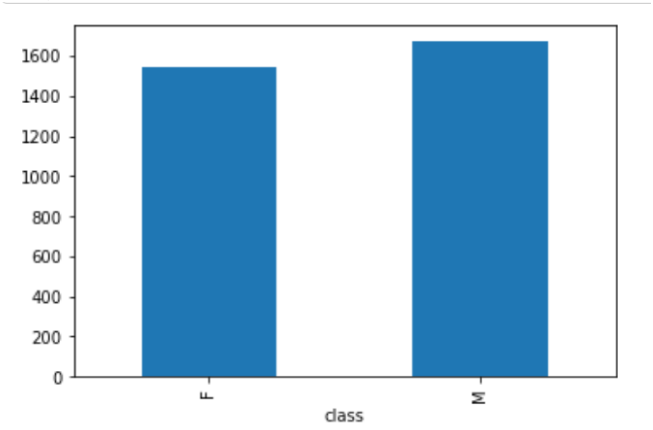

Figure 2: Bar graph of the clean data set

According to the Bar chart in 2, the number of blogs written by female authors is smaller than the number of blogs written by male authors: 1547 and 1677. The clean data set is a little imbalance, but the difference between two categories is only 2.4% percentage. In my opinion, this issue is not significant.

4.2 Natural Language Processing Techniques

The blogs in the data set must go through data cleaning process, so they could be changed into numeric data and become input for CNN model. I use many Python packages to perform this step, including Numpy, Re, Pandas, and NLTK. I could know the basic steps in NLP thanks to an online article(Arshad).

First, I remove punctuation in each blog by using functions in Numpy and Re. Next, I use tokenizer in NLTK package to tokenize each sentence in each blog of the data set. After the tokenizing step, each tokenized words will be changed into lower case. After lower casing the data, stop words are removed from data using NLTK's stopwords. At this step, I have performed all necessary functions of NLP, and the data set can be splitted into training set and testing set. In the final experiment, I use 85% data set for training and 15% data set for testing. To prepared for weight embedding step, I build training vocabulary and testing vocabulary from training set and testing set.

4.3 Weight embedding

To generate weights for input layer of CNN, I used en_core_web_lg model, a pre-trained word

embedding model in Spacy package. Previously, I used Google News Word2Vec as pre-trained word embedding model(Arshad). As I tried to improve model and solve overfitting issue, I switched to Spacy_core_web_lg model. The training data is put into the Spacy model to generate training embeddings with 300 as embedding dimension.

With the pre-trained model and list of training vocabulary, I assign each word to an integer and register that integer in a list. That list will become feature input for training model. The maximum length of a sequence in the list is 200. Next, I do the same progress with the list of testing vocabulary. People call this progress as padding(Arshad). In this padding process, I depend on Tokenizer() in Keras library.

After finishing padding, I get embeddings from en_core_web_lg model and save them corresponding to the sequence number I assigned to each word. If I could not get embeddings, then I save a random vector for that word.

4.4 Convolution Neuron Network model

To build CNN model, I use Keras library in Python. In figure 3 and 4, there are descriptions about design of my CNN model. I designed a simple model to limit issue of overfitting.

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 200)               0
_________________________________________________________________
embedding_1 (Embedding)      (None, 200, 300)          17831100
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 200, 32)           28832
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 100, 32)           0
_________________________________________________________________
dropout_1 (Dropout)          (None, 100, 32)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 150)               109800
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 302
=================================================================
Total params: 17,970,034
Trainable params: 138,934
Non-trainable params: 17,831,100
_________________________________________________________________
```

Figure 3: Detail of CNN model

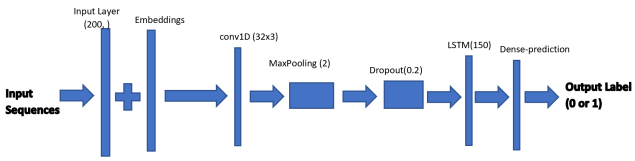

Figure 4: Graph of CNN model

The model starts with an input layer which will receive the input X and Y. X is the processed list of training sequences which were created by padding. Y is the list of training label. After the input layer, there is an embedding layer which will carry the embedding weights which were generated by en_core_web_lg model. Next, the processed features enter a Convolution layer with 32 filters, kernel_size=3, and "Relu" activation method. After going through the Convolution layer, the data will enter a MaxPooling1D layer and a Dropout layer. The Dropout layer helps reduce overfitting.

Next, the data will enter a LSTM layer. I read online articles, and I knew that the LSTm layer could help improve result of classification in CNN model(Ahamed). The Keras library in Python provides the available LSTM layer. Finally, the data will enter a dense layer with "sigmoid" activation method.

The CNN model is compiled with "binary_crossentropy" loss function. The optimizer is Adam from Keras library. The learning rate is set as 0.0001.

4.5 Evaluation Metrics

To evaluate the performance of the CNN model, I depend on testing accuracy and F-1 score. Sklearn library provides me enough tool to compute testing accuracy and F-1 score. F-1 score is an evaluation metric that combines both precision and recall:

$$f1 = 2 * (precision * recall)/(precision + recall)$$

F-1 score is the harmonic mean of precision and recall, being more suitable for ratios than arithmetic mean(Kanstren). If precision or recall is too small, then F-1 score will be very small. F1-score can help balance the metric across samples that has binary values.

5 Result

Figures 5 and 6 show the best result in training on the CNN model with 100 epochs and batch size being equal 512. The training loss keeps decreasing during training on 100 epochs. Meanwhile, the testing loss decreases strongly at the beginning until the 20th epoch. After the 20th epoch, testing loss fluctuates between 0.575 and 0.6. The trend of training loss and testing loss shows that overfitting still exists.

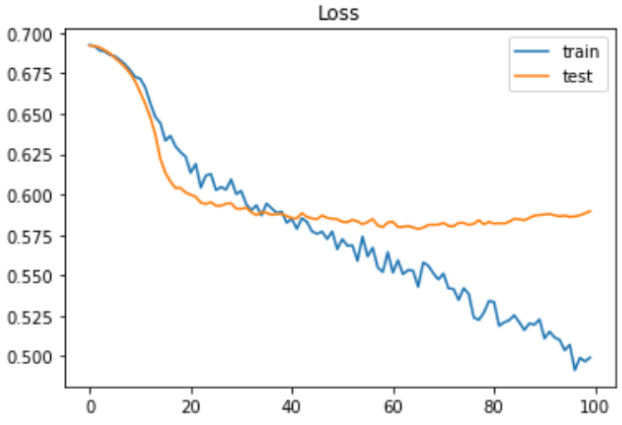

Figure 5: Plot of Training loss and testing loss

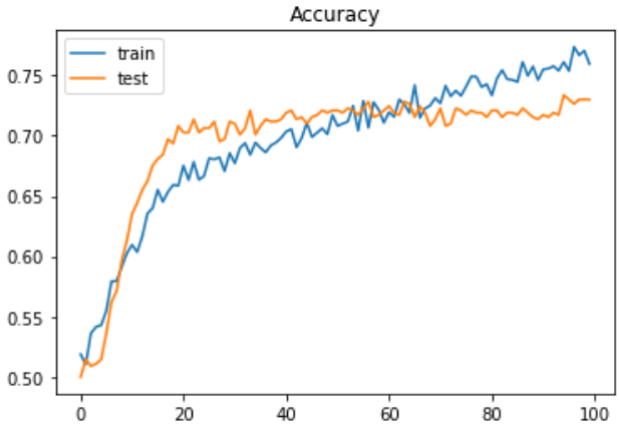

Figure 6: Plot of Training accuracy and testing accuracy

The training accuracy increases gradually during training process. The testing accuracy was increasing and higher than training accuracy at first, but after the 60th epoch, the training accuracy bests the testing accuracy.

When I ran model on the testing set for prediction, the accuracy is 69.11%. The result of F-1 score is 69%. Either predicting male label or female label, the model has 69% for accuracy.

6 discussion

In my opinion, the result is not truly satisfactory. I was aiming to have above 80% for accuracy and F1-score. The current overfitting issue could be due to errors in NLP processing. I may consider using the most suitable libraries for NLP tasks, not depending on only NLTK and Keras libraries for resources. The pre-trained en_core_web_lg model may not be the most suitable choice for the chosen data set. The current Parameter selection can be optimized more.

During the project, I could learn many experiences. The biggest experience would be working with overfitting issue. I tried many different methods to solve the issue, such as parameter selection, simple CNN model, pre-trained embedding model, or one-hot encoding. The second biggest experience is the time constrain. I may plan things, but the training and improving phrase is time-consuming. The third biggest experience is rebuilding things from scratch. I tried to build a CNN model for classification task, so I had to change my code and tools many times to improve the training result.

7 Future Plan

I plan to improve the result of the project, aiming for 80%. I will try to improve my NLP experience. I have not been able to utilize Part-of-Speech in this project. The usage of Part-of-Speech may deliver more positive result in testing accuracy.

Another limit in my project is the resource. I might use my own laptop and Kaggle to try to train the CNN model with a vast amount of data, but things got frozen. Hence, I plan to conduct a bigger text classification project in a strong Cloud Computing service, such as Amazon Web Service or Google Cloud Platform.

This project is limited within the chosen data set and English language. It would be nice if the CNN model can predict gender of authors when blog authors use different languages to write blogs. Moreover, the trained CNN model can be used as backend of a website that can help predicting gender of blog writers.

8 Conclusion

This project aims to build a CNN model that can classify whether a blog is written by a male or a female author. In this project, I applied many NLP techniques to process blog text and perform weight embedding. I also researched many online resources to be able to build the CNN model at the end.

The result may not be as I planned, but it is still fair. The CNN model can predict the gender of blog authors with 69% for F-1 score. The CNN model can be improved in the future to perform training on a bigger text classification project which uses similar data set.

Works Cited

"1.4. Support Vector Machines." Scikit-learn developers, 2007. Web.

"1.9. Naive Bayes." Scikit-learn developers, 2007. Web.

Ahamed, Sabber. "Text Classification Using LSTM and visualize Word Embeddings: Part-1." Medium, Jan. 2018. Web.

Arshad, Saad. "Sentiment Analysis / Text Classification Using CNN (Convolutional Neural Network)." Towards Data Science, Sept. 2019. Web.

Kanstren, Teemu. "A Look at Precision, Recall, and F1-Score." Towards Data Science, Sept. 2020. Web.

Mukherjee, Arjun and Bing Liu. "Improving Gender Classification of Blog Authors" (Nov. 2010): 207–217. Web.

Schler, Jonathan, et al. "Effects of Age and Gender on Blogging." Jan. 2006. 199–205. Print.

"sklearn.linear_model.LogisticRegression." Scikit-learn developers, 2007. Web.