Code

Please download the code from : <https://github.com/thisunhemakumara/ir-txt-clssfr.git>.

First, download or clone the git repo. To execute the code, run jupyter notebook newsgroup\_classifier.ipynb from the directory to which you downloaded the code. Then run each cell in jupyter notebook. Cell 1, 2 and 3 contains the code of Features, Classifiers and Validation respectively.

Dataset and Preprocessing

The fetch\_20newsgroups() function python SciKit Learn library was used to load the 20 newsgroup dataset. It’s convenient to use this method since it provides option to load either a training, testing or the complete dataset by default. Another useful option this function provides is that it allows to remove the metadata of the documents like header, quotes and footer to prevent the classifiers from overfitting to these features (Line 24-28, 31-35 in Cell 1).

After loading these documents, they were preprocessed to remove line breaks, carriage return and line feed tags and unnecessary whitespaces. Numeric values were also removed. (function - remove\_string\_special\_characters; Line 46-53 in Cell 1 ). To efficiently preprocess a large no. of documents, multiprocessing queues were used (Line 57-63 in Cell 1).

Features

The Features tried out for 20 newsgroups corpus were Bag of Words, N-grams, TFIDF, POS-tags and Headwords (Lemma). After extracting the features, the results were saved to text files named bag\_of\_words, n-grams, tfidf, POS-tags and HeadWords. All the code related to features are in Cell 1 of newsgroup\_classifier.ipynb

1. Bag of Words

Bag of Words Feature was extracted from the corpus using the CountVectorizer from SciKit Learn library. The CountVectorizer was used with analyzer set to word and filtering out english stopwords (Line 67 in Cell 1). It convert a collection of text documents to a matrix of token counts, in this case, a matrix of word counts.

1. N-grams

The n-grams which were extracted from the corpus were trigrams. To extract the trigrams, python Natural Language Toolkit(nltk) was used. Using nltk.util.ngrams() function (Line 90 in Cell 1), trigrams were extracted from the preprocessed documents.

1. TF IDF

The third feature extracted from the corpus was the TF IDF values. This was done in several steps. First, the preprocessed set of documents were passed to the get\_doc(s) function (Line 99 in Cell 1) which made a list of documents with their word counts using count\_words(s) (Line 109 in Cell 1) function. Then, by using create\_freq\_dicts(s) function (Line 116 in Cell 1), a list of frequency dictionaries were created. By feeding the created list of document lengths and the list of frequency dictionaries to computeTF(doc\_length\_list, freq\_dict\_list) (Line 134 in Cell 1), the Term Frequencies for each word in the list of documents were calculated. In the same manner the Inverse Document Frequencies of the documents were calculated from computeIDF(doc\_length\_list, freq\_dict\_list) function (Line 146 in Cell 1). Finally, these TF and IDF scores were passed to computeTFIDF(TF\_scores, IDF\_scores) (Line 160 in Cell 1), to calculate the TFIDF values.

1. POS-tags

The next feature to be extracted was POS-tags. First the documents were tokenized using nltk.tokenize.word\_tokenize() function (Line 192 in Cell 1). Then those tokens were passed through nltk.pos\_tag(s) (Line 193 in Cell 1) to obtain the POS-tagged words.

1. Headwords

The final feature to be extracted was headwords. This was done using nltk.WordNetLemmatizer(s) function (Line 203 in Cell 1). Just as in POS-tags, preprocessed documents were tokenized and then passed through WordNetLemmatizer function to obtain the Headwords (Lemma).

Classifiers

The Classifiers used for 20 newsgroup dataset were MultinomialNB Classifier, SGDClassifier and svm.SVC Classifier. All these classifiers were from SciKit Learn Library. All the code related to features are in Cell 2 of newsgroup\_classifier.ipynb

1. MultinomialNB Classifier

The first classifier tested was the MultinomialNB Classifier. It was implemented with using Tdidf values. For this pipeline class is used to create the workflow. SciKit Learn's Pipeline class is designed as a manageable way to apply a series of data transformations followed by the application of an estimator. The pipeline used for this classifier was built using CountVectorizer(), TfidfTransformer() and MultinomialNB() classes (Line 33 in Cell 2). Then the data from the newsgroup train dataset and their classes (targets) are used to fit the pipeline. This allows extraction of tokens with CountVectorizer() and transformation of those in to Tdidf values with TfidfTransformer() and finally training the MultinomialNB classifier with the transformed features (Line 40 in Cell 2). This pipeline is then used to predict the classifiers of the test data (Line 43 in Cell 2). Then the predicted classifiers are compared with the actual classifiers of the test dataset to measure the accuracy of the classifier (Line 46 in Cell 2). To obtain the precision, recall and f1 score of the documents of each category, metrics.classification\_report() function is used (Line 49 in Cell 2).

1. SGDClassifier

SGDClassifier was implemented in the same way as the MultinomialNB classifier. Only difference being that the text\_clf\_SGDClassifier pipeline containing a SGDClassifier instead of a MultinomialNB classifier (Line 58 in Cell 2). Measurement of accuracy and precision, recall and f1 score values were also done in a similar manner to MultinomialNB classifier.

1. svm.SVC Classifier

The methodology of implementing the svm.SVC classifier follows the same process used in implementing the previous two classifiers. In this instance, the pipeline was built using svm.SVC Classifier (Line 85 in Cell 2). Measurement of accuracy and precision, recall and f1 score values were also done in a similar manner to other two classifiers.

Evaluation

The evaluation was carried out using 10-fold cross validation. For this the entire 20 newsgroup dataset was used without dividing it into training and testing subsets (Line 21 in Cell 3). To create the 10-fold cross validation dataset, the data from the dataset was converted to an array using np.asarray(dataset.data). Then using sklearn.model\_selection.KFold() function (Line 35 in Cell 3), the dataset is split into 10 different sections. Nine of them is used to train the model and the last section is used as the test dataset. To prevent any bias in an individual section affecting the validation measurements, each different sector is used as the test dataset, while others train the model. This means 10 classifier pipelines are built, which are then trained with rest of folds and tested with each different fold (Line 46 in Cell 3). For cross validation of this dataset, a group of pipelines with CountVectorizer(), TfidfTransformer() and SGDClassifier() are used (Line 47 in Cell 3). These trained pipelines are then used to predict the class of test fold, and by comparing those predictions with actual class values of the documents in the test fold, accuracy (Line 60 in Cell 3), precision (Line 64 in Cell 3), recall (Line 67 in Cell 3) and f1 (Line 70 in Cell 3) scores are calculated for each fold as well as the mean values of these scores for the entire dataset (Line 84-88 in Cell 3). The functions used for calculating those values are accuracy\_score, precision\_score, precision\_score and recall\_score and f1\_score from sklearn.metrics.