Fake News Identification

Username: gcdk35

March 11, 2018

1 Data Cleaning and Representation

The first step in both shallow and deep learning is to take the dataset given (as a CSV file), filter it, and store the data in a way that can assist learning.

First, the entire file is loaded in, with each component being stored as a list (0 index being the index ID, 1 index being the corpus content, and 2 index being the label).

From here, we go through the list, removing all punctuation from the corpus, leaving only words seperated by a single space, all converted to lower case. Lower case conversion ensures that the same word is not counted differently (i.e. "Fox" will be seen as a different word to "fox" for example). Punctuation removal ensures that punctuation doesn't cause words to be perceived as different words (e.g. "fox!" will be seen as a different word to "fox").

From here, each classifier undergoes their own specific routines, which shall be revealed in future sections.

2 Shallow Learning Approach

This approach uses a Multinomial Bayes Classifier, with different methods of extracting features. Each one shall be outlined below, along with the resulting accuracies.

2.1 N Gram Only

Features can be extracted using n-grams (a set of words). The default within sklearn is size N=1 (one word) but increasing this size does yield slightly better performance on its own. The results, including precision, accuracy, recall and F1 scores can be found in Table 1

2.2 Term Frequency

This automatically uses ngrams, then inputting into a Term Frequency Algorithm. Term Frequency goes through, calculating the number of times a particular ngram has occurred in the corpus. THe intuition to this method is that more important words have a greater

Table 1: Ngram only performance

Ngrams	Precision	Accuracy	Recall	F1
1	0.885852229040716	0.88348594884749	0.88348594884749	0.883237395852024
2	0.894373523019958	0.877170824123776	0.877170824123776	0.875624072747213
3	0.898585092457609	0.882222923902747	0.882222923902747	0.880823839680174
4	0.883984637640072	0.873697505525734	0.873697505525734	0.872688593358964
5	0.864856516164587	0.860435743605936	0.860435743605936	0.859890709235547

count that the less important ones. While this is true, certain common words such as "the" might have a high frequency but add nothing to the overall meaning of the corpus. Therefore, stopwords are used, which are filtered from the corpus, to ensure their presence doesn't impact the results of the output.

TODO: TABLE OF RESULTS FOR TF WITH NGRAMS

2.3 TF-idf

Much like Term Frequency above, this adds an additional weight to each ngram. After finding the term frequency score, we multiply it by the Inverse Document Frequency score (idf score). The IDF score represents the amount of information a given ngram contains - if many documents contain the same ngram, then it's likely that the ngram is a common ngram, and thus not important. This results in an advanced version of TF that automatically removes stopwords without needing to supply a list.

TODO: table of results for TFIDF with NGRAMS

3 Deep Learning

To carry out the deep learning process, Keras was used with the Tensorflow backend. In the case of all deep learning networks, they were trained on 60% of the dataset, with the remaining 40% of the data being used as test data. This provides sufficient data to train on, but also ensures we can test the model using enough data.

Several different models were created using LSTMs.

3.1 LSTM with 100 nodes

3.2 LSTM with 200 nodes

3.3 LSTM with 200 nodes and increased dropout

3.4 LSTM with larger dense layer

Instead of just passing the output of the LSTM straight to the dense layer with 2 nodes, we put a layer with 100 nodes between these two layers, the idea being to provide further weights and nodes to train a more complex model. Results by adding this layer increased, yielding an 83% accuracy on the test data.