UNIVERSITY OF MANNHEIM

Data and Web Science Group
Prof. Dr. Christian Bizer
B6 – B1.15
68159 Mannheim

Data Mining – FSS 2021

Exercise 7: Text Mining

Note: You need to install the Text Mining Extension to be able to complete this exercise!

7.1. Which documents are similar?

1. The file documents.zip is provided in ILIAS and contains three corpora. Load and vectorize the 4-documents corpus into RapidMiner using the Process Documents from Files operator. How many different attributes has the generated example set?
   Solution: Setup the process. Include the *tokenize* operator within the nested *process documents from files* operator. Have a look at the Meta Data View on the ExampleSet Tab of your Result View.

   Conclusion: 994 regular attributes and 4 special attributes.

2. Examine the generated word list. What are the most common words? Look for the three most common words that might be helpful for text mining tasks?
   Solution: Have a look at the WordList Tab of your Result View. Order the words by the folder you selected in your *Process Documents from Files* operator.

   Conclusion: It's hard to find the most common word which would help to mine the text because the top words are so called stopwords. At position 30 you can find Madrid followed by United which may indicate a football game. At position 46 League is listed which underlines the first conclusion.

3. Remove stopwords and apply the porter stemmer. By how many attributes do the operators reduce the size of your example set?
   Solution: Add the stopwords filter operator and the stemmer (porter) operator within your nested *Process Documents from Files* operator.

   Conclusion: Number of attributes is reduced to 706 regular and 4 special attributes.

4. Use the *Data to Similarity* operator to generate a similarity matrix for the documents. Which documents are most similar? Can you confirm the judgment of the algorithm by reading the documents?
   Solution: Connect the *Data to Similarity* operator with your *Process Documents from Files* operator.

   Conclusion: Document 2 and 4 and document 3 and 4 are most similar (least distance).

5. Experiment with different similarity metrics as well as with different vector creation methods. Which combination produces the best similarity scores?
   Solution: The vector creation is setup within the *Process Documents from Files* operator and the similarity metrics are setup in the *Data to Similarity* Operator.

   Conclusion: The conclusion for 7.1.4 can be exposed more clearly using TF-IDF with CosineSimilarity or BinaryTermOccurences with Jaccard Similarity.

7.2. Cluster the 30-Documents Corpus

1. The 30-documents corpus contains postings from three news groups. Vectorize the 30-documents corpus, remove stopwords and maybe stem the corpus.
   Solution: Select all three included subfolders for the Process Documents from Files one by one to retrieve the classification information via this operator.

   Conclusion: After including stemming (porter) and stopwords filter, the data set has 2551 regular attributes.

2. Use the K-Means operator to cluster the corpus. Examine the resulting clustering using the folder view. How many documents ended up in the wrong cluster? What can you do to improve the clustering?
   Conclusion: Running the k-means clustering with standard setup for TF-IDF vectors all of the soc.religion documents are clustered correctly. The other two groups of documents are mixed up with 6 wrongly classified documents (2 from sci.space and 4 from talk.politics.guns). Without stemming the number of miss-clustered items is the same but there are wrong documents in each cluster also in soc.religion. Using the term frequency for the vector generation (with stemming(porter)) and filtering tokens which are smaller than length 5, the number of miss-clustered items is reduced to 5.

3. Examine the distribution of frequent words over the three different classes in the word list. Does the distribution give you an idea how you could improve the clustering using any of the prune methods of the process documents operator?
   Solution: Have a look at the table on the WordList Tab in your result view. Order the words first by total occurrences (desc). Than order the words by the occurrences within the document groups (desc).

   Conclusion: The words appearing more often are almost in all documents (between 20 and 30 document occurrences). Words that are significant for some documents appear less than 10 times within the documents. This is a hint to use e.g. absolute pruning. Using the ranges above 10 and below 4 the clustering miss-clusters only 2 documents.

7.3. Learn a Classifier for the 300-Documents Corpus

The 300-documents corpus contains postings from three different news groups. Vectorize the 300-documents corpus and learn a classifier for classifying the postings. Evaluate the classifying using 10-fold X-Validation. Which accuracy does your classifier reach? Increase the performance of your classifier by pruning the document vectors.

Conclusion: Without using any pruning or parameter adaption the process using a naïve bayes classifier with TF-IDF comes up with a accuracy of 79.33% (Recall: 79.33%, Precision: 81.79%).

Hint: check the boxes for “weighted mean recall” and “weighted mean precision” of the “Performance” operator to obtain Recall and Precision values.

Vector Creator	Prune below	Prune above	Accuracy (%)
TF-IDF	20	90	83.67
TF-IDF	20	70	82.67
Term Freq	20	110	84.00
B T O	10	100	86.33
B T O	0	130	91.00

But: Using Binary Term Occurences without any pruning an accuracy of 91.00% (Recall: 91.00%, Precision: 91.78%) is achieved.

7.4. Learn a Classifier for the Job Postings

1. The Job Postings corpus contains 500 descriptions of open positions belonging to 30 different job categories. The corpus is provided as an Excel file in ILIAS. Import the corpus into RapidMiner. Vectorize the corpus using the *Process Documents from Data* operator. Learn a Naïve Bayes classifier for classifying the job adds. Evaluate the classifying using 10-fold X-Validation. Analyze the classifier performance and the word list. What do you discover?
   Solution: Import the data and set both attributes to text. Set category as label and run the process. There will be several warnings which do not affect the process. If you want to remove those warnings you need to assign an Id to each example to use the operators properly. As there is no Id included in the data set you can use the Generate ID operator. Select only those attributes (id and job text) which are relevant for the vector creation. Join the data afterwards so that you get the label (category) back to your data set and run the validation.

   Conclusion: Running the process will lead to the following results: Accuracy: 48.51%, Recall: 27.04%, Precision: 24.61%. Within the wordlist are some HTML fragments which might be good to be filtered. Also the data set has a low number of text descriptions per class which might not be enough to learn a good classifier.

2. Experiment with different vector creation and pruning methods as well as different types of classifiers in order to increase the performance. What is highest accuracy that you can reach? Which problem concerning precision and recall does remain?
   Conclusion: Using Term Frequency and 0/28 pruning the classifier has an accuracy of 50.92%. Recall and precision are still really low: Recall: 30.28% and Precision: 28.37%