Marifel Barbasa
Sunday, July 29, 2018
MLND March 2018 Class

Identifying Horror Authors from Samples of their Writings

Machine Learning Capstone Proposal
Machine Learning Engineer Nanodegree, Udacity

Domain Background

Authorship attribution or identification is the problem of identifying who wrote a piece of text. Determining the author of a body of written work is an important task in natural language processing (NLP), information retrieval, and computational linguistics, as doing so correctly can help resolve forensic investigation cases, uncover plagiarists, give proper attribution to historical or ancient works, provide readers with recommendations of authors with similar writing styles, and determine anonymous authors or those who pose under a pseudonym. Consequently, solutions to authorship attribution help fight crimes and make it more difficult for people with malicious intent to distribute written content, such as those with the aim of posting fake news under a pseudonym or multiple pseudonyms.

Personally, I'm interested in learning and implementing NLP techniques to solve a wide array of problems with natural language data; thus, the problem of authorship attribution seems like a fun challenge and introduction to the field. I can see how it is truly critical in this day and age of vast information overload to be able to determine which authors we can trust. Furthermore, I hope to take what I learn here and eventually deploy a production application that, in the style of an author, generates textual prompts, be it song lyrics, fiction literature, technical writing, or blog post prompts.

Problem Statement

Given labeled samples of their writings, this project aims to predict the probability that each author in the corpus wrote a piece of text that the machine learning model has not yet seen. This multiclass classification task is quantifiable because it seeks to solve the authorship attribution problem by expecting an output probability distribution across authors per unseen writing sample. The probability predicted per author translates to how likely the author is to have written the sample, and the probabilities predicted per sample should approximately sum to 1. The problem is also measurable because, given a number of unseen samples with the actual labels set aside, the predictions on those unseen samples can be measured using multiclass logarithmic loss or logloss. And the lower the logloss, the better the model is at making accurate predictions. The problem is also replicable because, given the same set of data, one can reproduce the experiments.

One potential solution to the problem would be to apply NLP techniques such as text cleaning, tokenization, bag of words, and word embeddings to transform the text data into a numerical format before feeding it into various machine learning models such as Naive Bayes classifiers and recurrent neural networks (RNNs).

Datasets and Inputs

The Spooky Author Identification Playground Prediction Competition hosted by Kaggle[1] challenges machine learning engineers to identify the horror authors Edgar Allan Poe (EAP), Mary Wollstonecraft Shelley (MWS), and H.P. Lovecraft (HPL) from samples of their writings. Hence, the dataset calls for a multiclass classification solution to the problem, in which each of the three authors is a target class and the only input is the textual excerpts of their fiction writings. The task is to assign probabilities to EAP, MWS, or HPL, given a sentence from one of their works; in other words, the task is to determine how likely it is that each of the three authors wrote a particular sample.

While the data is real in that the authors truly wrote these pieces, Kaggle has prepared and labeled this dataset of 19,579 training samples, where each sample is a sentence drawn from a written work of the author. Kaggle states that, because the "data was prepared by chunking larger texts into sentences using CoreNLP's MaxEnt sentence tokenizer," occasional incomplete sentences may appear in the samples.[2]

This appears to be the perfect introductory dataset to authorship attribution since it is a relatively clean dataset and yet still poses a challenge due to usage of uncommon words such as "pursy" and uncommon spellings, such as "untill." An added bonus is that I have always had an admiration of the English language used in centuries past due to its elegance and sophistication, so this characteristic of the dataset might pose a challenge as well.

Solution Statement

In this project, I plan to use bag of words and pre-trained word embedding features with support vector machines (SVMs), Naive Bayes classifiers, and neural networks, including convolutional neural networks (CNNs), plain RNNs, and RNNs with long short-term memory (LSTM) and gated recurrent unit (GRU) layers. I would also like to narrow down these algorithms by using random search over certain hyperparameters. And in order to test model performance, I will use 5-fold or 10-fold cross validation on the logloss metric.

---

[1] https://www.kaggle.com/c/spooky-author-identification
[2] https://www.kaggle.com/c/spooky-author-identification/data

Benchmark Model

I have examined the distribution of target classes on the Spooky Author Identification dataset and will use those numbers, of samples per author, as a basis for random guessing. I can see that Kaggle has already provided that benchmark in the `sample_submission.csv` file, and the probability per author (same probabilities across all samples, so that EAP is always 0.40, HPL is always 0.29, and MWS is always 0.31) exactly matches the percentage of samples that each author has in the total training dataset of 19,579 samples. But my benchmark model will be a little different from Kaggle's benchmark in order to more closely mimic an actual machine learning model. The benchmark model I'm proposing will be different in that, for every sample, a random probability distribution across the three authors will be generated, with the constraint that EAP is assigned the highest probability for 40% of the samples, HPL is assigned the highest probability for 29% of the samples, and MWS is assigned the highest probability for 31% of the samples. I will generate 100 of these predictions, compute the logloss for each, and then take the mean logloss score as my benchmark to beat.

Evaluation Metrics

An evaluation metric that can quantify the performance of both the benchmark and solution models is the logloss, which is defined as the negative of the mean log-likelihood over the number of samples. This means that, for each test sample, select the model's output predicted probability for the true class, `p`, and take its logarithm. Then add all of these values together and divide by the number of test samples; lastly, negate this value to obtain the logloss for the model. Note that, despite the model outputting a probability for each author per test sample, only the true author's probability will be included in the logloss score. I will use the natural logarithm for the log function to match Kaggle's own evaluation metric and, also to match Kaggle, will replace predicted probabilities, `p`, with `max(min(p, 1 - 10^-15), 10^-15)` in order to avoid the extremes of the log function[3].

Project Design

I will employ the typical approach to working with text data, which involves four steps: exploratory data analysis (EDA), text processing, feature extraction, and modeling. With EDA, I plan to visually skim the data to see what I'm working with, analyze count features such as basic statistics over all the text samples and those per author, and discover the number of the most common unigrams/bigrams/trigrams used per author.

With text processing, I plan to attempt various cleaning tasks and evaluate what combination works best. Some of these tasks are: lowercasing, replacing punctuation with spaces, removing stopwords, eliminating infrequent occurrences (such as less than 5 or less than 10) of words

[3] https://www.kaggle.com/c/spooky-author-identification#evaluation

from the vocabulary, stemming, and lemmatization. After the text is cleaned, I'd like to try out tokenization at the character and word levels.

For feature extraction, I plan to try bag of words with count and term frequency-inverse document frequency (TF-IDF) scoring and then attempt pre-trained word embeddings. With pre-trained word embeddings, I'd like to experiment with word2vec, GloVe, fastText, ELMo, and CoVe and evaluate what works best.

Once the text has been converted to numerical input vectors, I plan to apply the following models during the modeling phase: an SVM classifier with a linear kernel using LinearSVC from scikit-learn, a Naive Bayes classifier using GaussianNB and MultinomialNB from scikit-learn, and CNNs, plain RNNs, LSTMs, and GRUs from Keras. These models will be applied to different features, such as bag of words and word embeddings.

Each model will be evaluated based on the logloss using either 5-fold or 10-fold cross validation; the lower the logloss, the better the model. Apart from evaluation, for either the top two performing algorithms or for each algorithm, I plan to run random search to tune certain hyperparameters for each algorithm for at least 60 iterations[4] (but I'll lower that number if it ends up taking way too long) in order to find the best model for this multiclass classification problem of authorship attribution.

[4] https://stats.stackexchange.com/questions/160479/practical-hyperparameter-optimization-random-vs-grid-search