CMT316 - Coursework 1, Part 2

Peter Kennedy, 2092220

April 20, 2021

Part 2

Overview

Throughout this article, a cross section of feature extraction and engineering techniques are explored. A range of combinations of techniques are mentioned in order to demonstrate the authors understanding of the subject. There are likely better combinations of features and techniques which could be used. The possible improvements will be discussed under the final heading.

The associated code and data may be found in the following GitHub repository: https://github.com/Mirrafoil/CMT316-Coursework1.

Throughout this article, references will be made to 'Code Blocks' (CBs) within the Jupyter notebook.

The file `CMT316 − Coursework1.yml` may be used to ensure that the python environment being used is configured properly. This YAML file may also be found within the GitHub repository.

Cross validation

To determine prediction metrics for the chosen features during experimentation, cross validation is used. As a convention, 10-fold cross validation is used.

Feature extraction

Article Length

The first, and simplest feature which was used to attempt to classify the documents was simply the length of the articles (number of characters).

Simple vocabulary - Bag of words

The second feature extraction method used was a simple bag of words. A vocabulary is generated by iterating through each word within each article in the training set. If the lemmatised word is a stopword, it is ignored. If the word does not exist within the vocabulary dictionary, it is added and the value of the word set to 1. If the word is not already in the dictionary, it is incremented by 1.

Once the dictionary is constructed it is then sorted in order of frequency. The top l occurring words are then returned in a list as the desired vocabulary. This vocabulary is then be used to convert each article within the training set into a vector, where the occurrence of each word is counted as an instance of the respective component of the vocabulary.

Part of Speech (POS) vocabulary

The third feature extraction method used was to construct a vocabulary from the most commonly occurring 'Parts of Speech' from each of the categories.

The POS tags in a given article are found by utilising the spaCy library. Firstly, a given article is converted into a spaCy document. Next, each 'token' within the document object is checked for its POS tag[1]. Each token within the article is then iterated through, and is only added to the vocabulary dictionary if it is tagged with an permissible POS tag.

This POS vocabulary dictionary is constructed in a slightly different way than the simple vocabulary to ensure that each category contributes proportionally to the vocabulary[2]. The vocabulary is sorted and the articles vectorised in the same way as it was for the simple vocabulary.

Feature engineering

Best K features

The best K features can be found when using the simple or POS vocabularies as features. This can be done using the chi squared distribution to check which features are more 'important' with respect to each other, and only using the top K most important.

Normalisation

Performance of the classifier can be seen to improve by normalising the vectorised articles. This ensures that each component of the vectorised articles have the same normalised length scale.

TF-IDF

Performance of the trained classifier can also be improved by applying the Term Frequency - Inverse Document Frequency (TF-IDF) transformation to the vectorised articles. This works in a similar way to normalisation, but also applies a stronger weighting to less frequently occurring words, and a lighter weighting to more commonly occurring words.

[1]Tokens in spaCy document objects can take one of seven different POS categories: https://universaldependencies.org/docs/u/pos/

[2]Explained further in lines 41-46 of CB3

Model fitting

The model is a simple SVM classifier with a linear kernel. The linear kernel has been chosen since it also appears to perform much better than the polynomial and RBF kernels during experimentation.

Results

Throughout experimentation, 10-fold cross validation was used.

Article length

After running an experiment using the article length as the sole feature, the following results were found

Average accuracy: 0.2966240409207161
Average precision: 0.18944273442233997
Average recall: 0.2859051816770329
Average F1-score: 0.2035779170961211

Although each of these metrics appear to be quite poor, they are in fact better than random predictions. There are 5 categories, so would expect a random prediction to yield 0.2 on each of these metrics. Since these the accuracy metric is approximately 0.3, it could be said that there is at least some information in the length of the article which might be used to classify it.

Simple vocabulary

Using the simple vocabulary, the following parameters were found to be the best which had been tested

Vocabulary length: 3000
K-Best features: 1500
TF-IDF Transformation: True

Which yielded the following results

Average accuracy: 0.9513554987212276
Average precision: 0.9519158917510685
Average recall: 0.9498378034867512
Average F1-score: 0.9504414441275977

From this it may be seen that the simple vocabulary works as a very good feature to train the classifier on, with only 5% of articles being misclassified.

POS vocabulary

After running a model for every possible combination of POS tags[3], a vocabulary from only the nouns and proper nouns within each article yielded performance metrics comparable to those in the simple vocabulary. By setting the following parameters during experimentation

POS array: ['NOUN', 'PROPN']
Vocabulary length: 3000
K-Best features: 1500
TF-IDF Transformation: True

the following performance metrics were achieved

Average accuracy: 0.9431713554987212
Average precision: 0.9457891719432807
Average recall: 0.940510330736722
Average F1-score: 0.9426236454298994

Although greater performance metrics have not been achieved than using the simple vocabulary, it is interesting to note that the when it comes to classifying the news articles, the vast majority of information appears to reside within the nouns which make up the articles. By using any combination of POS tags which does not include nouns, the performance of the classifier which is trained is significantly diminished.

Using the best model

The parameters for the best (found) POS vocabulary have been written into the saved notebook. For training the model on the entire dataset for making predictions on unseen data, instructions are in the notebook itself.

Improvements

Word Embedding

To improve on the performance of the classifier, the next features which should be explored are word embeddings. By using the word2vec algorithm (or similar), additional context of words may be added to the vector. These higher dimensional vectors could then be used to train a classifier on.

[3]Methodology described in lines 22-28 of CB4.

Parameter and feature combinations

Since the number of parameter combinations is effectively infinite, an approach that could be taken would be to apply a genetic algorithm to these parameters. Admittedly, this would require running many models, however, this would be significantly fewer models required if parameter tuning was attempted by brute force.

Experimentation and training time

Improvements could easily be made on the time which experimentation and model training takes by optimising the vocabulary construction functions, making better use of sklearn 'in-built' functions and in the case of cross validation, using the python multiprocessing library to carry out training fold evaluation of multiple folds simultaneously.