CAP6776 Justin Johnson

Information Retrieval Fall 2017

Assignment 2

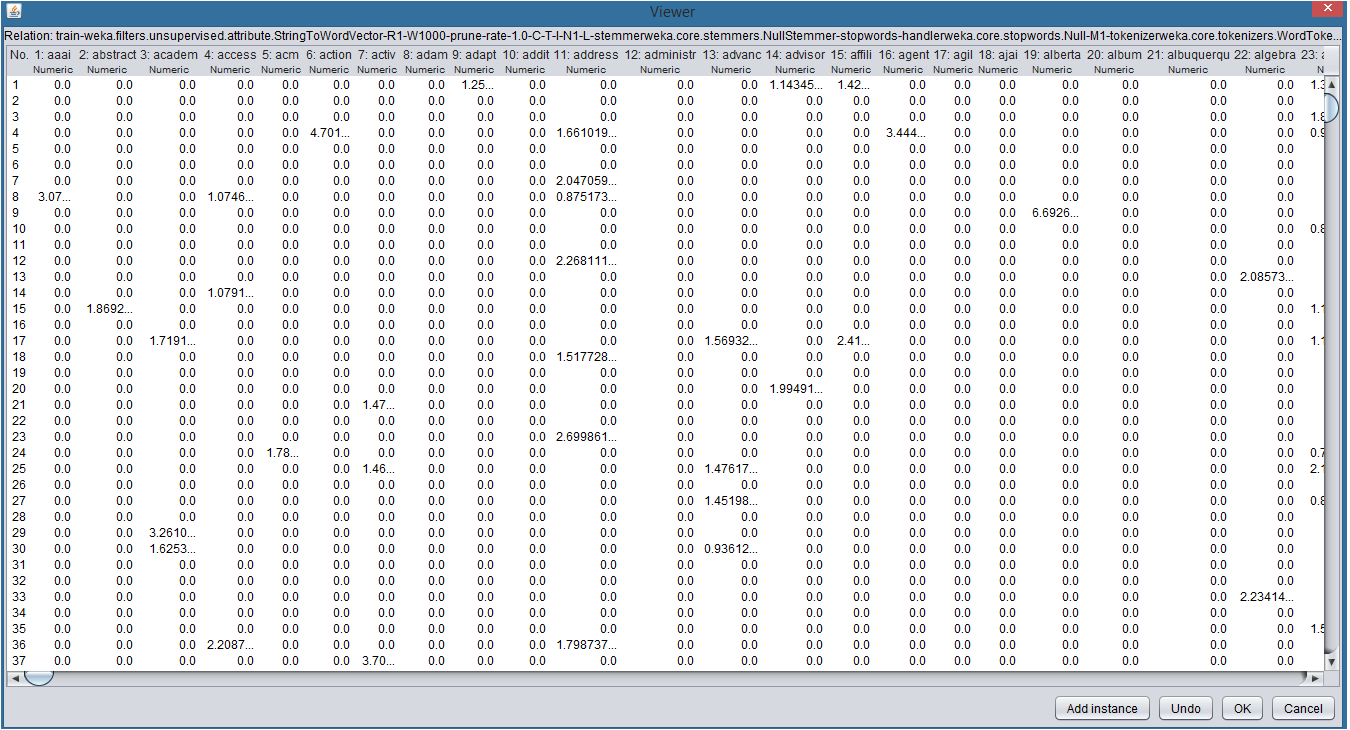
Using Weka to Calculate Document-Word Matrix & Perform Text Classification

**Part I) Preparing Data Set for Weka**

The WEKA Text Classification tool requires input data to be in the format of .arff files. .arff files contain a header that defines qualities of data set, such as class label options and the set’s relation. The included python script, generate-arff.py, reads the raw data set (training and test text files), and transforms the data into .arff format.

**Part II) Generating Document-Word TF-IDF Matrix**

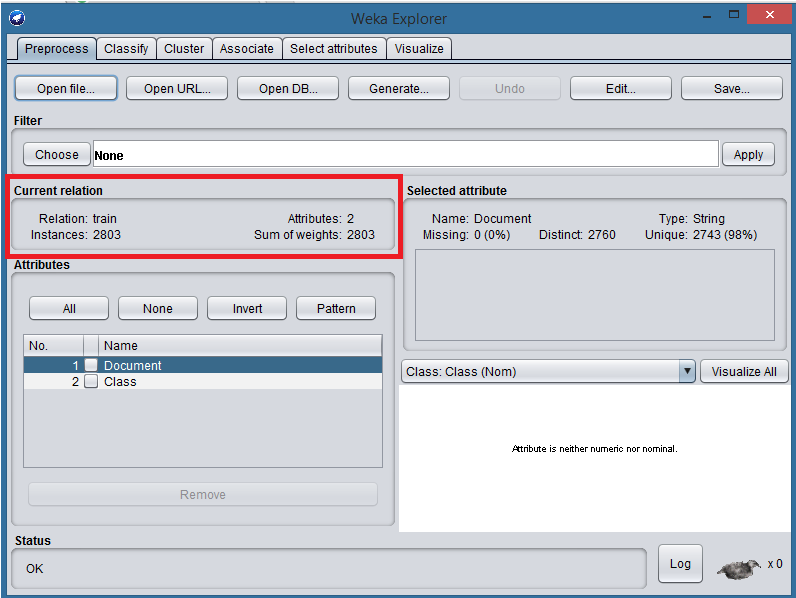
Weka’s StringToWordVector filter was utilized to construct a document-word matrix from the training data.



Screenshot provided displays small portion of document-word matrix generated by Weka. Filter options include tf-idf transformation and normalization.

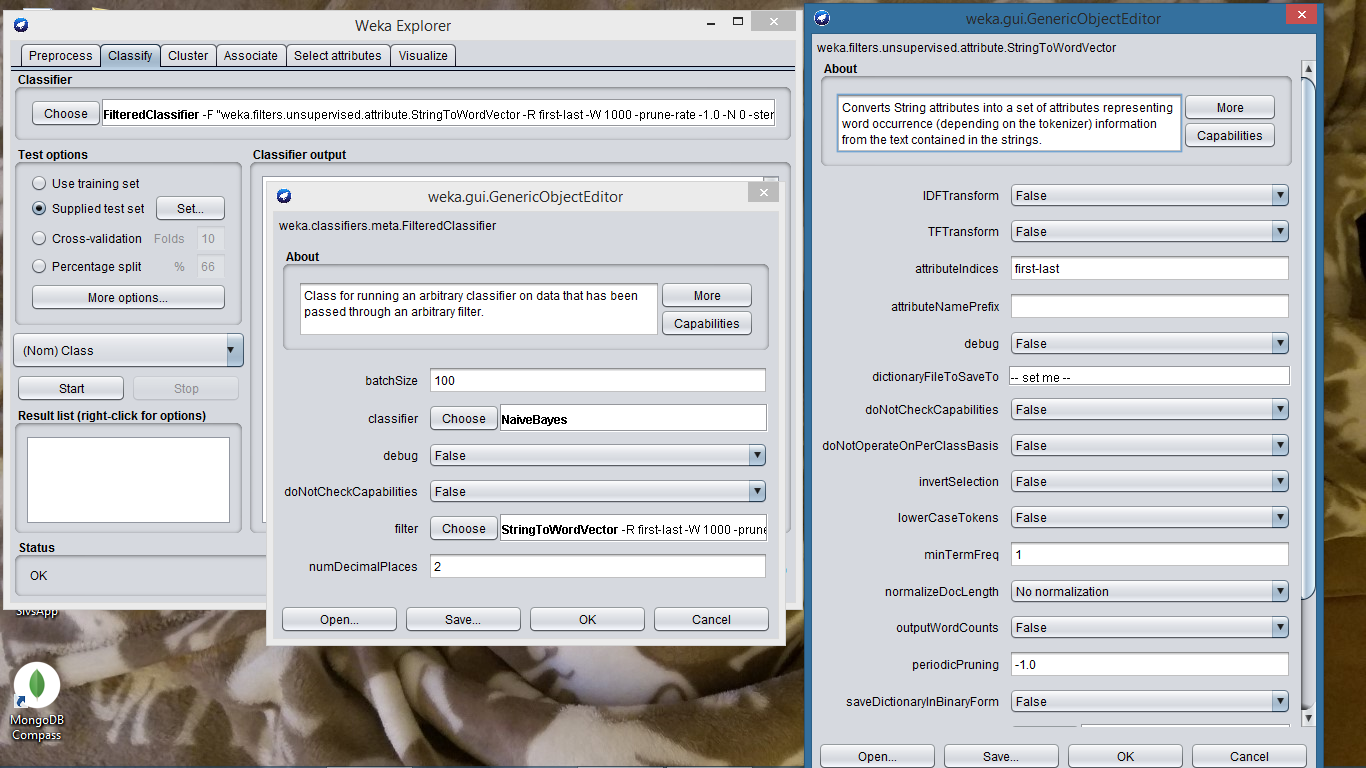
**Part III) Loading Training Data into Weka**

As visible in below screenshot, the training data containing 2803 samples has been loaded into Weka via the ‘Open file’ option along top toolbar. The test data was then loaded through the ‘Classifier’ tab’s ‘Supplied Test Set’ option.

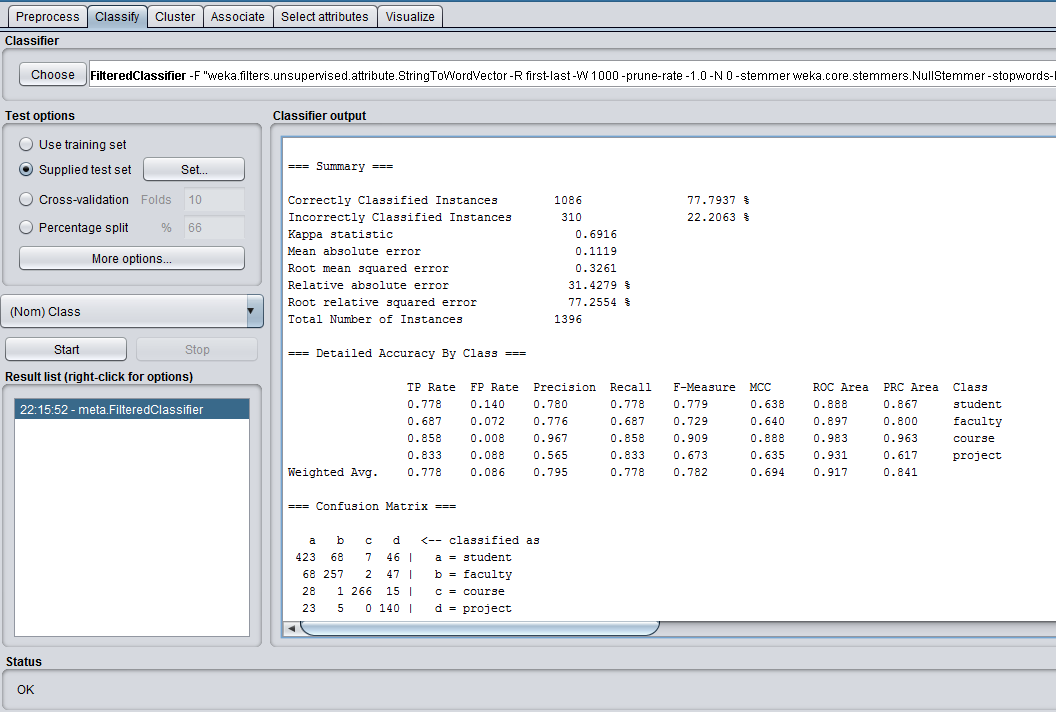


**Part IV) Classification with Naïve Bayes Classifier**

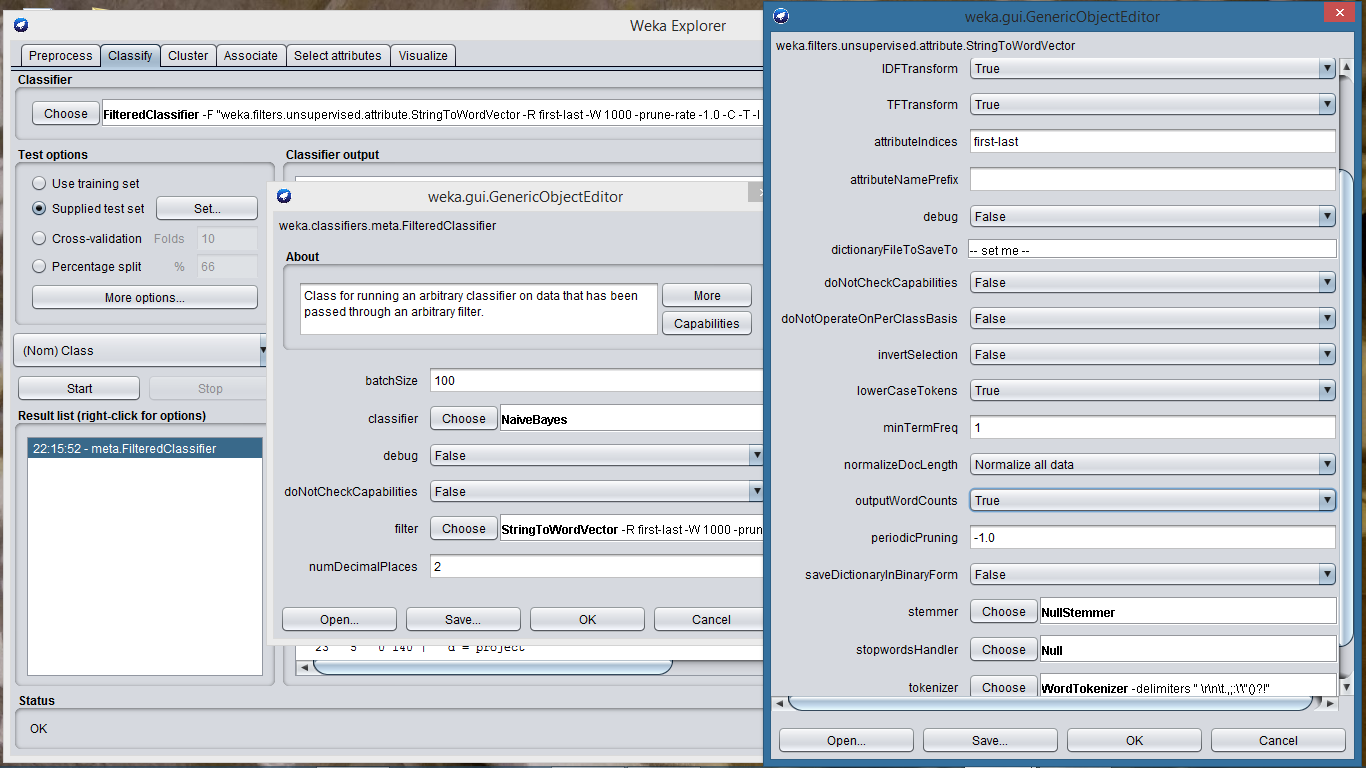
A) Default StringToWordVector filter options were used with Naïve Bayes Classifier. Below screen shot displays these default options.



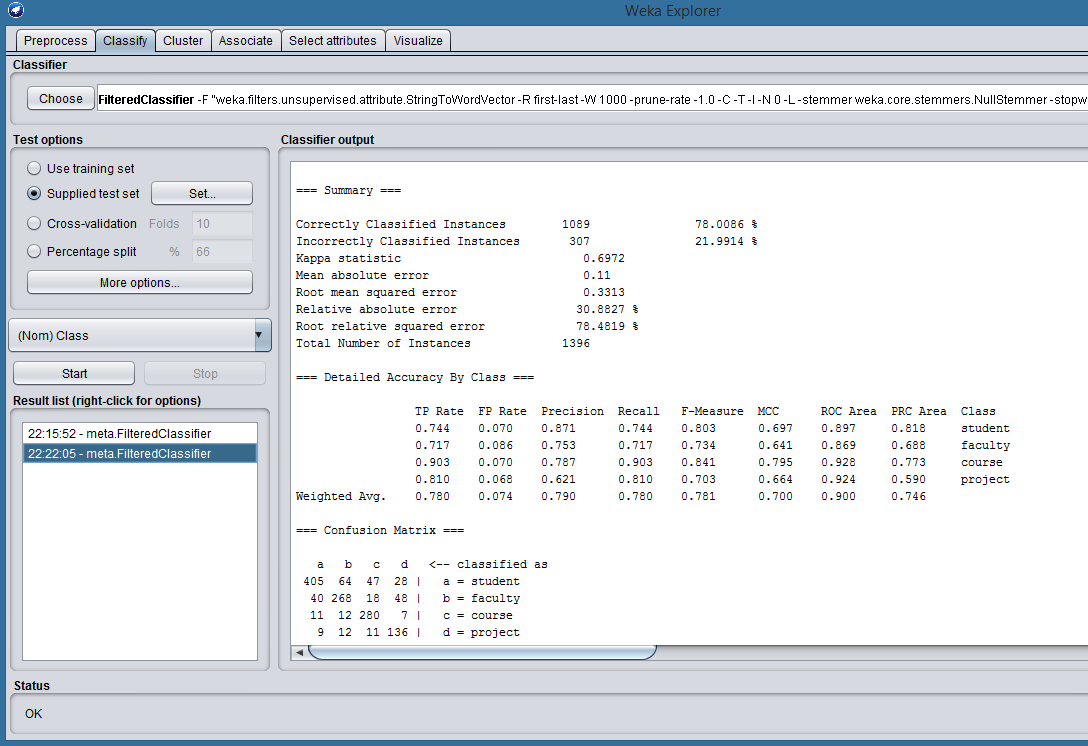
The default settings using training and test data accordingly with Naïve Bayes classifier resulted in accuracy of 77.79%, as displayed in below screenshot.



B) Custom StringToWordVector filter options were used with Naïve Bayes Classifier. Below screen shot displays these default options. These options include tf-idf transformation and normalization.

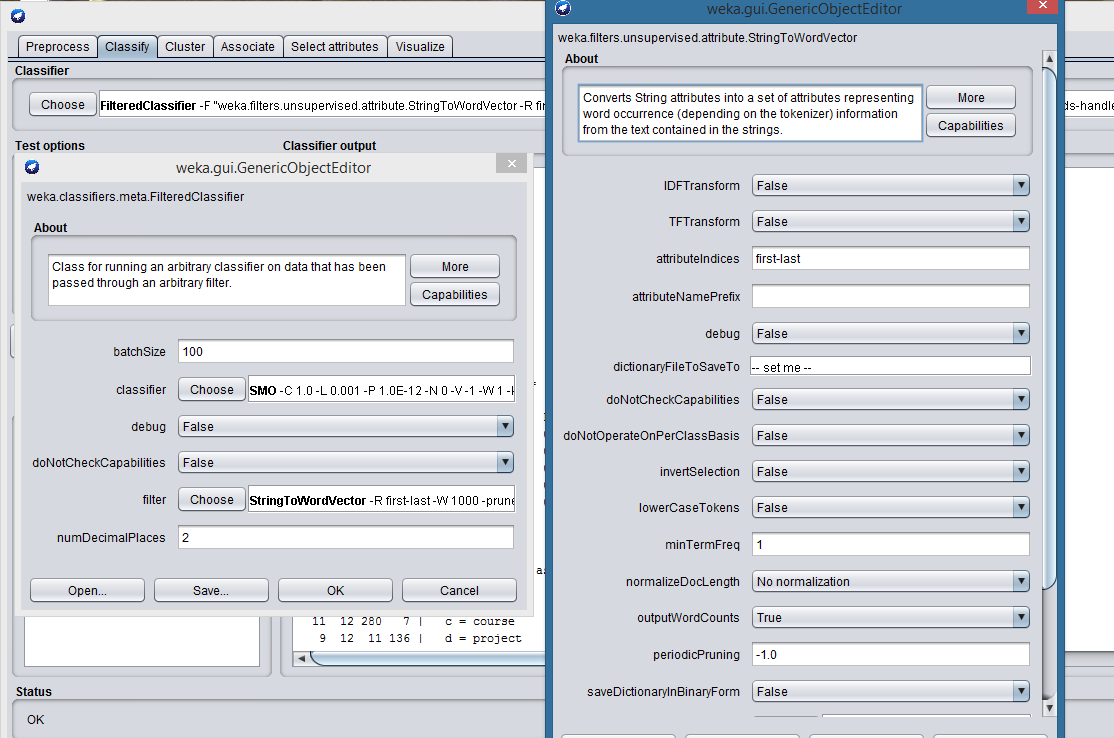


The custom settings using training and test data accordingly resulted in accuracy of 78.01%, as displayed in below screenshot. The Naïve Bayes classifier performed slightly better using normalized tf-idf, but the difference is not significant.

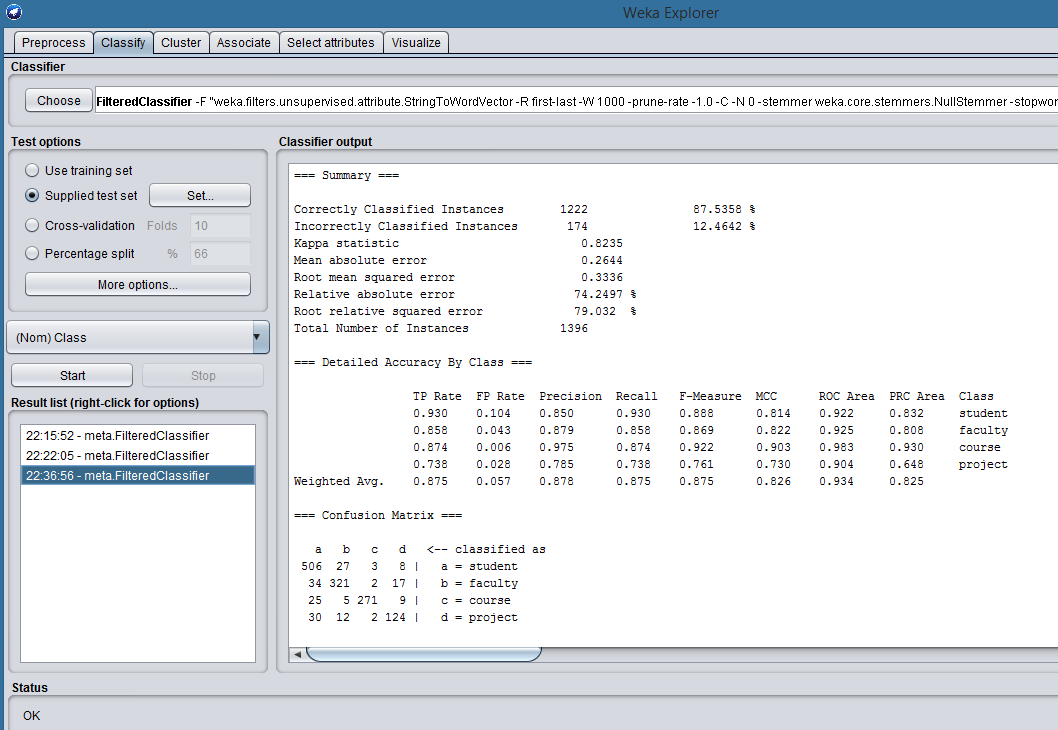


**Part V) Classification with SVM Classifier**

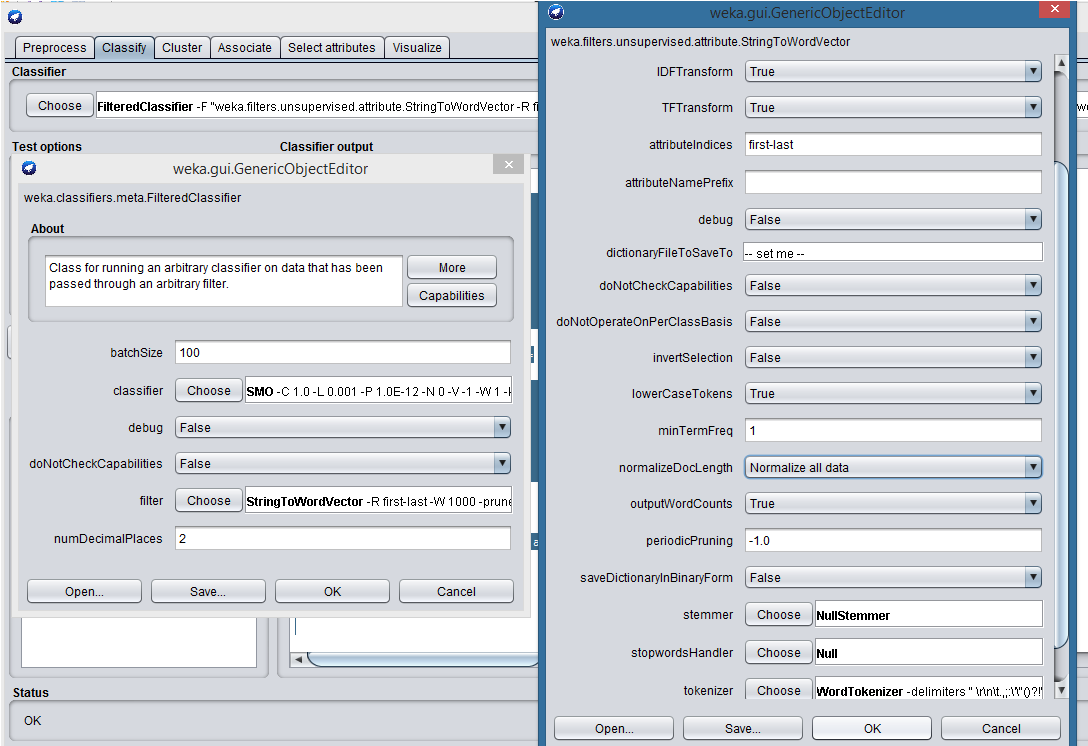
A) Default StringToWordVector filter options were used with SVM Classifier. Below screen shot displays these default options.



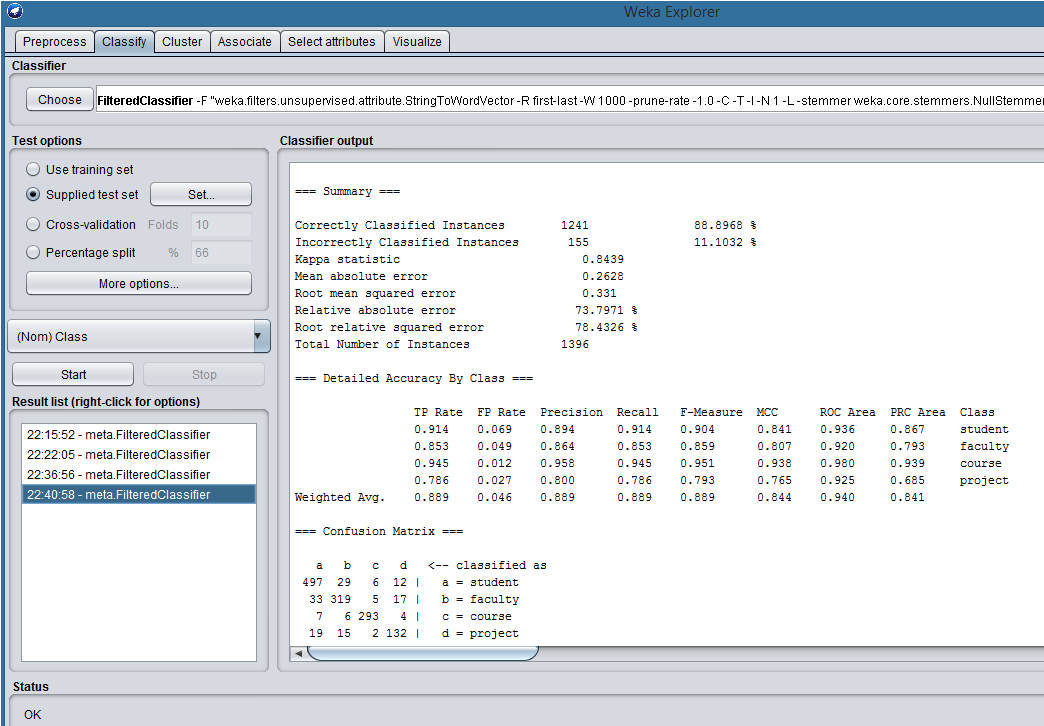
The default settings using training and test data accordingly with SVM resulted in accuracy of 87.54%, as displayed in below screenshot.



B) Custom StringToWordVector filter options were used with SVM Classifier. Below screen shot displays these default options. These options include tf-idf transformation and normalization.



The custom settings using training and test data accordingly with SVM classifier resulted in accuracy of 88.90%, as displayed in below screenshot. The SVM classifier performed significantly better using normalized tf-idf.



**Part VI) Conclusions**

|  |  |  |
| --- | --- | --- |
| **Classifier** | **StringToVector Filter Options** | **Accuracy** |
| Naïve Bayes | Default | 77.79% |
| Naïve Bayes | Custom (tf-idf + normalization) | 78.01% |
| SVM | Default | 87.54% |
| SVM | Custom (tf-idf + normalization) | 88.90%, |

The Naïve Bayes and SVM classifiers were trained and tested with Weka using the provided data set containing both training and test data. Both models were tested using the document-word matrix generated by Weka’s default StringToWordVector and the document-word matrix generated through custom StringToWordVector settings. The custom settings included a tf-idf transform and normalization.

The Support Vector Machine classifier out-performed the Naïve Bayes classifier, achieving an approximate 10% increase in test data classification accuracy.

For both classifiers, there was a slight increase in accuracy when using the custom StringToWordVector options that included tf-idf transform and normalization.