**Dataset:**

All tests were performed using four categories (politics.guns, politics.mideast, politics.misc, religion.misc) of articles from the 20newsgroups dataset available through sklearn. This amounted to a total dataset including 1952 training articles and 1301 test articles with the following distribution of classes:

|  |  |  |
| --- | --- | --- |
| Class | Training Count | Testing Count |
| politics.guns | 546 | 364 |
| politics.mideast | 564 | 376 |
| politics.misc | 465 | 310 |
| religion.misc | 377 | 251 |

**Text Cleaning:**

All cleaning of the test text dataset was performed using a python script with the SpaCy package unless otherwise stated. Cleaning was performed in the following process:

1. Remove header information
2. Tokenize the text at the word level, which also cast everything to lowercase and split conjunctions
3. Removal of stop words
4. Removal of punctuation
5. Lemmatize each word token
6. Manually remove special characters and excess newline delimiters missed in first pass

**Text Preparations for Modeling:**

I used the following text preparation/vectorization methods on the cleaned text before modeling. I will indicate which methods (by number) were used with each algorithm.

1. Bag of Words (including single tokens, bigrams, and trigrams)
2. Term Frequency – Inverse Document Frequency (tests ranging from single word tokens up to 6-grams)
3. Token sequence vectors (testing including only single word tokens and token chunking up to trigrams)
4. Token sequence with individual word embeddings (Capped at 2000 words per document. Any documents with fewer than 2000 words were zero padded on the right.)
   1. Starting with random word embeddings that are then trained in the neural networks
   2. Training the embedding vectors using the dataset with SpaCy and gensim’s Word2Vec
   3. Starting with pretrained GloVe embedding vectors
5. Mean text embedding
6. Bag of Words + mean text embedding
7. TF-IDF + mean text embedding
8. Token sequence vectors + mean text embedding
9. Individual word embeddings + mean text embedding

**Modeling Algorithms tested:**

Below is a list of classification algorithms I have tested, along with a list of text preparations used on each one.

1. Naïve Bayes (1, 2, 3, 5) Note: I had to shift text embedding values to all positive for use with NB
2. Linear Logistic Regression (1, 2, 3, 5, 6, 7, 8)
3. Support Vector Classifier (1, 2, 3, 5, 6, 7, 8)
4. Neural Networks
   1. Fully connected dense networks (1, 2, 3, 4, 5, 6, 7, 8, 9)
   2. CNN (1, 2, 3, 4, 5, 6, 7, 8, 9)
   3. RNN-LSTM (3, 4)
   4. RNN-GRU (4)
   5. CNN + RNN-LSTM (4)
   6. CNN + RNN-GRU (4)

Note: I have not done a deep dive on tuning hyperparameters for each of these models. I have done only done a quick grid search on the best performing versions of the Naïve Bayes, Linear Logistic Regression, and SVC tests. I have not done any fine tuning of parameters in each layer of the Neural Nets and have been sticking to common use parameters from papers and articles on the subject.

**Results**

Results from my tests can be broken into two categories, with the first category being the most accurate models possessing transparent results (i.e. non-neural network algorithms using non-embedding text preparations), and most accurate models without transparency in their results.

For the first category, the most successful model was an SVC using TF-IDF vectorization all combinations of text up to 5-grams. Inclusion of up to 6-grams did not result in any further increase in accuracy using this algorithm. The progression of accuracy in the SVC models tested using a TF-IDF preparation with progressively increasing number of word combinations (N) was as follows:

|  |  |
| --- | --- |
| N | Accuracy |
| 1 | 83.0% |
| 2 | 84.8% |
| 3 | 85.0% |
| 4 | 85.2% |
| 5 | 85.5% |
| 6 | 85.5% |

Results from these tests were extremely easy to understand as a list of important features (words) used to determine each category could be retrieved, and the most important words for each category were very well matched. Below is a sample of the five best predicting words for each of the four categories:

Politics.guns: gun, weapon, firearm, fire, handgun

Politics.mideast: Israel, Israeli, jews, turkish, armenians

Politics.misc: tax, president, cramer, drug, clinton

Religion.misc: god, jesus, christian, bible, objective

The Naïve Bayes algorithm run with Bag of Words preparation including up to 6-grams in the text did technically outperform the best SVC model by achieving an 86.1% accuracy, but the resulting list of important features for each category were not nearly as transparent as those shown above. And if we are willing to give up transparency for an increase in accuracy, then the best performing models become deep neural networks. In this category, the model with the highest accuracy achieved so far was 91.0% using a neural net with multi-channel cnn + multi-channel lstm layers on word vectors with pretrained GloVe embeddings. Current next level of testing involve increasing the depth of the cnn layers and the inclusion of batch normalization.