20 Newsgroups Classification

Dataset: The dataset is a database of newsgroup emails spanning a number of different topics. I used the dataset that split the data into training and testing sets in the proportion of 60:40.

Classification task: Binary classification of documents into computer related and not computer related. This seemed like a logical task for this dataset since so many groups relate to computers and it is a narrower subject than science, which also has a similar number of groups in the dataset. I had wanted to explore multiclass classification but Pyspark 1.6.0 does not have support for multiclass classification for logistic regression and SVMs and I wanted to do a comparison of algorithms.

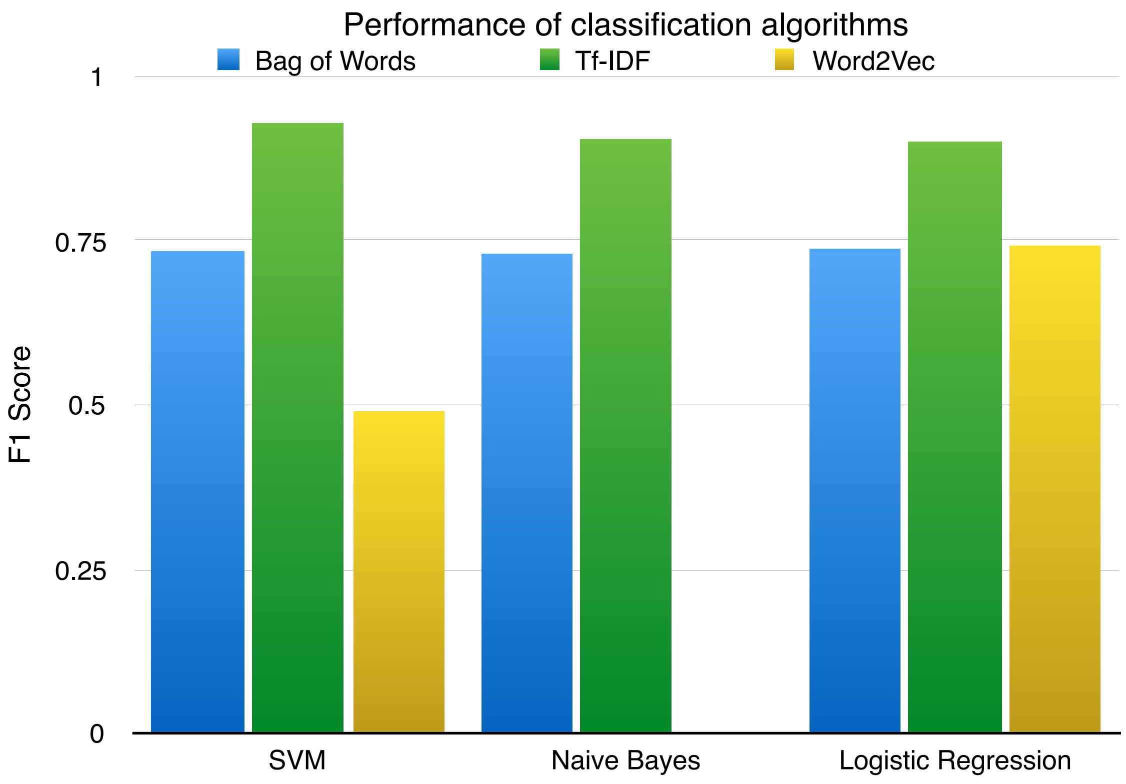
Methods and algorithms used: 3 text representation formats - Bag of words, TF-IDF and Word2Vec and 3 classification algorithms - Naive Bayes, Logistic Regression and Support Vector Machines.

Evaluation metric: F1 Score which is a good measure of both precision and recall.

Steps taken:

- First I created two RDDs - one of the training data and one of the test data.
- I then labelled all the documents as 1 if it is computer related (all groups beginning with comp) and 0 if it is not.
- Then I did some data cleaning - removing extraneous characters, numbers, footers that have irrelevant information, some headers that seemed to distract from the main task of classifying based on topic vs classifying based on irrelevant header information like fields From, NNTP-Posting-Host. I converted all text to lowercase.
- I tokenised the data and removed the stop words.
- Then I built three ML pipelines - one to implement a bag of words representation, another to implement a TF-IDF representation and the third to implement a Word2Vec representation.
- I fit both the training and testing datasets using these pipelines to convert all documents to feature sets.
- Then in a loop, I ran three algorithms - Naive Bayes, Logistic Regression and SVM on all three data representations to classify the documents. I built the model using the training data and validated the model using the testing dataset. I experimented with different regularisation parameters and other parameters and chose the parameters that yielded the best results.

Results:



- The best performing pair is SVM with Tf-IDF (~93%). SVMs are known to perform better than others as classifiers and they have done so here as well.
- The best data representation format was Tf-IDF which saw all three algorithms giving their best performance. This makes sense since Tf-IDF emphasises the more important words in a document, thereby helping the classification process.
- I think the Word2Vec vector results are misleading since I was forced to keep the vector size to 300. It might have performed better had the vector size been larger, but I kept getting an out of memory error whenever I tried to increase the vector size.
- Compared to Tf-IDF, Bag of words performs poorly on all three algorithms - since it does not differentiate between common and rare words and does not capture any essence of the document, it predictably lags behind Tf-IDF as an aid to classification.
- Since Word2Vec yields negative values as well as positive ones, I couldn't use it with Naive Bayes as Bayes only accepts positive values.

All quantitative results are documented in output.txt

Notes on usage:

Please type the following to execute the program:
spark-submit 20newsgroups.py

The input directories should be unzipped and placed in HDFS. I used 20news-bydate.tar.gz version of the dataset.

Word2Vec results change from iteration to iteration.

An output file with the results will be generated, called 'output.txt'.

References:

https://wesslen.github.io/twitter/predicting_twitter_profile_location_with_pyspark/
https://spark.apache.org/docs/1.6.0/api/python/pyspark.mllib.html
https://spark.apache.org/docs/1.6.1/api/python/pyspark.ml.html
https://spark.apache.org/docs/2.1.0/ml-features.html
https://spark.apache.org/docs/2.2.0/mllib-evaluation-metrics.html
https://spark.apache.org/docs/2.2.0/mllib-linear-methods.html
https://docs.cloud.databricks.com/docs/latest/sample_applications/07%20Sample%20ML/MLPipeline%20Newsgroup%20Dataset.html
https://spark.apache.org/docs/2.2.0/mllib-feature-extraction.html
https://spark.apache.org/docs/2.1.0/ml-classification-regression.html