# Text Mining Project

Tadhg Cuddihy

## Automatic classification and clustering of biology documents

When designing an automatic classifier, the first thing to do is insect the data to ensure the following:

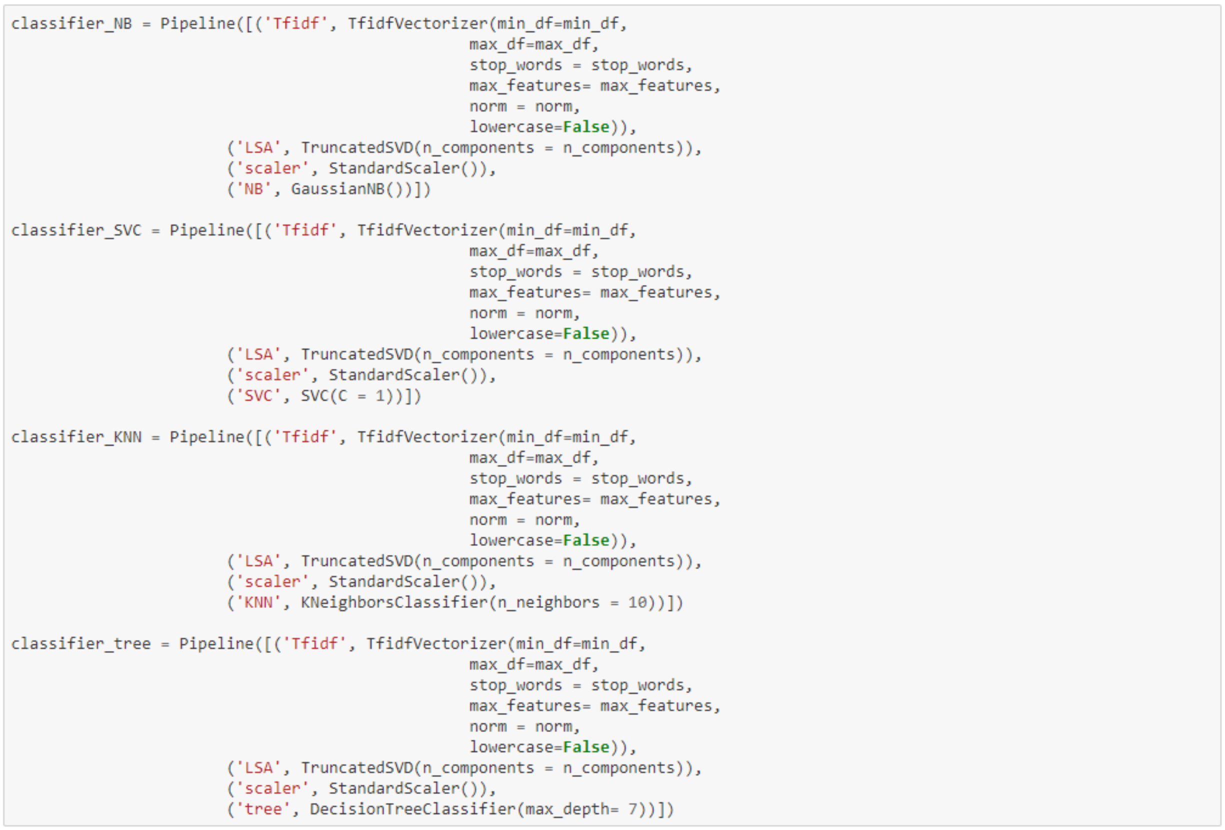
1. You understand how the data is represented
2. There is enough data to train with.
3. There is roughly the same amount of data for each label

When inspecting the datasets (Dataset 1 - Negative and positive abstracts) and (Dataset 2 - Alzheimer, Breast cancer, Bladder cancer, Cervical Cancer and non-relevant abstracts), the data set up with the following; Document number, Document Title, Abstract. The dataset 1 contains 8156 documents and dataset 2 contains 5000 documents, all documents are evenly split among the labels.

## Part I. Construction of an automatic classifier

Pre-processing is the first step in handling the data. The data is cleaned by converting all letters to lower case, removing stop words which provide no classifying formation for the classifier and splitting the documents into words. This is achieved by a simple regular expression shown in Figure 1 below

The data was then split into 70/30 train/test. The classifiers will be trained using the train data and the accuracy and performance of each classifier will be tested using the test data. For this problem, four classifiers are built in order to test which will give the best results. These can be seen in Figure 2 below.



```
classifier_NB = Pipeline([('Tfidf', TfidfVectorizer(min_df=min_df,
                                                  max_df=max_df,
                                                  stop_words = stop_words,
                                                  max_features= max_features,
                                                  norm = norm,
                                                  lowercase=False)),
                        ('LSA', TruncatedSVD(n_components = n_components)),
                        ('scaler', StandardScaler()),
                        ('NB', GaussianNB())])

classifier_SVC = Pipeline([('Tfidf', TfidfVectorizer(min_df=min_df,
                                                  max_df=max_df,
                                                  stop_words = stop_words,
                                                  max_features= max_features,
                                                  norm = norm,
                                                  lowercase=False)),
                        ('LSA', TruncatedSVD(n_components = n_components)),
                        ('scaler', StandardScaler()),
                        ('SVC', SVC(C = 1))])

classifier_KNN = Pipeline([('Tfidf', TfidfVectorizer(min_df=min_df,
                                                  max_df=max_df,
                                                  stop_words = stop_words,
                                                  max_features= max_features,
                                                  norm = norm,
                                                  lowercase=False)),
                        ('LSA', TruncatedSVD(n_components = n_components)),
                        ('scaler', StandardScaler()),
                        ('KNN', KNeighborsClassifier(n_neighbors = 10))])

classifier_tree = Pipeline([('Tfidf', TfidfVectorizer(min_df=min_df,
                                                  max_df=max_df,
                                                  stop_words = stop_words,
                                                  max_features= max_features,
                                                  norm = norm,
                                                  lowercase=False)),
                        ('LSA', TruncatedSVD(n_components = n_components)),
                        ('scaler', StandardScaler()),
                        ('tree', DecisionTreeClassifier(max_depth= 7))])
```

*Figure 1 - Pipeline for building the classifier*

# Text Mining Project

Tadhg Cuddihy

In order to test the effectiveness of each classifier, they will be compared to the default values of the classifiers. These are the values listed in the sklearn documentation. To try and maximize the quality of the classifiers, the following parameters were changed:

| **Parameter** | **Values** | | | |
|---|---|---|---|---|
| | **Default** | **1** | **2** | **3** |
| **TfidfVectorizer** | | | | |
| **stop_words** | english | None | | |
| **Max_features** | 200 | 500 | 2000 | 5000 |
| **min_df** | 0.2 | 0.05 | 0.15 | 0.2 |
| **max_df** | 0.8 | 0.3 | 0.6 | 0.9 |
| **norm** | l1 | l2 | None | |
| **LSA** | | | | |
| **n_components** | 20 | 10 | 30 | 50 |
| **LSA** | With LSA | No LSA | | |

*Figure 2 - Parameters for classifiers.*

*Note 1: The stop words were provided from nltk.corpus.*
*Note 2: When no LSA was used, scaler = StandardScaler(with_mean=False)*

The accuracy for each classifier was calculated using accuracy_score from sklearn.metrics. From the formula in figure 3, the fraction of true predictions divided by the number of samples equals the accuracy. The maximum score is 1 (perfect prediction) and the lowest is 0 (no correct predictions)

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_\text{samples}} \sum_{i=0}^{n_\text{samples}-1} 1(\hat{y}_i = y_i)$$

*Figure 3 - Accuracy score formula*

**The Results**

Below are the results for the two and five datasets. In figure 4, we can see from the default values, the best classifier is C-Support Vector Classification (SVC) with an accuracy value of 0.953, and the worst is K Nearest Neighbors (KNN) with an accuracy of 0.899. As the parameters change, there is no real change of results across the classifiers however, for the Decision Tree (tree) and Naive Bayes (NB) classifiers, they see the biggest change in accuracy when they are not dimensionality reduction using truncated SVD (LSA). This would indicate that when using these classifiers, you would be required to reduce dimensions to achieve the optimal results. The best scores for each classifier are highlighted in green with the overall best score being SVC min_df = 0.05.

The results for the 5-label dataset are less accurate but again the best classifier here is SVC with 0.8373 (min_df = 0.05). Looking specifically at the No-LSA result for KNN, at 0.4693, it is 36% lower than the best result indicating that now with 5 labels, dimension reduction is far more important than it was before in the 2-label data.

# Text Mining Project

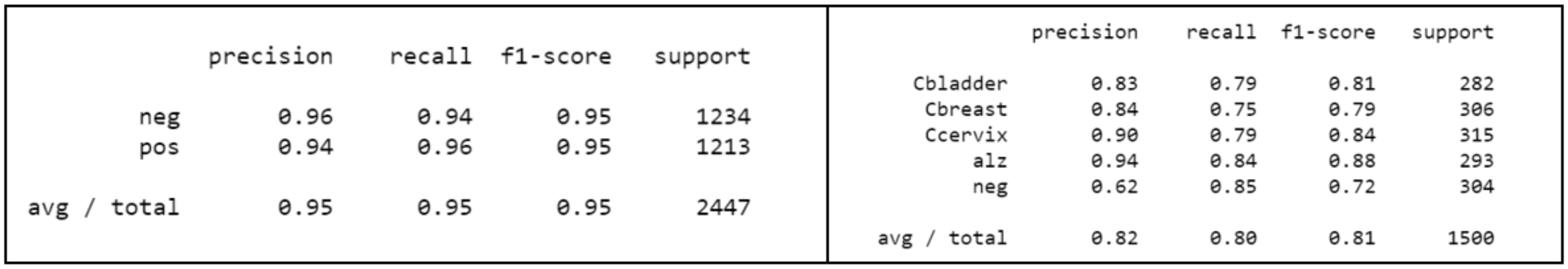
Tadhg Cuddihy

The classification report shows the fraction of true positive results for both datasets. We can see that there is a higher score (0.95) for the 2-label dataset compared to the 5-label dataset (0.82). The differences in the scores could be due to the fact that the 2-label dataset has more documents compared to the 5-label dataset. Potentially including more documents in the 5-label classifier could improve its ability to classify correctly.

| 2 label dataset | | | | | |
|---|---|---|---|---|---|
| **Parameter** | **Value** | **Tree** | **NB** | **SVC** | **KNN** |
| **TfidfVectorizer** | | | | | |
| **Default** | Default | 0.9428 | 0.9252 | 0.9526 | 0.8991 |
| **stop_words** | none | 0.9354 | 0.9273 | 0.9497 | 0.9027 |
| **Max_features** | 200 | 0.9367 | 0.9273 | 0.9555 | 0.9027 |
| | 500 | 0.9403 | 0.9260 | 0.9538 | 0.9003 |
| | 2000 | 0.9354 | 0.9252 | 0.9510 | 0.9044 |
| | 5000 | 0.9407 | 0.9256 | 0.9555 | 0.9036 |
| **min_df** | 0.05 | 0.9338 | 0.9256 | 0.9563 | 0.9064 |
| | 0.15 | 0.9362 | 0.9273 | 0.9530 | 0.9019 |
| | 0.2 | 0.9391 | 0.9244 | 0.9555 | 0.9011 |
| **max_df** | 0.3 | 0.9367 | 0.9289 | 0.9522 | 0.8966 |
| | 0.6 | 0.9412 | 0.9260 | 0.9546 | 0.9019 |
| | 0.9 | 0.9383 | 0.9236 | 0.9542 | 0.9040 |
| **norm** | L2 | 0.9399 | 0.9252 | 0.9522 | 0.8987 |
| | none | 0.9403 | 0.9268 | 0.9534 | 0.9031 |
| **LSA** | | | | | |
| **n_components** | 10 | 0.9371 | 0.9264 | 0.9534 | 0.8999 |
| | 30 | 0.9350 | 0.9224 | 0.9522 | 0.9011 |
| | 50 | 0.9395 | 0.9256 | 0.9510 | 0.9064 |
| **no_LSA** | removed | 0.9125 | 0.9040 | 0.9363 | 0.9093 |

| 5 label dataset | | | | | |
|---|---|---|---|---|---|
| **Parameter** | **Value** | **Tree** | **NB** | **SVC** | **KNN** |
| **TfidfVectorizer** | | | | | |
| **Default** | Default | 0.6687 | 0.7307 | 0.8340 | 0.7333 |
| **stop_words** | none | 0.6633 | 0.7353 | 0.8333 | 0.7320 |
| **max_features** | 500 | 0.6760 | 0.7347 | 0.8340 | 0.7293 |
| | 2000 | 0.6660 | 0.7307 | 0.8333 | 0.7320 |
| | 5000 | 0.6787 | 0.7327 | 0.8353 | 0.7353 |
| **min_df** | 0.05 | 0.6500 | 0.7393 | 0.8373 | 0.7340 |
| | 0.15 | 0.6640 | 0.7320 | 0.8347 | 0.7333 |
| | 0.2 | 0.6833 | 0.7307 | 0.8327 | 0.7293 |
| **max_df** | 0.3 | 0.6653 | 0.7400 | 0.8327 | 0.7360 |
| | 0.6 | 0.6567 | 0.7300 | 0.8360 | 0.7273 |
| | 0.9 | 0.6600 | 0.7313 | 0.8347 | 0.7300 |
| **norm** | L2 | 0.6640 | 0.7333 | 0.8353 | 0.7300 |
| | none | 0.6940 | 0.7320 | 0.8333 | 0.7353 |
| **LSA** | | | | | |
| **n_components** | 10 | 0.6600 | 0.7367 | 0.8367 | 0.7333 |
| | 30 | 0.6653 | 0.7300 | 0.8340 | 0.7353 |
| | 50 | 0.6927 | 0.7287 | 0.8293 | 0.7307 |
| **no_LSA** | removed | 0.8900 | 0.7287 | 0.8033 | 0.4693 |

*Figure 4 - Scores for 2 label dataset (left)-and 5 label dataset (right). Best scores are highlighted in green*



```
             precision    recall  f1-score   support

        neg       0.96      0.94      0.95      1234
        pos       0.94      0.96      0.95      1213

avg / total       0.95      0.95      0.95      2447
```

```
             precision    recall  f1-score   support

   Cbladder       0.83      0.79      0.81       282
    Cbreast       0.84      0.75      0.79       306
    Ccervix       0.90      0.79      0.84       315
        alz       0.94      0.84      0.88       293
        neg       0.62      0.85      0.72       304

avg / total       0.82      0.80      0.81      1500
```

*Figure 5 - Classification score for 2 label dataset (left)-and 5 label dataset (right). Classifier: SVC, Parameters: Best from figure 4*

## Tips for construction an automatic classifier:

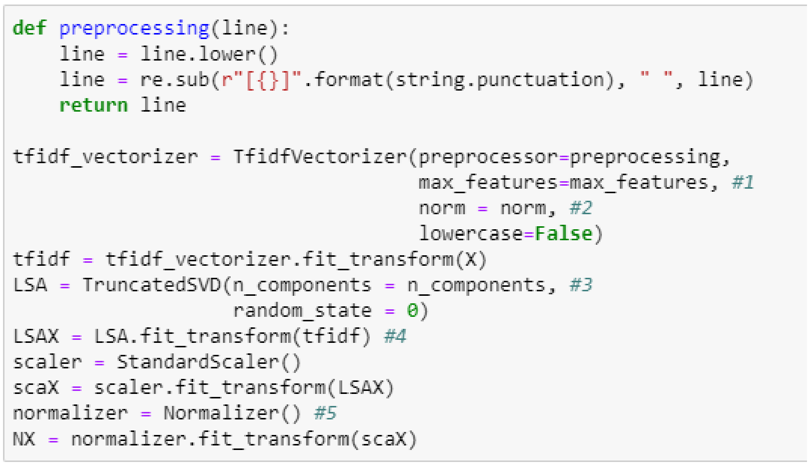
From both datasets, the min_df gave the best results for SVC. The stop_words and max_features give similar results regardless of the change. To further tune the classifiers, I would continue to look into getting the optimal min_df.

Choosing the parameters for the classifier depends a lot on the type of data you are working with. As both datasets were different, simply choosing parameters that worked for one dataset may not work well for another.  Therefore, I would recommend analyzing the classification problem first before jumping into building the classifier. Although SVC gave the best scores in both cases, there could be a better classifier suited for the 5-label dataset.

## Part II. Construction of a clustering of biology documents

This is a slightly different problem from above. We already know the classes for the datasets but this time we want to see how well the clustering program can group the documents together without knowing the class. For this problem, we will not use the class in training the program but we will call it later to determine the accuracy of the clustering. The data will also not be split into training and test. As we are performing unsupervised clustering, all the data will be used in developing the clustering program.

Below is the code used for building the clustering program. The number of components changes depending if we are working with the 2 or 5 label datasets. In order to change the vocabulary size, the number of features was altered.



```
def preprocessing(line):
    line = line.lower()
    line = re.sub(r"[{}]".format(string.punctuation), " ", line)
    return line

tfidf_vectorizer = TfidfVectorizer(preprocessor=preprocessing,
                                   max_features=max_features, #1
                                   norm = norm, #2
                                   lowercase=False)
tfidf = tfidf_vectorizer.fit_transform(X)
LSA = TruncatedSVD(n_components = n_components, #3
                   random_state = 0)
LSAX = LSA.fit_transform(tfidf) #4
scaler = StandardScaler()
scaX = scaler.fit_transform(LSAX)
normalizer = Normalizer() #5
NX = normalizer.fit_transform(scaX)
```

*Figure 6 – Code for building the clustering program*

| Parameter | Values | | | |
|---|---|---|---|---|
| | Default | 1 | 2 | 3 |
| **TfidfVectorizer** | | | | |
| **stop_words** | english | None | | |
| **Max_features** | 500 | 2000 | 5000 | 10000 |
| **norm** | l1 | l2 | None | |
| **LSA** | | | | |
| **n_components** | 20 | 10 | 30 | 50 |
| **LSA** | With LSA | No LSA | | |

*Figure 7 - Parameters for clustering.*

*Note 3: The stop words were provided from nltk.corpus.*
*Note 4: When no LSA was used, scaler = StandardScaler(with_mean=False)*

# Text Mining Project

Tadhg Cuddihy

**What is K-means?**

Form the sklearn documentation on K-means, this algorithm aims to find the centroid to a predetermined number of clusters. With this centroid, it will cluster all the points are nearest to it. The centroid is chosen in order to minimize the within-cluster sum of squares criterion. This is given by the formula below:

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2)$$

*Figure 8 - Formula used in K-means*

K-means does have limitations in its use.

1. Data must have no missing values. If there is missing data in the dataset, it should either be removed entirely or estimated based on the data from the dataset.
2. K-means responds poorly to elongated clusters or clusters with irregular shapes
3. Depending where the initialization of the first centroid is, K-means can converge on different locations, thus giving different clusters. A method to address this is the use of the parameter init='k-means++'. This tries to ensure that the centroids are always initialized at a distance from each other rather than randomly initializing them which could place them close together.

**What is Ward?**

Ward is a method of hierarchical clustering. This type of clustering aims to create a tree with the cluster as the root and the leaves as the unique samples. Ward is a type of merge strategy that aims to minimize the sum of the squared difference within all clusters. Similar to K-means, this is a variance minimizing approach. This type of hierarchical clustering also outputs a tree where the distribution of the documents can be seen.

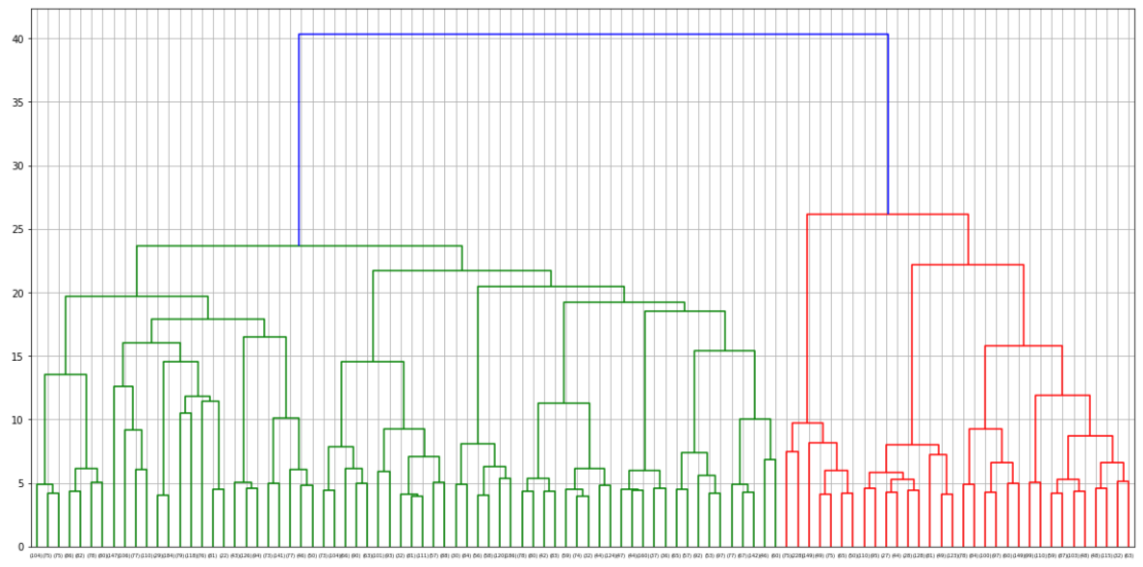


*Figure 9 - Example of dendrogram from hierarchical clustering*

# Text Mining Project

Tadhg Cuddihy

**The results:**

Figure 8 below represents the results for the clustering using two different algorithms. K-means and Ward. In a perfect scenario, the clustering would be Cluster0 = 50 pos and Cluster1 = 50 neg (or vice versa).

For K-means, the results are fairly consistent across all the parameter changes, with No LSA giving the best result. When looking at the Ward results, these are less successful. Ward appears to group the majority of the documents into one cluster every time, with the best results for no normalization. Without the prior labels to train the clustering program, it has a harder time to achieve good results compared to the classifier above.

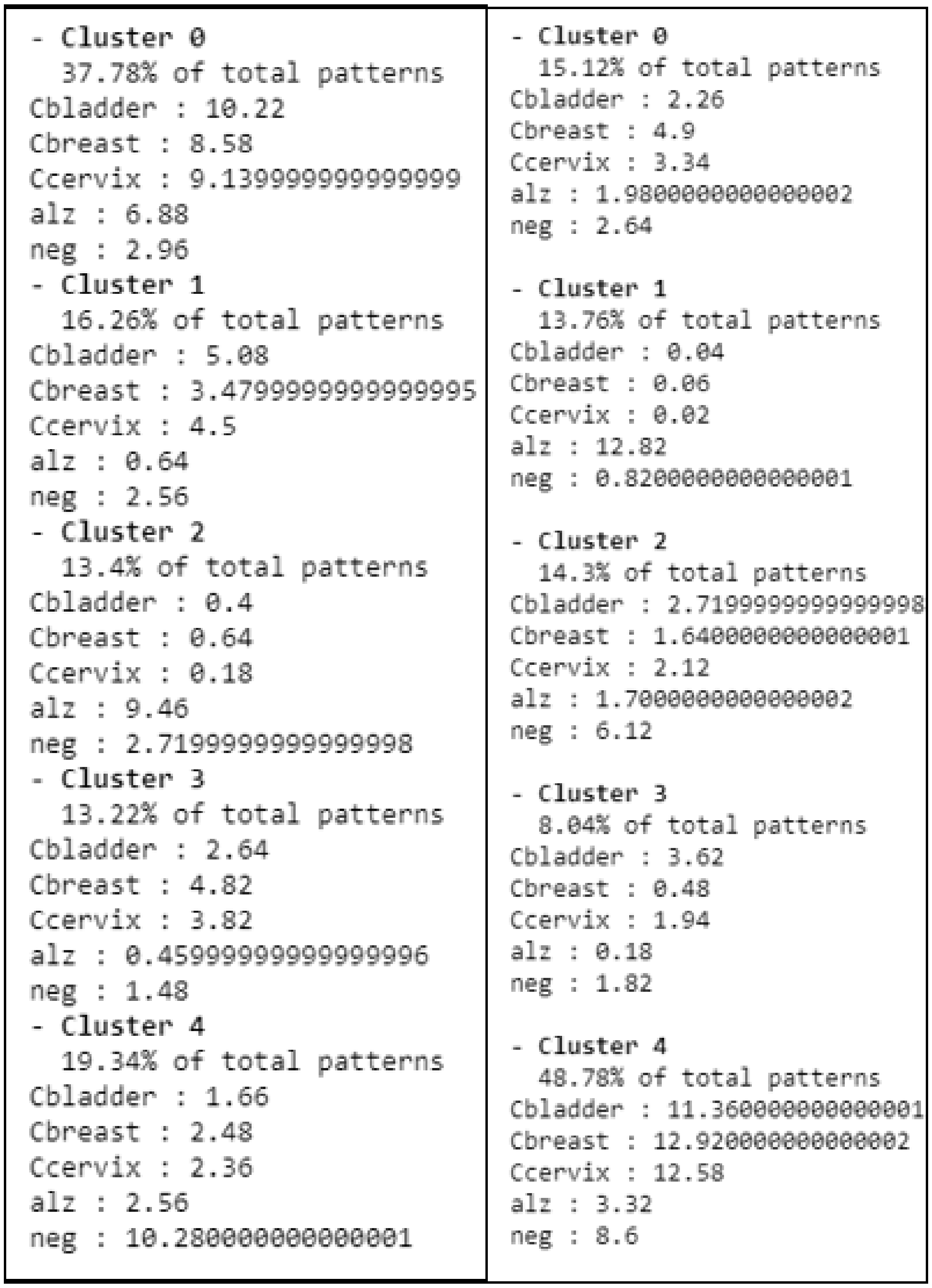
| **Clustering** | | | | | |
|---|---|---|---|---|---|
| **2 label dataset** | | | | | |
| | | **K means** | | **Ward** | |
| **Parameter** | **Value** | **Cluster 0** | **Cluster 1** | **Cluster 0** | **Cluster 1** |
| **TfidfVectorizer** | | | | | |
| **Default** | | 7.4301 | 42.5699 | 13.7200 | 36.2800 |
| | Default | 49.8406 | 0.1594 | 49.5954 | 0.4046 |
| **Max_features** | | 7.4792 | 42.5208 | 17.3982 | 32.6018 |
| | 500 | 49.8283 | 0.1717 | 49.8038 | 0.1962 |
| | | 42.4105 | 7.5895 | 22.6336 | 27.3664 |
| | 2000 | 0.1226 | 49.8774 | 0.0245 | 49.9755 |
| | | 7.3811 | 42.6189 | 13.0947 | 36.9053 |
| | 5000 | 49.8651 | 0.1349 | 49.8038 | 0.1962 |
| **Normalization** | | 8.3619 | 41.6381 | 41.0741 | 8.9259 |
| | None | 49.5586 | 0.4414 | 2.0721 | 47.9279 |
| **norm** | | 7.4792 | 42.5208 | 15.9392 | 34.0608 |
| | L2 | 49.8529 | 0.1471 | 49.6935 | 0.3065 |
| | | 7.4792 | 42.5208 | 15.9392 | 34.0608 |
| | None | 49.8529 | 0.1471 | 49.6935 | 0.3065 |
| **LSA** | | | | | |
| **n_components** | | 7.1849 | 42.8151 | 12.9475 | 37.0525 |
| | 10 | 49.8406 | 0.1594 | 49.1663 | 0.8337 |
| | | 7.8225 | 42.1775 | 15.5346 | 34.4654 |
| | 30 | 49.8529 | 0.1471 | 0.0736 | 49.9264 |
| | | 7.9819 | 42.0181 | 19.7401 | 30.2599 |
| | 50 | 49.8529 | 0.1471 | 49.3747 | 0.6253 |
| **no_LSA** | | 7.9819 | 42.0181 | NA | NA |
| | None | 49.8529 | 0.1471 | NA | NA |

*Figure 10 - Scores for 2 label dataset (left)-and 5 label dataset (right). Negative documents are highlighted in yellow, positive documents are in white. Cells with NA could not attain a result*

## Text Mining Project

Tadhg Cuddihy

The results for the dataset with 5 labels were worse than those compared to 2 labels. Below is an example of the output. In the majority of clusters, there is no clear dominance of any particular document type. This shows the difficulty the clustering algorithm had in separating the documents by type.



```
- Cluster 0
  37.78% of total patterns
Cbladder : 10.22
Cbreast : 8.58
Ccervix : 9.139999999999999
alz : 6.88
neg : 2.96
- Cluster 1
  16.26% of total patterns
Cbladder : 5.08
Cbreast : 3.4799999999999995
Ccervix : 4.5
alz : 0.64
neg : 2.56
- Cluster 2
  13.4% of total patterns
Cbladder : 0.4
Cbreast : 0.64
Ccervix : 0.18
alz : 9.46
neg : 2.7199999999999998
- Cluster 3
  13.22% of total patterns
Cbladder : 2.64
Cbreast : 4.82
Ccervix : 3.82
alz : 0.45999999999999996
neg : 1.48
- Cluster 4
  19.34% of total patterns
Cbladder : 1.66
Cbreast : 2.48
Ccervix : 2.36
alz : 2.56
neg : 10.280000000000001
```

```
- Cluster 0
  15.12% of total patterns
Cbladder : 2.26
Cbreast : 4.9
Ccervix : 3.34
alz : 1.9800000000000002
neg : 2.64

- Cluster 1
  13.76% of total patterns
Cbladder : 0.04
Cbreast : 0.06
Ccervix : 0.02
alz : 12.82
neg : 0.8200000000000001

- Cluster 2
  14.3% of total patterns
Cbladder : 2.7199999999999998
Cbreast : 1.6400000000000001
Ccervix : 2.12
alz : 1.7000000000000002
neg : 6.12

- Cluster 3
  8.04% of total patterns
Cbladder : 3.62
Cbreast : 0.48
Ccervix : 1.94
alz : 0.18
neg : 1.82

- Cluster 4
  48.78% of total patterns
Cbladder : 11.360000000000001
Cbreast : 12.920000000000002
Ccervix : 12.58
alz : 3.32
neg : 8.6
```

*Figure 4 - Results for 5-label dataset. K-means(left) Hierarchical clustering-ward (right)*