Machine Learning Engineer Nanodegree

Capstone Proposal

Japtej Singh Arneja
July 29th, 2019

Proposal

Domain Background

Natural language processing (NLP) is a subfield of computer science and artificial intelligence that is concerned with interactions between computer and humans[1] and covers manipulation of natural language like speech and text by computers. Text is everywhere – newspapers, web pages, documents, SMS, emails and online surveys are all examples of textual information. Still, the ability to process and extract insights from textual data using computers was limited up until 1990 as researchers mainly relied on bespoke set of hand-written rules[2] e.g. writing rules for grammar, stemming, etc. However starting 1990s, the usage of machine learning for NLP applications gained momentum and widespread industry adoption and especially in the last decade, with the advent of higher computing power (e.g. GPUs), deep learning frameworks (like ANN, CNN, LSTM, etc.) have penetrated every aspect of NLP achieving state of the art performance on many textual tasks. Specifically in this project, we aim to employ various machine learning and deep learning frameworks for sentence/excerpt classification. In the process, we would also compare performance of conventional frameworks with some of the more contemporary ones.

Personally, I often work with textual information including survey-based data where we are required to extract insights from open-ended verbatim such as its sentiment, topic, etc. Expertise in various machine learning based NLP frameworks for text classification would be a huge plus in order to effectively handle such kinds of problems in the future.

Problem Statement

The problem is to be able to map textual excerpts to their respective (three) authors. Specifically, the excepts are all taken from spooky stories penned by each of these authors and hence the fundamental topic (horror) is the same – therefore the prediction algorithm has to mainly rely on inferring the writing styles to be able to predict the author. Each sentence belongs to only one author and hence the model accuracy and hence performance can be directly evaluated as per the metric in the Evaluation section of this proposal below.

Datasets and Inputs

The dataset used is procured from a past Kaggle competition. The link to access the data file is the following:

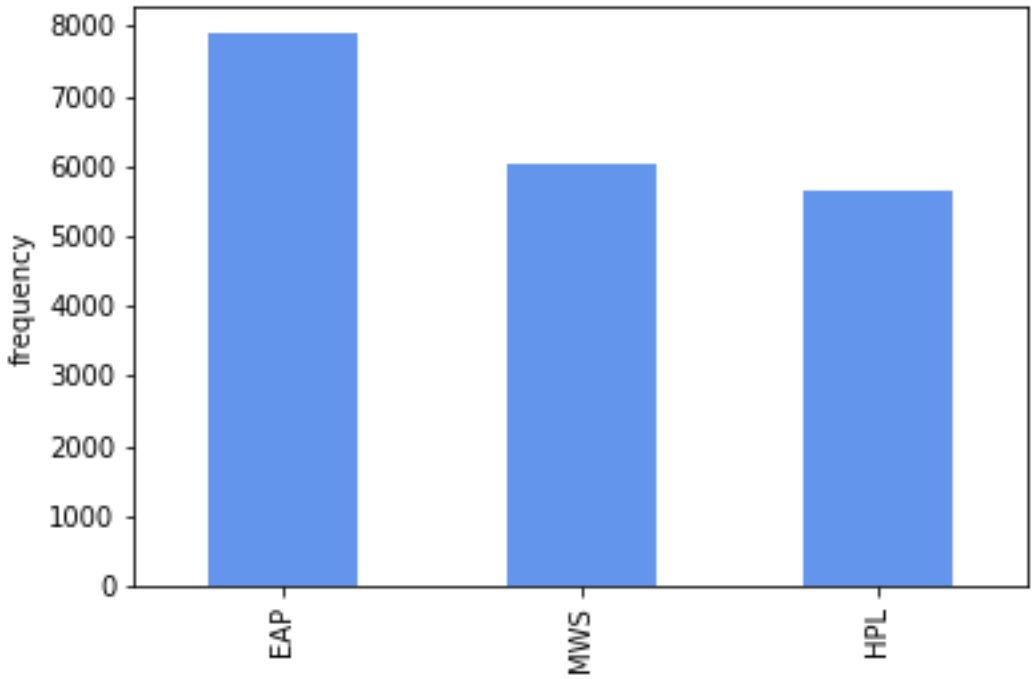
- https://www.kaggle.com/c/spooky-author-identification/data

The excerpts are single long sentences and are sourced from various works of fiction written by the following three authors viz.:

1) Edgar Allan Poe (EAP),
2) HP Lovecraft (HPL),
3) Mary Wollstonecraft Shelley (MWS)

This dataset would be used to predict one of the three authors for a given excerpt. The datasets provided is divided into train and test. The train dataset contains 19579 labeled excerpts while the test dataset contains 8392 excerpts. However, since the test dataset is unlabeled (for Kaggle scoring purposes), for the purposes of evaluating the models in this project, the test dataset would be carved from within the given train dataset. The test data would be in the range of 10-20% of the overall train data (would be decided during project implementation).

As can be seen from the train data frequency plot below, each author has substantial number of excerpts with the author EAP having the highest number of excerpts in the data. Hence, we are not likely going to suffer with unbalanced class problem in this data and don't really need to differentially sample (oversample/under-sample) any class. We can still try oversampling the MWS and HPL classes in case we are not getting much success.



In addition, as would be mentioned in the solution and project design, we would also be using pre-trained word embeddings as input to our model. They can be (but not limited to) the following:

- Word2Vec: https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?usp=sharing
- GloVe: http://nlp.stanford.edu/data/glove.6B.zip

Solution Statement

As the input at hand is textual data, this problem falls in the realm of Natural language processing (NLP). We would fit various ML/DL models on the processed textual data and its metadata. We would try various kind of inputs viz. bag of words (both 1-gram and n-grams), TF-IDF, word embedding (word2vec, glove) or even simply the excerpt metadata (like word count, character count, etc.). On the modelling front, we would start with conventional ML models like Logistic regression, Naïve Bayes, Support vector machine, etc. and then also graduate to neural net models suitable for text classification viz. dense ANN, CNN, RNN (LSTM, GRU), etc. We would also explore character based n-

gram models like Facebook's FastText, which are faster even while running on CPU, work better for small train corpus and also address out of vocabulary (OOV) problem to a large extent. Finally, if time permits, we might also try the recently constituted BERT[4] model by Google which has shown state of the art performance on various language modeling tasks including text classification.

Benchmark Model

The benchmark model would be multi-class logistic regression model fitted on single words (1-gram) bag of words input. We would then compare the performance of the proposed solution with this baseline model using the evaluation metric listed in the evaluation section.

Evaluation Metric

We will be using categorical cross-entropy (or multi-class logarithmic loss) as the evaluation metric here which is also the metric used in the Kaggle competition.

Cross entropy is based on the concept of maximum likelihood of predictions i.e. a model which predicts higher probabilities for the actual observed events is better than models that don't quite does so. Mathematically, cross entropy (or log loss) is given by the expression:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

where, N = number of observations (text excerpts in this case)
M = number of classes (3 in this case)
y_{ij} = 1 if observation i belongs to class j else 0
p_{ij} = predicted probability that observation i belongs to class j

The theoretical range of cross entropy loss is 0 to ∞.

Project Design

The submission would be broken down into the following parts:

1) Data exploration
2) Data pre-processing
3) Feature engineering
4) Model fitting

Data exploration

Here we would conduct exploratory data analysis (EDA) on the textual data e.g. distribution of target labels, word clouds of excerpts by various authors, and try to visualize some of the text meta-features that might help us in model prediction, for e.g.:

- count of characters in each excerpt,
- count of words in each excerpt,
- count of punctuations in each excerpt,
- count of nouns, adjectives, etc. in each excerpt,
- average length of words in each excerpt
- *and more*

We would be drawing EDA plots to verify any discernable patterns in writing styles of each of the authors.

Data pre-processing

Next, we would be doing some data processing on the text based on the objectives, which might include:

- spell corrections
- stemming/Lemmatization
- stop words removal
- punctuation removal
- and more

We would be using libraries like NLTK or spaCy for this task. Various classification models would be explored with and without some of the above pre-processing steps. The intuition is that some of the features like punctuation, etc. might still be useful in deciphering the writing styles and hence prediction of the three authors.

Feature engineering

Next we would compile features that can be used as inputs to various text classification models. Some of the features that would be tried are:

- Bag of words (including n-grams)
- TF-IDF vectors
- Word embeddings from pre-trained models like Word2Vec and GloVe
- text meta-features like character count, word count, etc.

Model fitting

Next task would be to fit models for author prediction. They could be broken down into various classes:

- Conventional models like Logistic regression, Naïve Bayes, Random Forest but also more powerful models like xgboost, etc.
- Neural net models like simple Dense NN, CNN, RNN (LSTM/GRUs), Bi-RNN (e.g. Bi-LSTM), etc.
- Character based models like FastText (by Facebook)

We might also try making an ensemble of some of these models to try to achieve an even higher evaluation metric score. All along, we would also try to tune the hyperparameters for the promising models to achieve an even higher score on evaluation metric.

Finally time permitting, we might try more contemporary models like BERT (by Google) that are achieving state of art performance on many language tasks including text classification.

For model fitting, we would be mainly relying on *scikit-learn* library for conventional ML model fitting and *keras* for neural net based model fitting.

References

[1] https://en.wikipedia.org/wiki/Natural_language_processing

[2] https://www.forbes.com/sites/forbestechcouncil/2018/11/06/the-evolution-of-natural-language-processing-and-its-impact-on-ai/#131b51b51119

[3] https://arxiv.org/pdf/1410.5329.pdf

[4] https://arxiv.org/pdf/1810.04805.pdf