Honours Degree in Computing

Text Analytics Assessment:
Analyse an Unstructured dataset

Submitted by: Stephen Blaney, B00076157

Submission date 21^{h} March 2018

Declaration

Name: Stephen Blaney Dated: 05/03/2018

Table of Contents

1. Collect Text and Identify Business Objective 5
  1.1 Business objective 5
  1.2 Data Mining objective 5
  1.3 Text visualisations 6
    1.3.1 Winter Olympics word cloud 6
    1.3.2 Summer Olympics word cloud 6
    1.3.3 Horror reviews 7
2. Text Pre-processing 7
  2.1 Tokenizers 7
  2.2 Filters 8
    2.2.1 Filter token by length 8
    2.2.2 Custom stop words (Dictionary) 8
    2.2.3 filter token (By POS) 9
  2.3 Stemmers 10
    2.3.1 Porters 10
    2.3.2 Lovins 10
    2.3.3 Comparison between porters and lovins 11
    2.3.4 Stemmer (dictionary) 11
  2.4 Pruning 12
  2.5 Final Bag of words and processes 13
3. Feature selection statistics /counting 14
  3.1 Binary document vectors 14
  3.2 Term Occurrence 15
  3.3 TF-IDF 16
4.Data Mining 16
  4.1 Clustering 16
    4.1.1 K-means 16
    4.1.2 Agglomerative clustering 18
    4.1.3 Self-organising map (SOM) 20
    4.1.4. Clustering Evaluation 21
    4.1.5 Davies Bouldin Objective measure 22
    4.1.6 Conclusion 23

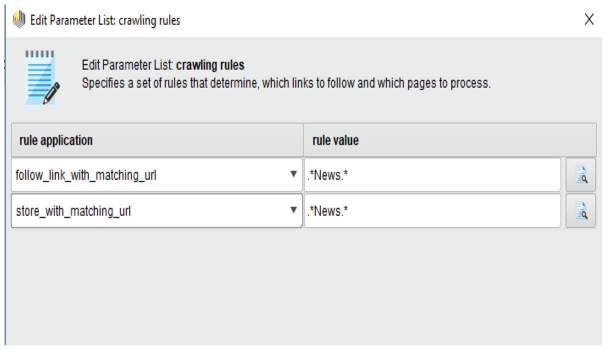
4.2 Classification- Supervised Learning 23
4.2.1 Decision tree 24
4.2.2 K-NN 24
4.2.3 Naïve Bayes 25
4.3 Dimensionality reduction 26
4.3.1 Attribute weighting 26
4.3.2 Attribute reduction via Principal component analysis (PCA) 30
4.4Conclusion 30
5. Analysing Results and Conclusion 31
5.1 Receiver Operator Characteristic 31
5.2 Conclusion 31
6. Appendix 32

1. Collect Text and Identify Business Objective

This dataset was sourced from 30 different text files divided into 3 categories Summer Olympics, Winter Olympics and horror reviews. Some of these text files where gathered using a web crawler (9 in total) these crawlers where also tailored using regular expressions to specify which text we wanted the crawler to capture. This dataset will be tested using 9 different text files (3 from each category) during classification to see if a correct predication can be made from the training data.

Both the winter and summer Olympic texts where sourced from the same Olympic website https://www.olympic.org/. Several new articles from the current winter games in PyeongChang and 2016 summer games in Rio where taken. The horror reviews where all taken from the top 100 horror reviews on rotten tomatoes. https://www.rottentomatoes.com/top/bestofrt/top_100_horror_movies/

The experience of using a web crawler as a method of extracting text was very tedious as a lot of extra information was captured that was useless such as navigation menus, headers and footers. However, creating a topical crawler was useful to download specific pages on particular themes or topics. In this case I created a regular expression that downloaded only news articles from the main Olympics website (Figure 1).



Edit Parameter List: **crawling rules**
Specifies a set of rules that determine, which links to follow and which pages to process.

rule application	rule value
follow_link_with_matching_url	.*News.*
store_with_matching_url	.*News.*

Figure 1: Rule value for topical crawler

1.1 Business objective

The purpose of the project is to produce high-quality information from each of the 3 categories of text, this will be done by detecting patterns and trends within the dataset.

1.2 Data Mining objective

Produce both classification and clustering model to predict which words are typical of each of the text and to group these texts from the collection of documents by using a similarity function.

1.3 Text visualisations

In order to aid the prepressing stage of this assessment its necessary to create word clouds for each of the 3 categories of texts in order to identify stop words (words too common to be of value) occurring terms and potential synonyms (Words that have the same meaning).

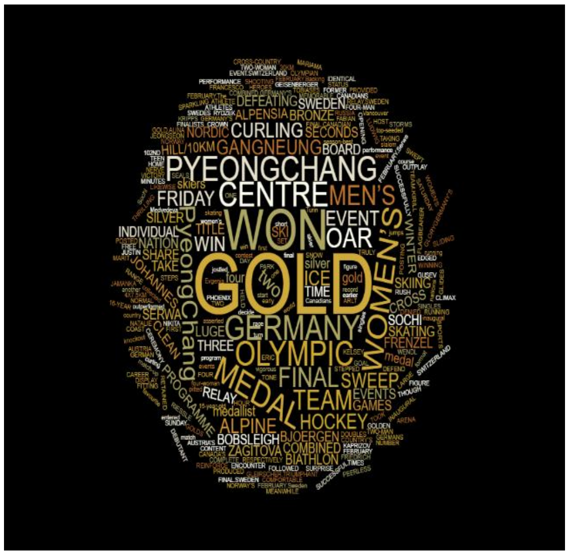


Figure 1.3.1 Winter Olympics word cloud

1.3.1 Winter Olympics word cloud

The most commom words in this cloud apperas to be Gold, Won , Pyeongchang Womens and Germany. All these words makes perfect sense to the domain we are exploring. Like all text minning projects there will be useless stops words within them e.g. two, four, the etc. There is quite an amount of synonyms such as the names of the athletes and words such as awards and medals.

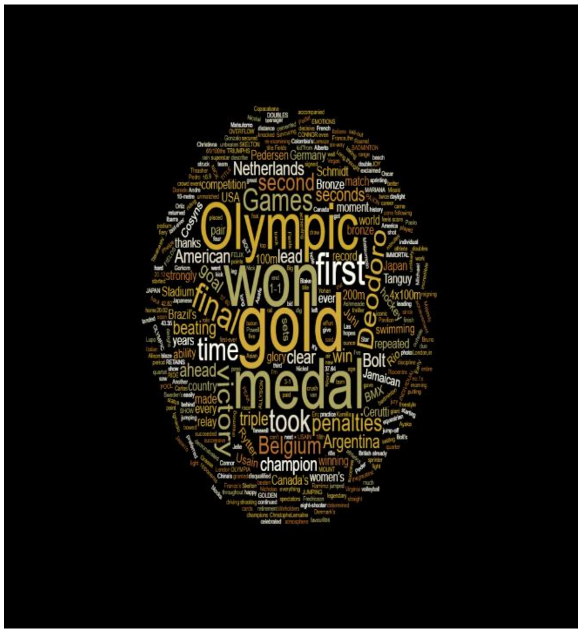


Figure 1.3.2 Summer Olympics word cloud

1.3.2 Summer Olympics word cloud

Very similar to the Winter Olympics in terms of certain words like Olympic medal, gold and final also appear to be very common. This is exactly what we want as it shows by simply looking at the cloud that there a correlation between the two Olympics. Like the first world cloud the amount of stop words such as a, the, and had will need to be addressed. Synonyms also appear to be the same with several players name e.g. Bolt and Phelps appear a bit. The summer games also have double the amount of sports in comparison to winter games

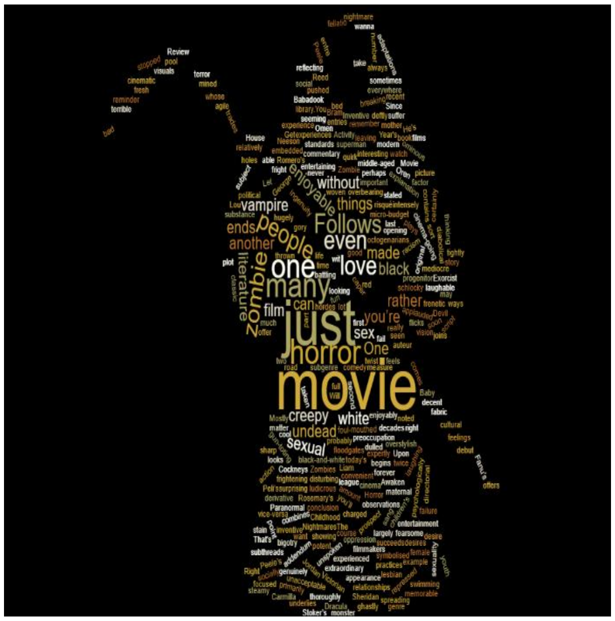


Figure 1.3.3 Horror Review word cloud

1.3.3 Horror reviews

This is the far cry to are other texts. This time we have words and phrase that are only unique to this word cloud. The most common being, Movie, Horror and Zombie. There seems to be a lot more stops words present in this world cloud as the word "Just" appears 7 times as well as being sprinkled with other words such as even, can and you'll. Synonyms identified here are words such as filmmaker and director along with zombie and zombies

2. Text Pre-processing

The first step in mining text is to organise the training files into folders and call them in using the operator process document from files and configure the edit list parameters to read the directories. Imbedded within in this process contains all the various tokenise /n-grams, filtering and stemmers specifically in that order as we cannot do this in any order we want. This will be used to construct our bag of words.

2.1 Tokenizers

In RapidMiner you are offered a choice of three operators in order to break up your text into words this is typical the first operator use in your pre-processing after reading in a document.

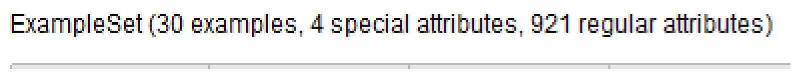
1. Standard string tokenizer this is what was eventually used in this project as it's easy to use and simple by just splitting the words into tokens.

2. N-gram (character) doesn't separate words into tokens instead predefined lengths of substrings are used. This operator was not chosen based on the performance issues it suffers from. Also, the quality of our texts is quite rich, not suffering from any spelling errors therefore no need to enable flexible matching using N-grams.

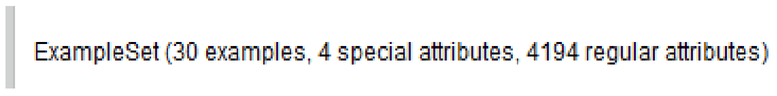
3. N-gram(term) this operator combines words into phrase in our case an example would be gold_medal this is used in conjunction with the string tokenizer. Can

be useful in sentiment analysis for example in horror reviews to pair terms like very_good and quite_bad.

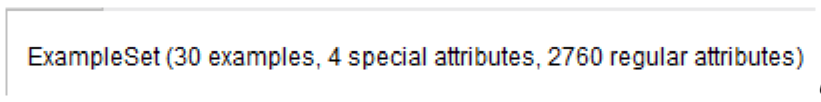
The problem with the N-grams is that in increases our bag of words exponentially as the following screenshots will illustrate. Our baseline for our attributes at this point in the project without applying any processing techniques to 921.


ExampleSet (30 examples, 4 special attributes, 921 regular attributes)

Using Tokenizer


ExampleSet (30 examples, 4 special attributes, 4194 regular attributes)

Using N-grams (character)


ExampleSet (30 examples, 4 special attributes, 2760 regular attributes)

Using N-grams(Terms)

2.2 Filters

As with all text mining projects there are words that are deemed useless to the text mining objective. These are referred to as stop words and can be removed using the operator filter stop words (English) **After** the tokenise operator. This is a build in stop word list for the English language. After applying this we reduced our bag of words down instantly to 772 attributes.

2.2.1 Filter token by length

Another filter that was utilized was the filter token (by length) which simply filters tokens by length. This is one of the most useless filtering operators as you don't know how short or long a character has to be to predict the class label. As a result, you are endangered of filtering out potential predictive attributes. In my case I scanned over my bag and words and came to the conclusion of configuring the minimum and maximum characters to 2 and 25 respectively in order to avoid this problem. As expected this minimally reduced our bag of words down to only 765 attributes.

2.2.2 Custom stop words (Dictionary)

A custom stop word list was defined based on domain knowledge as the genetic stop word list was not good enough to identify all of our possible stop words. This operator contains only one parameter to read in the file. The following screen shot contains configured stop words.

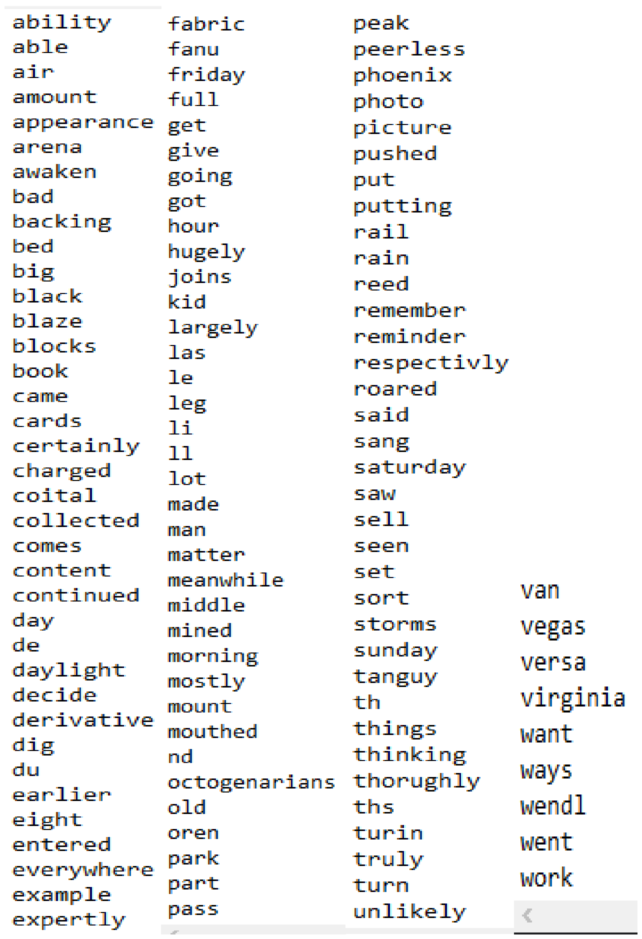
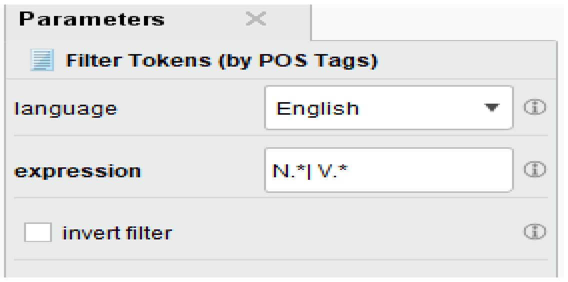

```
ability       fabric          peak
able          fanu            peerless
air           friday          phoenix
amount        full            photo
appearance    get             picture
arena         give            pushed
awaken        going           put
bad           got             putting
backing       hour            rail
bed           hugely          rain
big           joins           reed
black         kid             remember
blaze         largely         reminder
blocks        las             respectivly
book          le              roared
came          leg             said
cards         li              sang
certainly     ll              saturday
charged       lot             saw
coital        made            sell
collected     man             seen
comes         matter          set
content       meanwhile       sort          van
continued     middle          storms        vegas
day           mined           sunday        versa
de            morning         tanguy        virginia
daylight      mostly          th            want
decide        mount           things        ways
derivative    mouthed         thinking      wendl
dig           nd              thorughly     went
du            octogenarians   ths           work
earlier       old             turin
eight         oren            truly
entered       park            turn
everywhere    part            unlikely
example       pass
expertly
```

Figure 2.2.2: Defined stop words

After applying this collection of stop words we reduced of bag of words from 765 down to 663. These stop words where identified earlier from each of our 3 words clouds during the text visualisation phase.

2.2.3 filter token (By POS)

This method allows you to filter tokens based on their parts of speech tags. These tags are from the PENN system for English and are configured based on regular expression. The following screen shot shows the configured parameters for POS.


Parameters

Filter Tokens (by POS Tags)

language: English

expression: N.*| V.*

invert filter: ☐

Figure 2.2.3: Regular expressions parameter for POS

This was another operator I worried would filter out potential words that could be predicative of the documents. Therefore, this operator was not used.

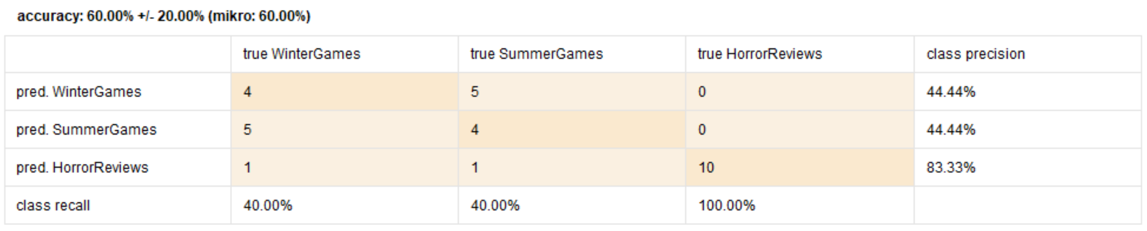
2.3 Stemmers

Stemmers are Algorithms that convert words in to other terms typically to the smallest unit or root of that word e.g. Buying to Buy. In this project I experimented with all 3 different forms of stemmers and compared both porter and lovins algorithms.

2.3.1 Porters

The idea of porter’s algorithms is that suffixes of all words are essentially just made up of more simpler and smaller suffixes. Porter removes these suffixes e.g. connection would be reduced to connect. This reduces the number of terms in our system, so it would be advantages in our process.

When we implement this operator in to our process our bag of words where reduced from 451 to 421 a decrease of 25 attributes. However, on further inspection of the bag of words it was discovered that some words where broken down too far losing some meaning of the word examples include favourite/favourit, bronze/bronz, comedy/comedi. What’s very interesting is that despite this there was an increase in model accuracy when using a decision tree classifier going from a base line of 56% to 60% an increase of 4%.


accuracy: 60.00% +/- 20.00% (mikro: 60.00%)

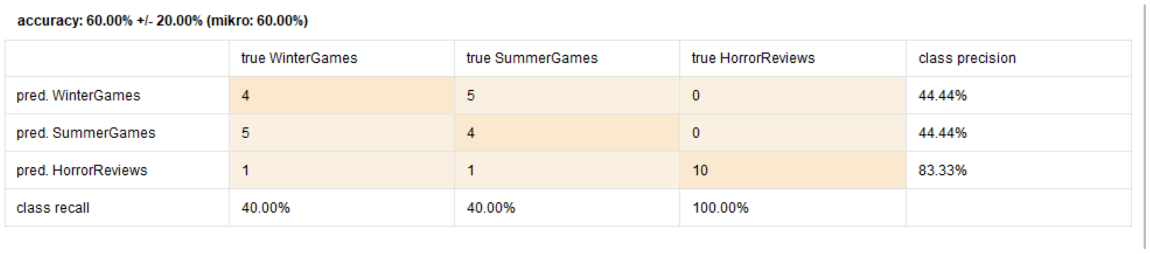
	true WinterGames	true SummerGames	true HorrorReviews	class precision
pred. WinterGames	4	5	0	44.44%
pred. SummerGames	5	4	0	44.44%
pred. HorrorReviews	1	1	10	83.33%
class recall	40.00%	40.00%	100.00%	

Figure 2.3.1: Accuracy when using porter’s stemmer

2.3.2 Lovins

One of the first popular stemmer that was effetely used is lovins. Its purpose is similar to porters but is implemented differently. Lovins is based on the longest match principle by performing variously lookups of tables with different endings (294) conditions (29) and transformation rules (35). The longest suffix of a given word is removed at least once as it an example of a single pass algorithm.

Similar to porter to does the same with the bag of words this time breaking down the word zombie to just zomb so it appears that lovins breaks down these terms even further then porters. However, the number of attributes reduced was even better then porters being reduced from 451 to 415. 11 attributes better then porters. In terms of accuracy there was no change when using a decision tree.


accuracy: 60.00% +/- 20.00% (mikro: 60.00%)

	true WinterGames	true SummerGames	true HorrorReviews	class precision
pred. WinterGames	4	5	0	44.44%
pred. SummerGames	5	4	0	44.44%
pred. HorrorReviews	1	1	10	83.33%
class recall	40.00%	40.00%	100.00%	

Figure 2.3.2: Accuracy is the same as porters but produces less attributes

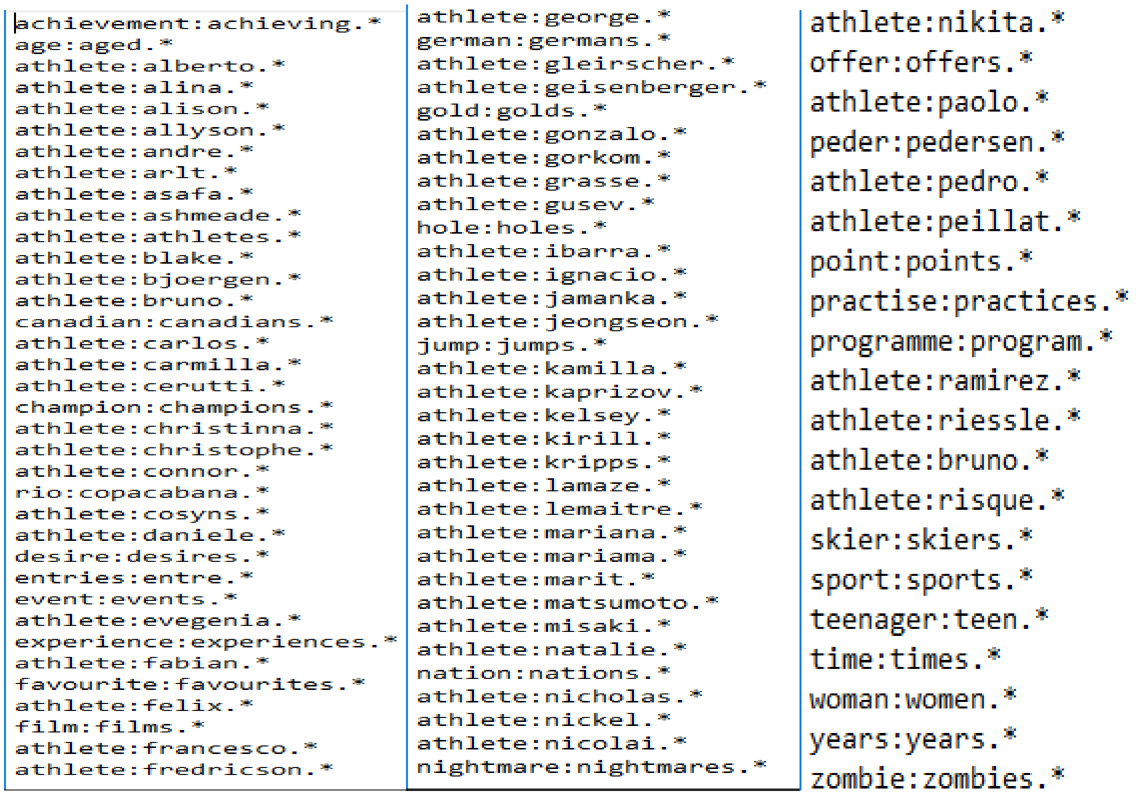
2.3.3 Comparison between porters and lovins

Lovins	**Porters**
Faster then lovins due to the nature of it being a single pass algorithm	Less error rate compared to lovins
Reduce attributes 451 – 415	Reduce attributes 451-421
Accuracy 60%	Accuracy 60%
Handles irregular plurals better	Lightweight stemmer

Figure 2.3.3: In conclusion I believe lovin to be the better algorithm only slightly.

2.3.4 Stemmer (dictionary)

A way we can improve these steamer is my defining our own dictionary to reduce terms to their base by using replacement rules. By using the world cloud earlier in the report, it was easy to identify possible synonyms, the following contains screenshots of our stemmer list. Seeing how this domain is in relation to the Olympics a number of athlete names where needed to be added to the synonym list.



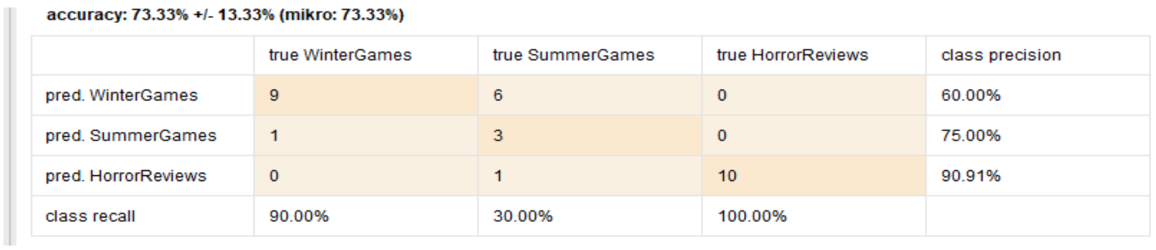
```
achievement:achieving.*
age:aged.*
athlete:alberto.*
athlete:alina.*
athlete:alison.*
athlete:allyson.*
athlete:andre.*
athlete:arlt.*
athlete:asafa.*
athlete:ashmeade.*
athlete:athletes.*
athlete:blake.*
athlete:bjoergen.*
athlete:bruno.*
canadian:canadians.*
athlete:carlos.*
athlete:carmilla.*
athlete:cerutti.*
champion:champions.*
athlete:christinna.*
athlete:christophe.*
athlete:connor.*
rio:copacabana.*
athlete:cosyns.*
athlete:daniele.*
desire:desires.*
entries:entre.*
event:events.*
athlete:evegenia.*
experience:experiences.*
athlete:fabian.*
favourite:favourites.*
athlete:felix.*
film:films.*
athlete:francesco.*
athlete:fredricson.*
athlete:george.*
german:germans.*
athlete:gleirscher.*
athlete:geisenberger.*
gold:golds.*
athlete:gonzalo.*
athlete:gorkom.*
athlete:grasse.*
athlete:gusev.*
hole:holes.*
athlete:ibarra.*
athlete:ignacio.*
athlete:jamanka.*
athlete:jeongseon.*
jump:jumps.*
athlete:kamilla.*
athlete:kaprizov.*
athlete:kelsey.*
athlete:kirill.*
athlete:kripps.*
athlete:lamaze.*
athlete:lemaitre.*
athlete:mariana.*
athlete:mariama.*
athlete:marit.*
athlete:matsumoto.*
athlete:misaki.*
athlete:natalie.*
nation:nations.*
athlete:nicholas.*
athlete:nickel.*
athlete:nicolai.*
nightmare:nightmares.*
athlete:nikita.*
offer:offers.*
athlete:paolo.*
peder:pedersen.*
athlete:pedro.*
athlete:peillat.*
point:points.*
practise:practices.*
programme:program.*
athlete:ramirez.*
athlete:riessle.*
athlete:bruno.*
athlete:risque.*
skier:skiers.*
sport:sports.*
teenager:teen.*
time:times.*
woman:women.*
years:years.*
zombie:zombies.*
```

Figure 2.3.4: Configured synonyms list

The number of attributes after configuring my own set of synonyms was reduced to 364 attributes combining this with both lovins and porter we get the number of attributes even further down to 350 and 360 respectively.

2.4 Pruning

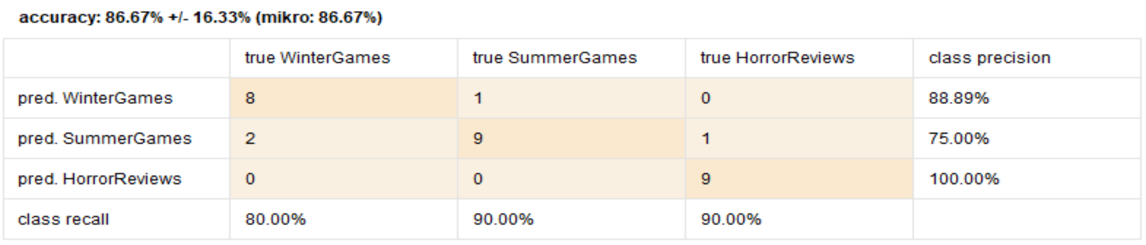
The very last step in producing our final bag of words is to apply pruning. This reduce our number of bags to a very fine number of attributes 58 in total. This was done by configuring the prune method to absolute and the below and above 3 and 19 (19 being the highest occurring attribute in our documents) respectively. This pruning method gives us an accuracy of 73.33 with K-NN, 86.67 with Naïve Bayes and 93.33 %. with decision tree using an occurrence document vector.



accuracy: 73.33% +/- 13.33% (mikro: 73.33%)

	true WinterGames	true SummerGames	true HorrorReviews	class precision
pred. WinterGames	9	6	0	60.00%
pred. SummerGames	1	3	0	75.00%
pred. HorrorReviews	0	1	10	90.91%
class recall	90.00%	30.00%	100.00%	

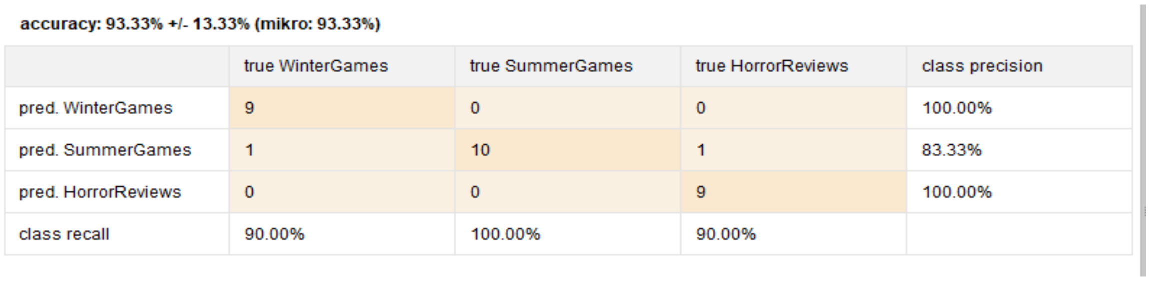
Figure 2.4: Accuracy using K-NN after pruning



accuracy: 86.67% +/- 16.33% (mikro: 86.67%)

	true WinterGames	true SummerGames	true HorrorReviews	class precision
pred. WinterGames	8	1	0	88.89%
pred. SummerGames	2	9	1	75.00%
pred. HorrorReviews	0	0	9	100.00%
class recall	80.00%	90.00%	90.00%	

Figure 2.4.1: Accuracy using Naïve Bayes after pruning



accuracy: 93.33% +/- 13.33% (mikro: 93.33%)

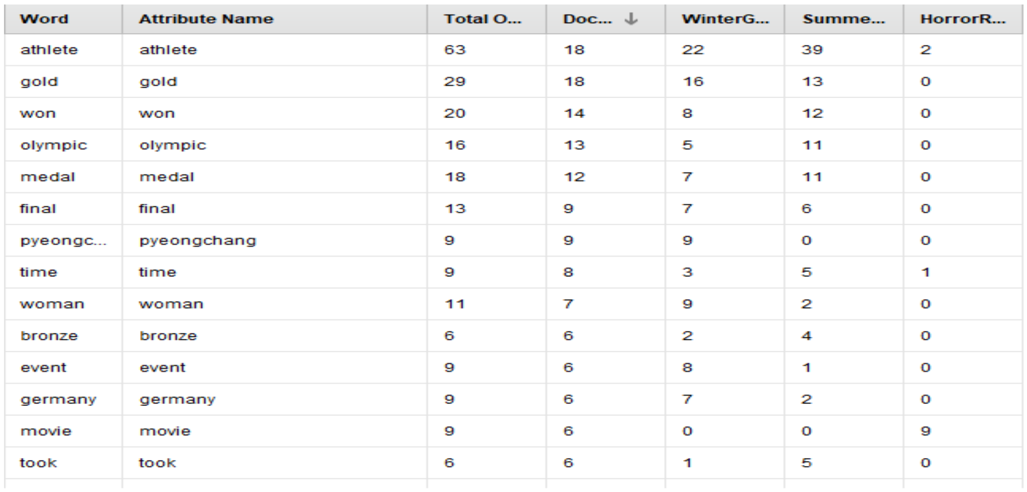
	true WinterGames	true SummerGames	true HorrorReviews	class precision
pred. WinterGames	9	0	0	100.00%
pred. SummerGames	1	10	1	83.33%
pred. HorrorReviews	0	0	9	100.00%
class recall	90.00%	100.00%	90.00%	

Figure 2.4.2: Accuracy using Naïve Bayes after pruning

The pruning stage of the project was very much an iterative process. Pruning above between 10 and 18 all gave the same level in accuracy (63.33 using decision tree). The same goes for pruning between 19 and 30 (93.33). This is a result of it picking up more common terms the document, so the best pruning value was 19 as a result

2.5 Final Bag of words and processes

After all these processing techniques are done and implemented a final bag of words was produced the following screenshot explains the range of occurrences in which what words where the most predictive.



Word	Attribute Name	Total O...	Doc...	WinterG...	Summe...	HorrorR...
athlete	athlete	63	18	22	39	2
gold	gold	29	18	16	13	0
won	won	20	14	8	12	0
olympic	olympic	16	13	5	11	0
medal	medal	18	12	7	11	0
final	final	13	9	7	6	0
pyeongc...	pyeongchang	9	9	9	0	0
time	time	9	8	3	5	1
woman	woman	11	7	9	2	0
bronze	bronze	6	6	2	4	0
event	event	9	6	8	1	0
germany	germany	9	6	7	2	0
movie	movie	9	6	0	0	9
took	took	6	6	1	5	0

Figure 2.5: Sample screenshot of our final bag of words

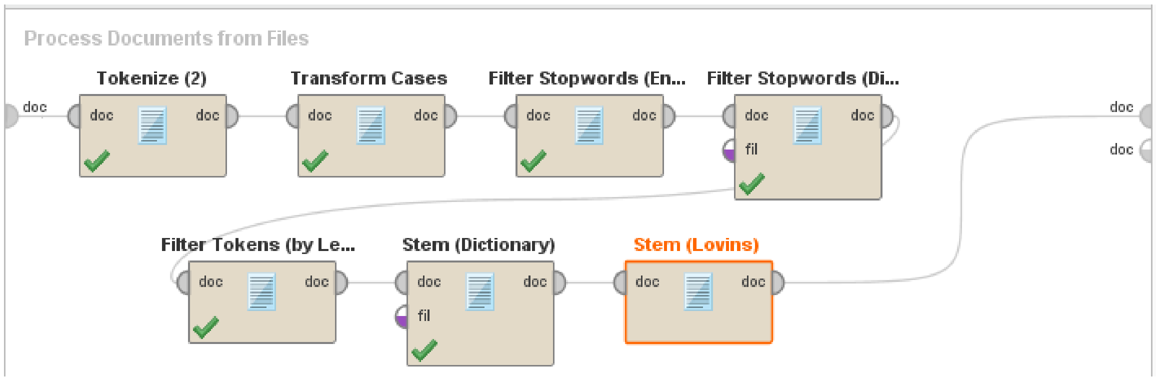


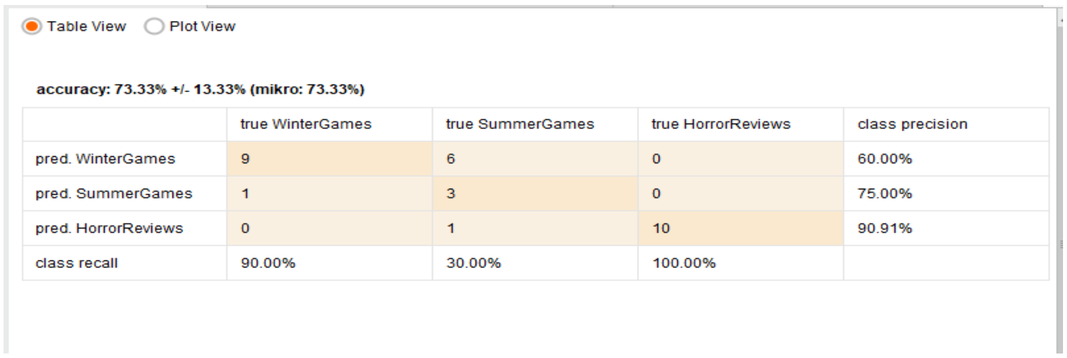
Figure 2.5.1: Final Pre-processing process

3. Feature selection statistics /counting

In this section several different modules will be experimented with based on binary document vectors, occurrence vectors and TF-IDF. All these processes are responsible for document vector generation. These can be configured by selecting the dropdown menu and selecting the following…

3.1 Binary document vectors

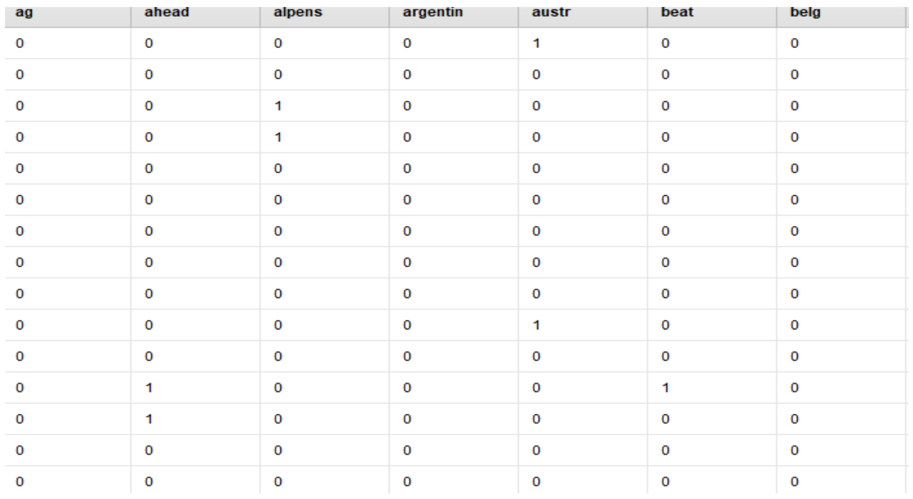
These are the cell entries of 0 and 1. 0 meaning the word is absent in the document and 1 means their word is present in the document (see figure). When this was selected the accuracy was 73.33% when using a decision tree.



accuracy: 73.33% +/- 13.33% (mikro: 73.33%)

	true WinterGames	true SummerGames	true HorrorReviews	class precision
pred. WinterGames	9	6	0	60.00%
pred. SummerGames	1	3	0	75.00%
pred. HorrorReviews	0	1	10	90.91%
class recall	90.00%	30.00%	100.00%	

Figure 3.1: Accuracy using a Binary document vector

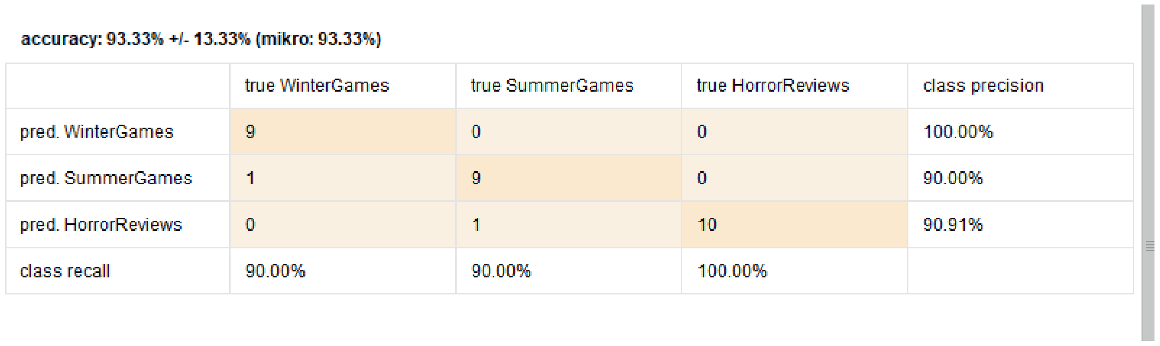


ag	ahead	alpens	argentin	austr	beat	belg
0	0	0	0	1	0	0
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	0	1	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	1	0	0
0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Figure 3.1.1 Binary document vector

3.2 Term Occurrence

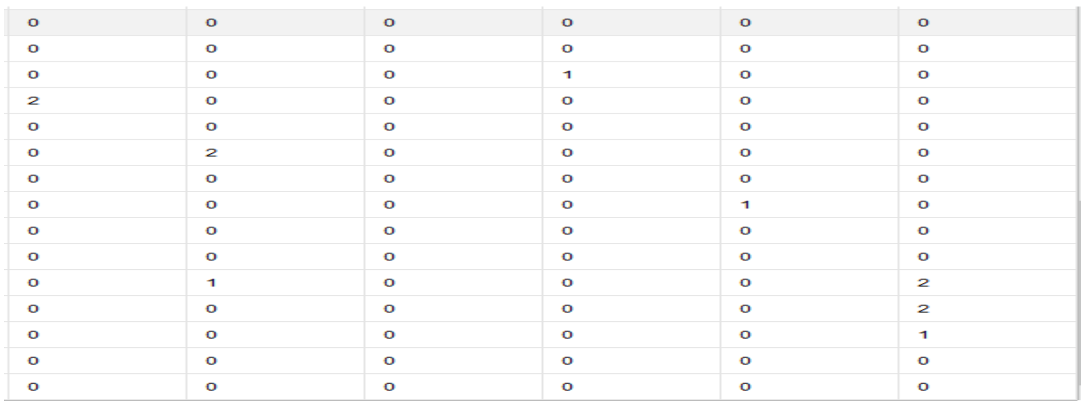
This indicates how often the word appears in the document through the conventional way. When we select this option the level in accuracy is 93.33% when using a decision tree.



accuracy: 93.33% +/- 13.33% (mikro: 93.33%)

	true WinterGames	true SummerGames	true HorrorReviews	class precision
pred. WinterGames	9	0	0	100.00%
pred. SummerGames	1	9	0	90.00%
pred. HorrorReviews	0	1	10	90.91%
class recall	90.00%	90.00%	100.00%	

Figure 3.2: Accuracy using term occurrence vector

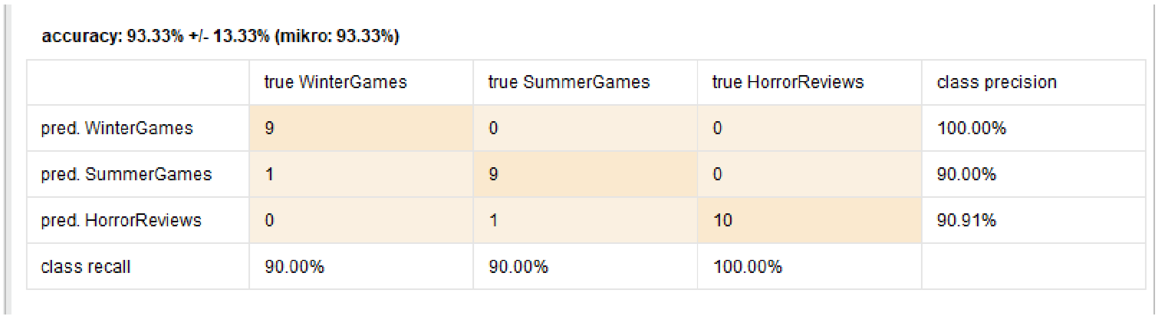


0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	1	0	0
2	0	0	0	0	0
0	0	0	0	0	0
0	2	0	0	0	0
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	0	0	0	0	0
0	1	0	0	0	2
0	0	0	0	0	2
0	0	0	0	0	1
0	0	0	0	0	0
0	0	0	0	0	0

Figure 3.2.1: Term occurrence vector

3.3 TF-IDF

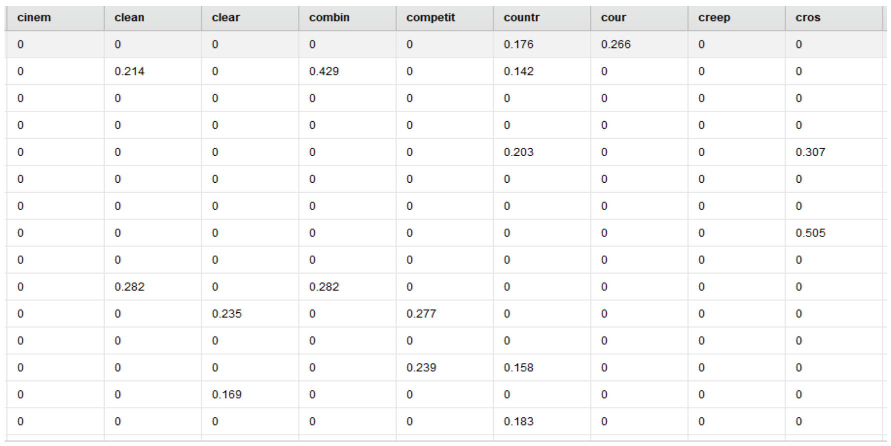
Treats infrequent terms with larger weights. In our case it increased our accuracy with a decision tree to 93.33%.



accuracy: 93.33% +/- 13.33% (mikro: 93.33%)

	true WinterGames	true SummerGames	true HorrorReviews	class precision
pred. WinterGames	9	0	0	100.00%
pred. SummerGames	1	9	0	90.00%
pred. HorrorReviews	0	1	10	90.91%
class recall	90.00%	90.00%	100.00%	

Figure 3.3 Accuracy using TF-IDF vector



cinem	clean	clear	combin	competit	countr	cour	creep	cros
0	0	0	0	0	0.176	0.266	0	0
0	0.214	0	0.429	0	0.142	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0.203	0	0	0.307
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0.505
0	0	0	0	0	0	0	0	0
0	0.282	0	0.282	0	0	0	0	0
0	0	0.235	0	0.277	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0.239	0.158	0	0	0
0	0	0.169	0	0	0	0	0	0
0	0	0	0	0	0.183	0	0	0

Figure 3.3.1: TF-IDF Vector

4.Data Mining

4.1 Clustering

Clustering is simply understanding patterns within a particle dataset by understanding the naturally occurring groups and clusters within that dataset. A class label isn’t defined in advance. Objects in a cluster have high similarity to one another but dissimilar to other clusters

4.1.1 K-means

The following screenshot illustrates the k-means clustering method. Cluster 1 had both winter and summer games within it. Probably as winter and summer texts are quite like each other. Cluster 2 has all 3 texts in the same cluster and the third cluster only had the horror reviews documents. K-means was used using a cosine measure this uses word frequencies rather than binary entries for them. other measures where used during the evaluation stage of the project.

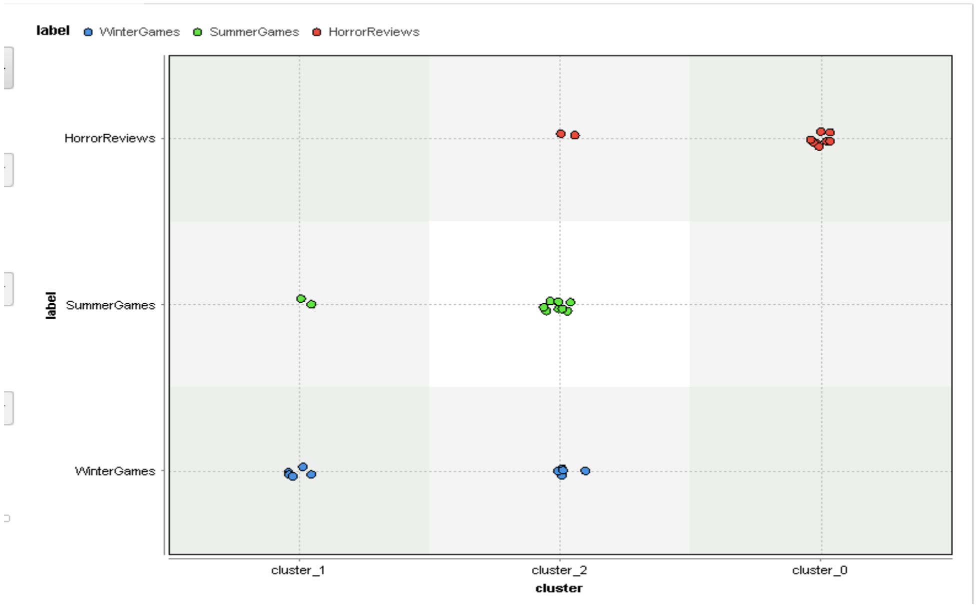
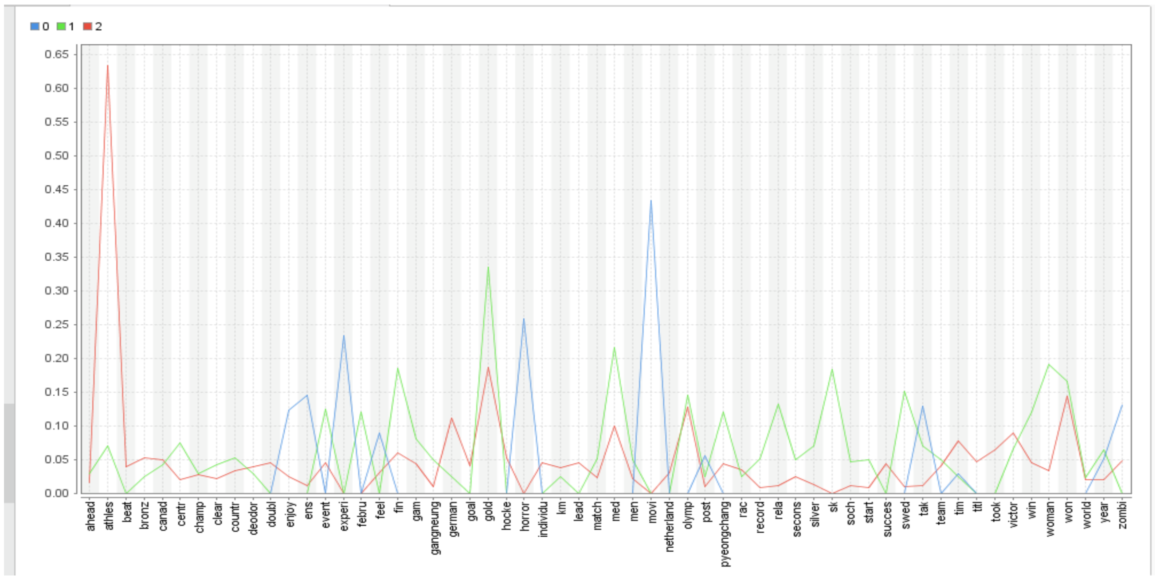


Figure 4.1.1: K-means cosine measure results on a scatter plot

The following parallel graph shows us which attributes are the most common you can tell by simply looking at the plot you can tell athletes is the most common term followed by movie and horror. The graph can also tell which document topics overlap.



4.1.2 Agglomerative clustering

This method of clustering works by grouping the objects into a tree of clusters where the child represents the optimal split for the group for the assignment it was experimented with both single and complete linkage.

4.1.2.1 Single linkage

This is the distance between two clusters based on the closest 2 points in those clusters. Good for non-globular shapes but the present of outliers can have an effect on how the clusters are merged. The diagram tells us that summer and winter texts are all put in to the one cluster in contrast to the k-means method where the two texts where separate in to two distinct clusters. Horror reviews has been allocated to its own cluster as expected.

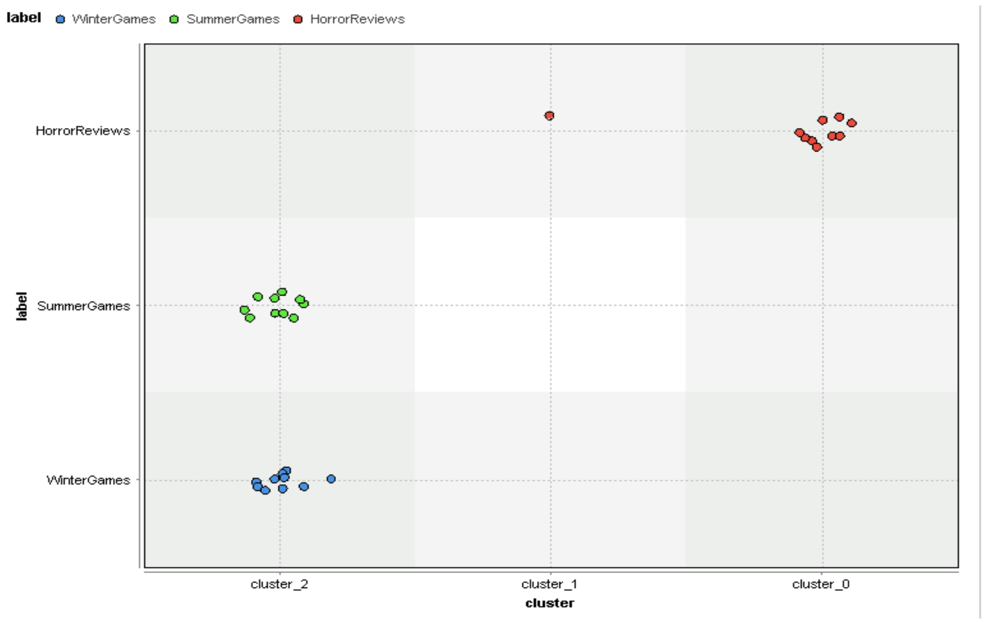


Figure 4.1.2.1: Agglomerative with single linkage results on a scatter plot

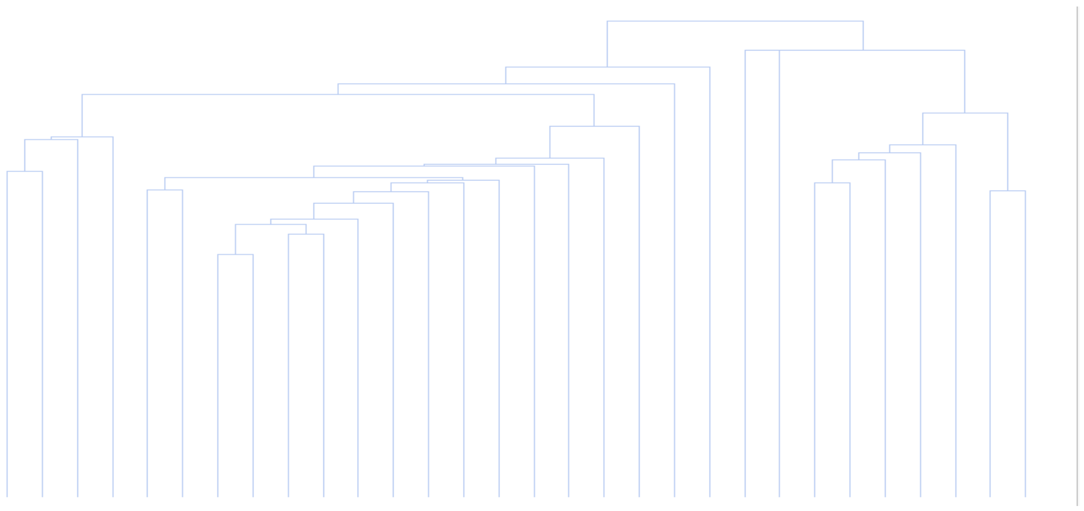


Figure 4.1.2.1: Agglomerative dendrogram with single linkage

4.1.2.2 Complete linkage

This is similar as single linkage but instead it's the distance between the further points that are in different clusters. Terms are measured on least similar points. The difference we can see is that cluster_1 contains all documents horror summer and winter the 2 and third cluster contain all horror texts it's seems out of the two-single linkage was the best choice out of the agglomerative clustering.

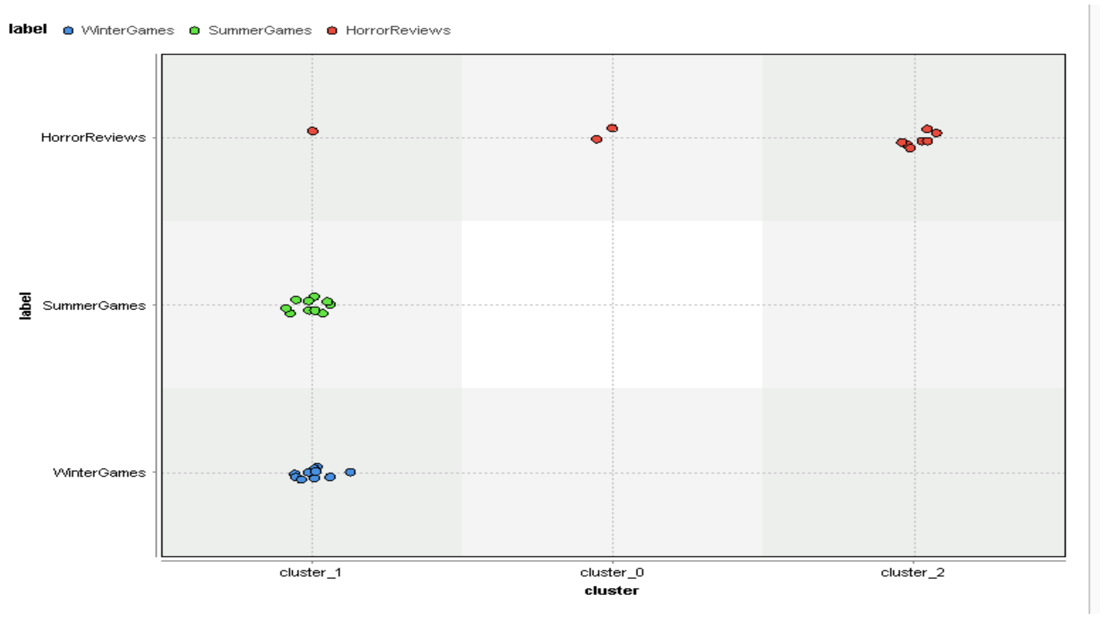


Figure 4.1.2.2: Agglomerative with complete linkage results on a scatter plot

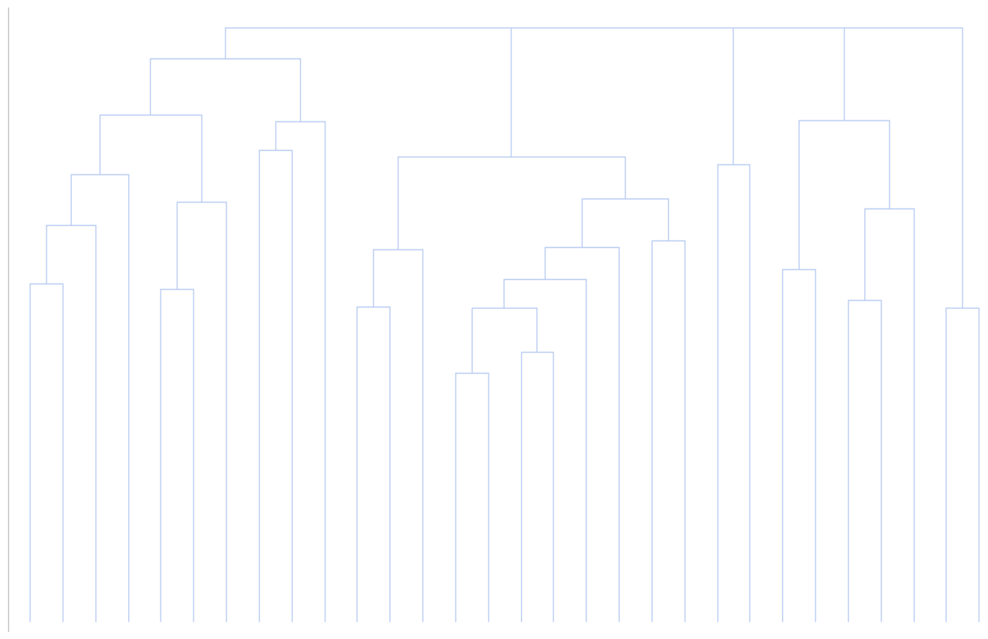


Figure 4.1.2.3: Agglomerative dendrogram with complete linkage

4.1.3 Self-organising map (SOM)

Using SOM, the clustering did not perform well at all maybe due to the fact that it’s a neural network algorithm which typically favours numeric attributes. Using Som it’s seems that winter and summer games get fixed up. This proved not to be very useful

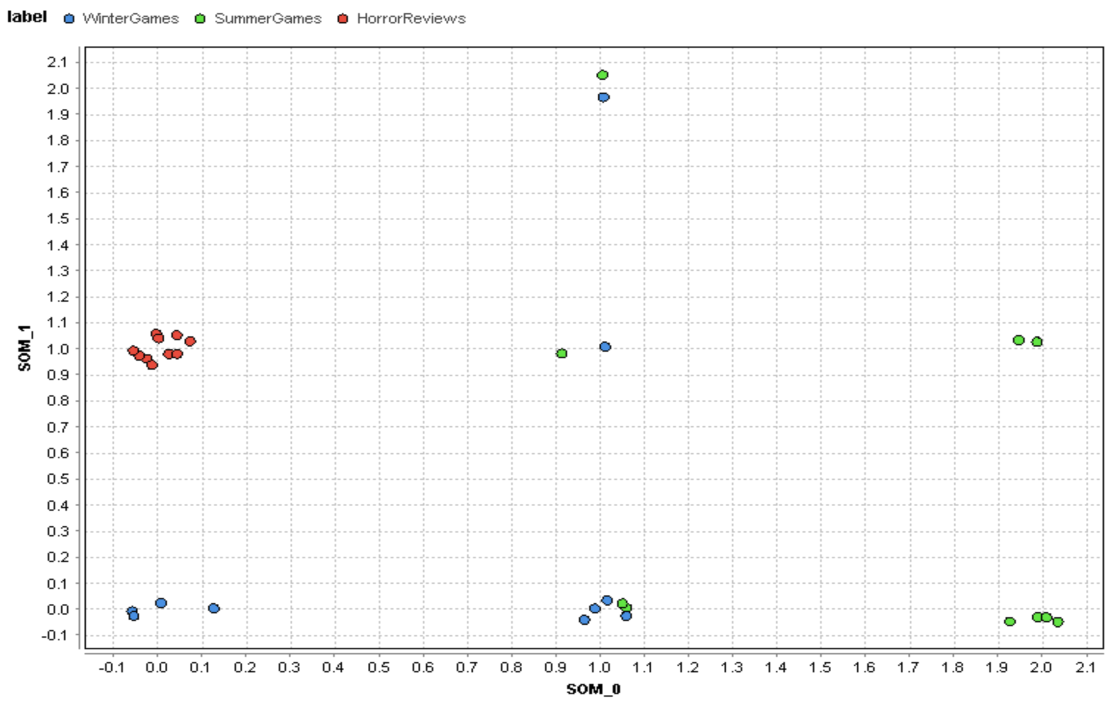


Figure 4.1.3: SOM results on a scatter plot 2x3

4.1.4. Clustering Evaluation
The next step in the project is to run symmetric distance measure (Euclidean and Manhattan) on each of these algorithms to see which one yields the best results.

4.1.4.1 K-means Euclidean
The Euclidean measure is like the cosine, but it seems to have less horror documents in cluster two instead putting it in cluster three which is good as it distinguishes them this is when we compare it to cosine which was the default that was set earlier in the report.

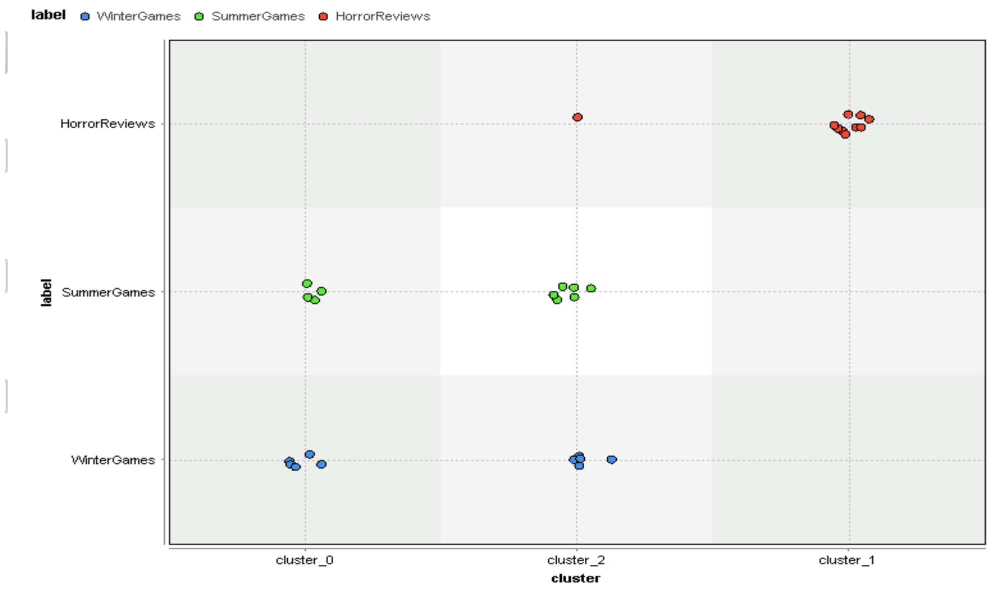


Figure 4.1.4.1: K-means Euclidean scatter plot results

4.1.4.2 K-means Manhattan
The Manhattan measure removes some attributes from cluster_0 and puts them into cluster_1 this is not what we want. Not that useful in our case but can be good distance measure for numeric attributes.

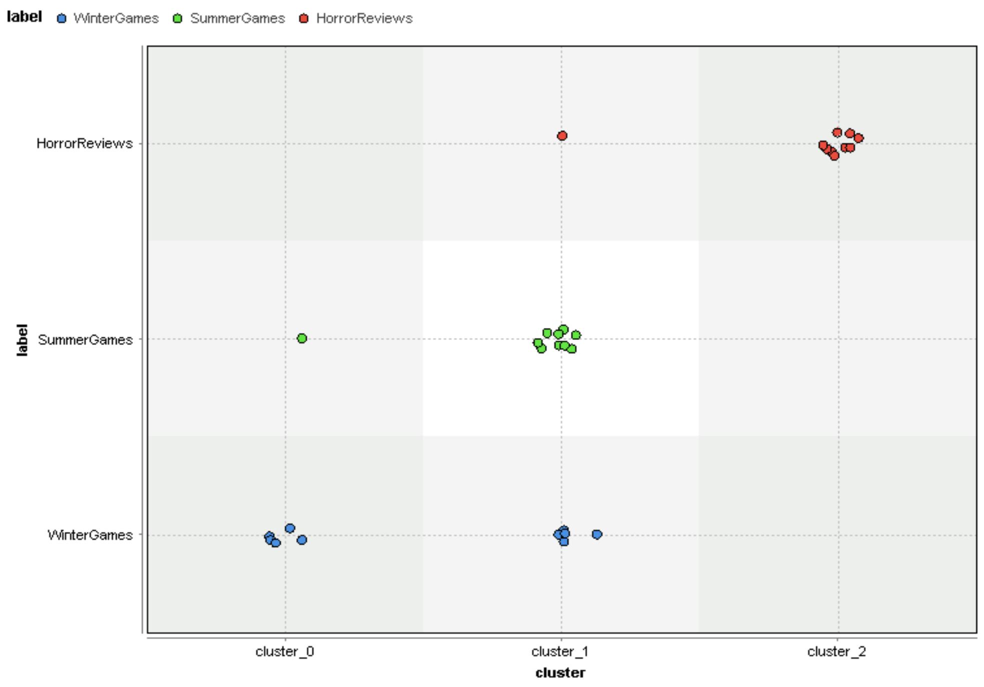
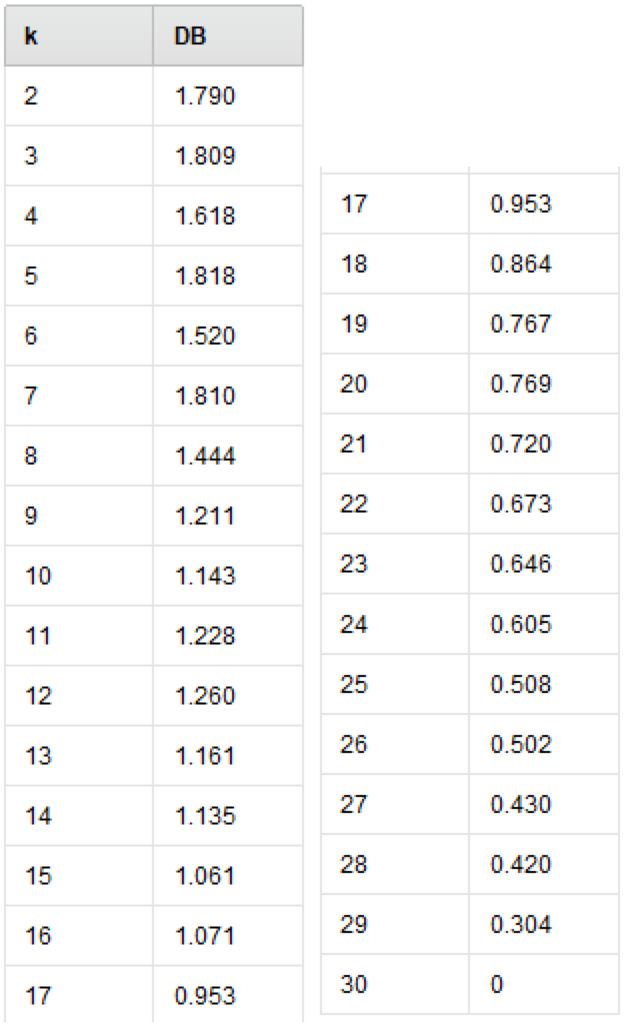


Figure 4.1.4.2: K-means Manhattan scatter plot results

4.1.5 Davies Bouldin Objective measure

Davies Bouldin is an objective measure evaluation of a cluster with no other external information such as labels required. The DB measurement is how far away a cluster is from each other we want this DB score to be as close to 0 as possible. As you can see from the screen shot Davies Bouldin didn't work out so well as the optimum DB was when K = 29 in comparison to when it was running on the iris dataset where K = 2 was optimal.



k	DB
2	1.790
3	1.809
4	1.618
5	1.818
6	1.520
7	1.810
8	1.444
9	1.211
10	1.143
11	1.228
12	1.260
13	1.161
14	1.135
15	1.061
16	1.071
17	0.953
18	0.864
19	0.767
20	0.769
21	0.720
22	0.673
23	0.646
24	0.605
25	0.508
26	0.502
27	0.430
28	0.420
29	0.304
30	0

Figure 4.1.5: Davies Bouldin calculations

4.1.6 Conclusion

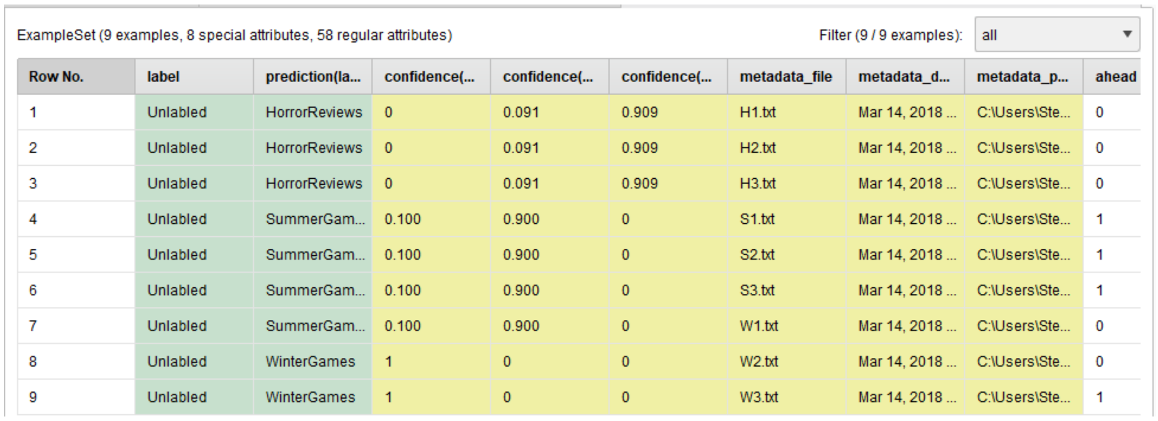
In conclusion the best clustering algorithm to use in my experience is agglomerative with single linkage as it gives the best separation among clusters. This was followed by K-means with Euclidean measure which was unusual as distance measure typically don't perform well with document vectors. The type of document vector makes no difference in the effectiveness of the algorithms with all 3 vectors containing the same accuracy.

4.2 Classification- Supervised Learning

During this stage we spilt our dataset up into both training and tests datasets. We train a model based on the 30 documents that was collected and see if it can calculate how many rows where predicted correctly. The three classification algorithms that where used was Decision tree, K-NN and Naïve-Bayes.

4.2.1 Decision tree

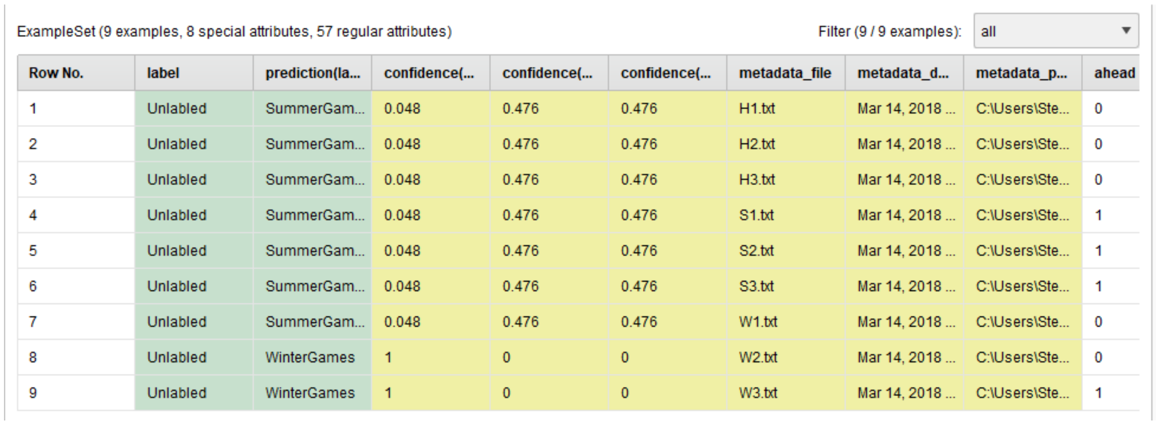
A decision tree model managed to do surprisingly well when it was ran only getting one document wrong by mixing up a summer text with a winter. The reason why this worked well was we allowed the attribute Gold to appear in the bag of words (this occurred in 19/20 documents) when we pruned this below 19 the accuracy dropped as it got 4 documents wrong. So obviously the word Attribute was a highly predictive term for a Decision tree.



ExampleSet (9 examples, 8 special attributes, 58 regular attributes) Filter (9 / 9 examples): all

Row No.	label	prediction(la...	confidence(...	confidence(...	confidence(...	metadata_file	metadata_d...	metadata_p...	ahead
1	Unlabled	HorrorReviews	0	0.091	0.909	H1.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
2	Unlabled	HorrorReviews	0	0.091	0.909	H2.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
3	Unlabled	HorrorReviews	0	0.091	0.909	H3.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
4	Unlabled	SummerGam...	0.100	0.900	0	S1.txt	Mar 14, 2018 ...	C:\Users\Ste...	1
5	Unlabled	SummerGam...	0.100	0.900	0	S2.txt	Mar 14, 2018 ...	C:\Users\Ste...	1
6	Unlabled	SummerGam...	0.100	0.900	0	S3.txt	Mar 14, 2018 ...	C:\Users\Ste...	1
7	Unlabled	SummerGam...	0.100	0.900	0	W1.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
8	Unlabled	WinterGames	1	0	0	W2.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
9	Unlabled	WinterGames	1	0	0	W3.txt	Mar 14, 2018 ...	C:\Users\Ste...	1

Figure 4.2.1: Prediction with the gold attribute 8/9



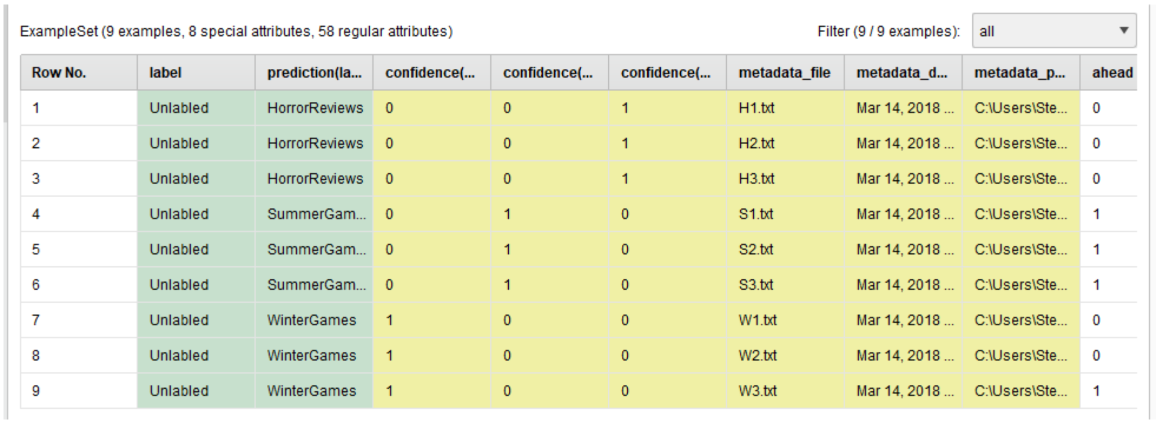
ExampleSet (9 examples, 8 special attributes, 57 regular attributes) Filter (9 / 9 examples): all

Row No.	label	prediction(la...	confidence(...	confidence(...	confidence(...	metadata_file	metadata_d...	metadata_p...	ahead
1	Unlabled	SummerGam...	0.048	0.476	0.476	H1.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
2	Unlabled	SummerGam...	0.048	0.476	0.476	H2.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
3	Unlabled	SummerGam...	0.048	0.476	0.476	H3.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
4	Unlabled	SummerGam...	0.048	0.476	0.476	S1.txt	Mar 14, 2018 ...	C:\Users\Ste...	1
5	Unlabled	SummerGam...	0.048	0.476	0.476	S2.txt	Mar 14, 2018 ...	C:\Users\Ste...	1
6	Unlabled	SummerGam...	0.048	0.476	0.476	S3.txt	Mar 14, 2018 ...	C:\Users\Ste...	1
7	Unlabled	SummerGam...	0.048	0.476	0.476	W1.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
8	Unlabled	WinterGames	1	0	0	W2.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
9	Unlabled	WinterGames	1	0	0	W3.txt	Mar 14, 2018 ...	C:\Users\Ste...	1

Figure 4.2.1: Prediction without the gold attribute 5/9

4.2.2 K-NN

K-NN proved to be the best algorithm as it managed to predict every single document correct. It's also didn't require the most common attribute in all the three documents to make this prediction. This is when K= 3. It was important that when choosing the value of K that we didn't select a value for K that was to small as it may be associated with an outlier. Likewise, for when K is to big it may include points from neighbours and other clusters.

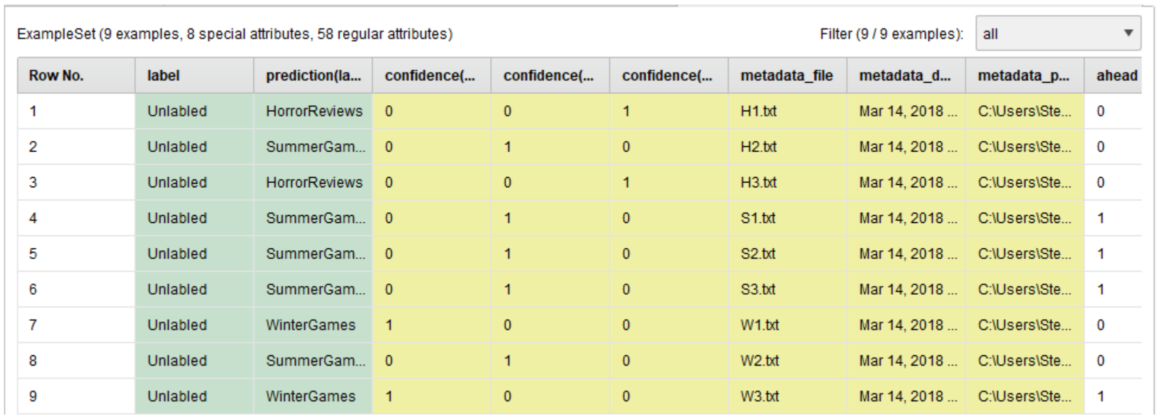

ExampleSet (9 examples, 8 special attributes, 58 regular attributes)　Filter (9 / 9 examples): all

Row No.	label	prediction(la...	confidence(...	confidence(...	confidence(...	metadata_file	metadata_d...	metadata_p...	ahead
1	Unlabled	HorrorReviews	0	0	1	H1.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
2	Unlabled	HorrorReviews	0	0	1	H2.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
3	Unlabled	HorrorReviews	0	0	1	H3.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
4	Unlabled	SummerGam...	0	1	0	S1.txt	Mar 14, 2018 ...	C:\Users\Ste...	1
5	Unlabled	SummerGam...	0	1	0	S2.txt	Mar 14, 2018 ...	C:\Users\Ste...	1
6	Unlabled	SummerGam...	0	1	0	S3.txt	Mar 14, 2018 ...	C:\Users\Ste...	1
7	Unlabled	WinterGames	1	0	0	W1.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
8	Unlabled	WinterGames	1	0	0	W2.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
9	Unlabled	WinterGames	1	0	0	W3.txt	Mar 14, 2018 ...	C:\Users\Ste...	1

Figure 4.2.2: Prediction using K-NN 9/9

4.2.3 Naïve Bayes

When Naïve Bayes was applied to the test data it predicated 2 rows wrong. This classification proved to be the worst among the other classifiers, but it could have been useful if our dataset had a very high dataset dimensionality. It makes sense that it would perform poorly as some of our attributes are closely related like gold and medal and our example dataset also includes synonyms of all the athlete names. Naïve Bayes treats each term independently and can perform poorly if dimensions contain these.


ExampleSet (9 examples, 8 special attributes, 58 regular attributes)　Filter (9 / 9 examples): all

Row No.	label	prediction(la...	confidence(...	confidence(...	confidence(...	metadata_file	metadata_d...	metadata_p...	ahead
1	Unlabled	HorrorReviews	0	0	1	H1.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
2	Unlabled	SummerGam...	0	1	0	H2.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
3	Unlabled	HorrorReviews	0	0	1	H3.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
4	Unlabled	SummerGam...	0	1	0	S1.txt	Mar 14, 2018 ...	C:\Users\Ste...	1
5	Unlabled	SummerGam...	0	1	0	S2.txt	Mar 14, 2018 ...	C:\Users\Ste...	1
6	Unlabled	SummerGam...	0	1	0	S3.txt	Mar 14, 2018 ...	C:\Users\Ste...	1
7	Unlabled	WinterGames	1	0	0	W1.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
8	Unlabled	SummerGam...	0	1	0	W2.txt	Mar 14, 2018 ...	C:\Users\Ste...	0
9	Unlabled	WinterGames	1	0	0	W3.txt	Mar 14, 2018 ...	C:\Users\Ste...	1

Figure 4.2.3: Prediction with Naïve Bayes 7/9

4.3 Dimensionality reduction

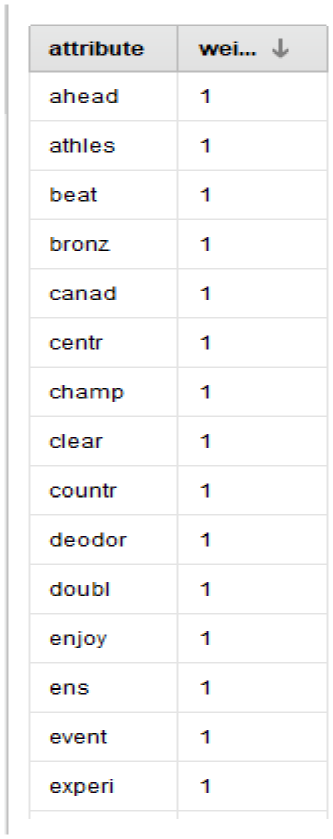
Typically, both classification and clustering algorithms tend to perform better if the dimensionality of the dataset is quite low it also shows the optimal number of terms that is required to classify our three groups of text. The weighting will also determine if the most predicative terms concur with what we expected to be the most predictive attribute The two categories of attributes reduction techniques that where perform where as follows.

1. Attribute weighting: This determines the attributes that are most predictive of the class label.
2. Attribute reduction: by merging attributes by using compression techniques like principle component analysis to minimise loss of information content.

4.3.1 Attribute weighting

This ranks attributes based on how useful they will be to the classification algorithms these weights are typically within the range of [0,1], the higher weight being the most predictive attribute. Seeing how we have three different classifiers well need weight operators.

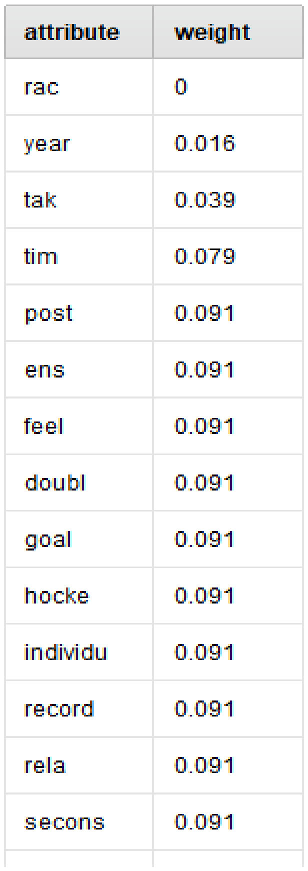
Weight by relief: This is used on K-NN and is based on two things. Does it have the values as it's neighbours in the same class or does it have a different value from neighbours in different classes. When we ran this in RapidMiner it didn't reduce our number of attributes (58). And it allocated the same weight to each of the attributes



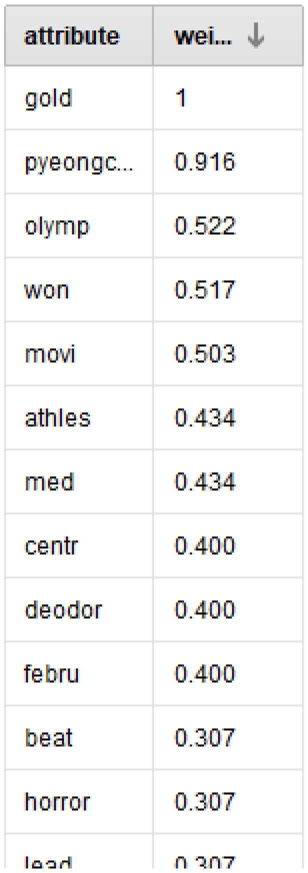
attribute	wei...
ahead	1
athles	1
beat	1
bronz	1
canad	1
centr	1
champ	1
clear	1
countr	1
deodor	1
doubl	1
enjoy	1
ens	1
event	1
experi	1

Figure 4.3.1: Weight by relief results

Wight by information gain: This is what a decision tree use when weighing attributes this ranks attributes by using information gain as a measure of chaos If an attribute is useful it should gives a clean branch that can spit classes. The results showed more promise then weight by relief as we managed to reduce our attributes down to 57 (The term race was removed) the most predictive attribute in our case by using information gain was gold.



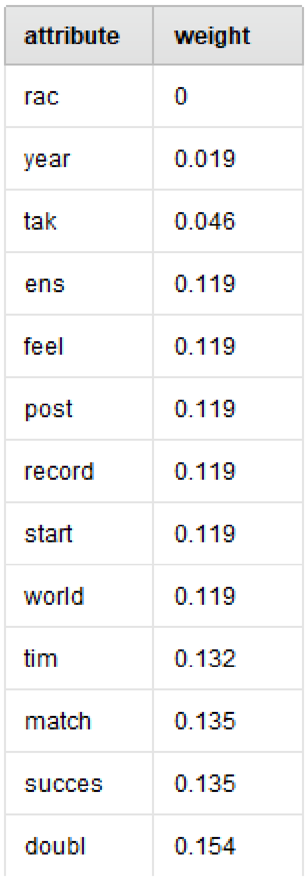
attribute	weight
rac	0
year	0.016
tak	0.039
tim	0.079
post	0.091
ens	0.091
feel	0.091
doubl	0.091
goal	0.091
hocke	0.091
individu	0.091
record	0.091
rela	0.091
secons	0.091



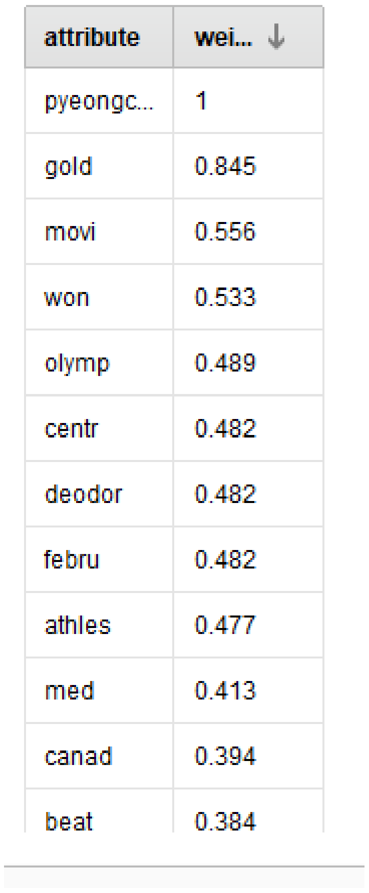
attribute	wei... ↓
gold	1
pyeongc...	0.916
olymp	0.522
won	0.517
movi	0.503
athles	0.434
med	0.434
centr	0.400
deodor	0.400
febru	0.400
beat	0.307
horror	0.307
lead	0.307

Figure 4.3.2: Weight by information gain results

Weight by uncertainty: Used by Naïve Bayes to weight attributes on the probability of a word belonging to a certain class like winter or summer games rather then equally being in all classes. The results here are similar to information gain as we are reduced down to 57 attributes with the word race being removed. What was interesting here was that PyeongChang proved to be the most predictive attribute instead of gold as was predicated using information gain.



attribute	weight
rac	0
year	0.019
tak	0.046
ens	0.119
feel	0.119
post	0.119
record	0.119
start	0.119
world	0.119
tim	0.132
match	0.135
succes	0.135
doubl	0.154



attribute	wei... ↓
pyeongc...	1
gold	0.845
movi	0.556
won	0.533
olymp	0.489
centr	0.482
deodor	0.482
febru	0.482
athles	0.477
med	0.413
canad	0.394
beat	0.384

Figure 4.3.1: Weight by Uncertainly results

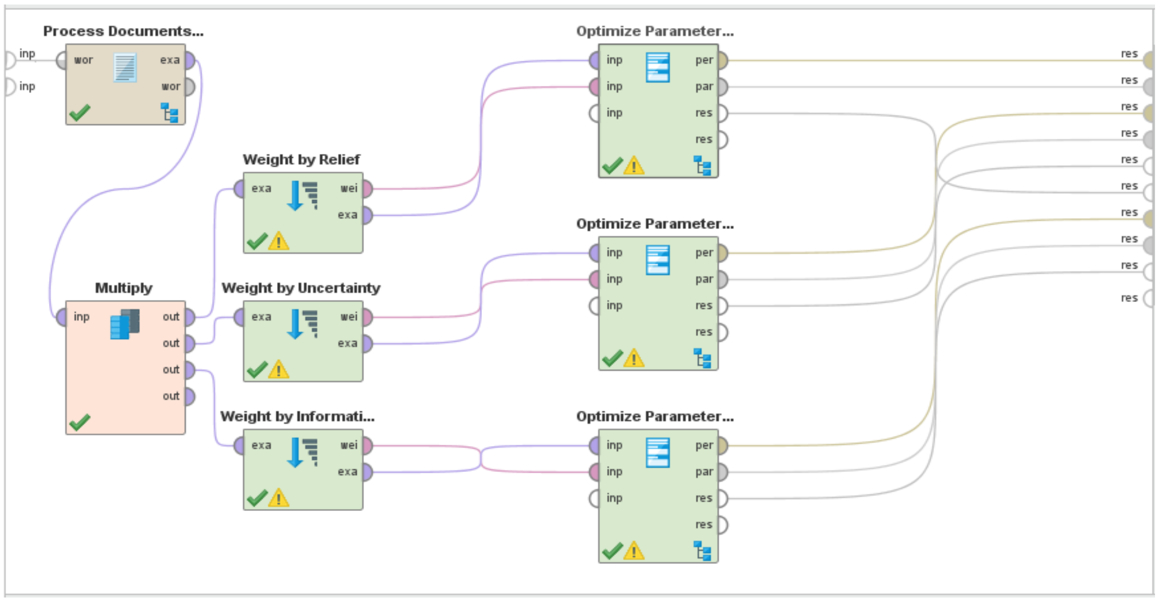
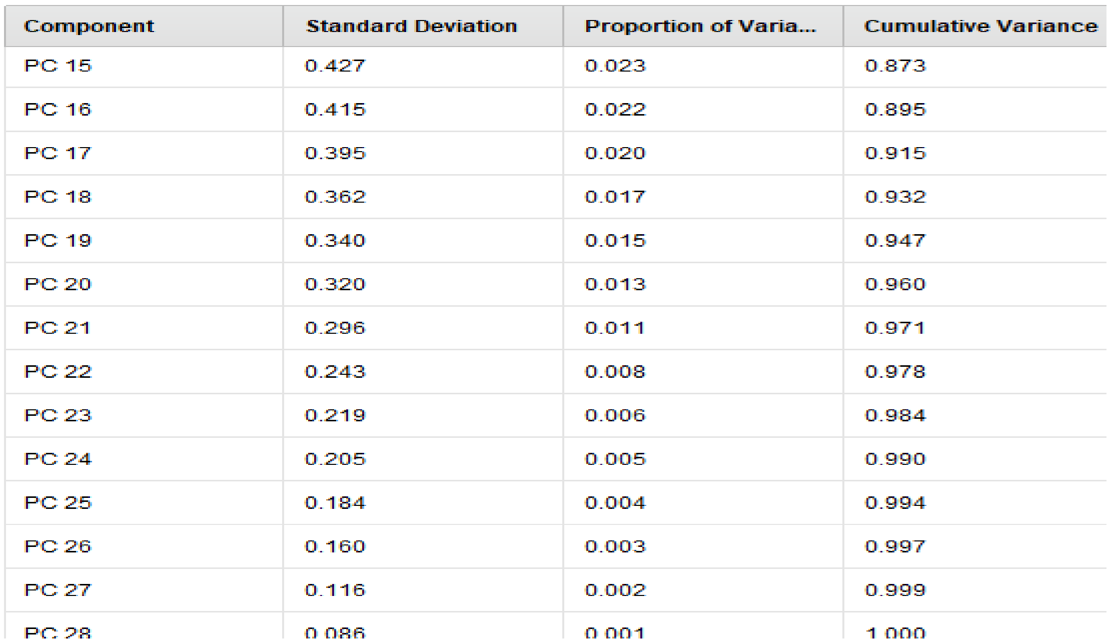


Figure 4.3.1: Weight process

4.3.2 Attribute reduction via Principal component analysis (PCA)

The reason we use PCA is to have the least number of attributes but still maintain all the relevant information from within the dataset. It does this by merging already existing attributes. Another reason we do this is also to get rid of any random noise or variation in the dataset. How it does this is by creating new attributes known as principal components that are essentially these combined attributes as previously mentioned. Typically, you would never have more then 3 PCA As we don't what too many as these PCA could capture the Noise or variance that we don't want. The first PCA captures the main pattern in the dataset while the second and third PCA merely capture the remaining patterns that the previously PCA failed to capture.

As we can see from the screen shot PC 28 captures all the variability in the dataset but to get rid of all the noise and variance it's best to use PCA19.



Component	Standard Deviation	Proportion of Varia...	Cumulative Variance
PC 15	0.427	0.023	0.873
PC 16	0.415	0.022	0.895
PC 17	0.395	0.020	0.915
PC 18	0.362	0.017	0.932
PC 19	0.340	0.015	0.947
PC 20	0.320	0.013	0.960
PC 21	0.296	0.011	0.971
PC 22	0.243	0.008	0.978
PC 23	0.219	0.006	0.984
PC 24	0.205	0.005	0.990
PC 25	0.184	0.004	0.994
PC 26	0.160	0.003	0.997
PC 27	0.116	0.002	0.999
PC 28	0.086	0.001	1.000

Figure 4.3.2: PCA results

4.4Conclusion

The terms chosen by each of the algorithms does concord in what I agree to be the most predictive for example the word zombie and movie was very good and detecting horror reviews. The same goes for terms in the summer and winter games thought it seems that it had a harder time in classifying them right which was to be expected as both text share common terms like medal and gold. After running the different weights, it showed no surprise as which attributes proved to be the most predictive words like gold, won medal, movie zombie and PyeongChang all proved to be very predictive of the class label By using PCA we can have an optimal number of PCA 19 to classify our three texts.

5. Analysing Results and Conclusion

5.1 Receiver Operator Characteristic

The following screenshot shows a Receiver Operator Characteristic (ROC) this describes the performance of algorithms. The area under the curve describes the Algorithm In our case it was SVM.

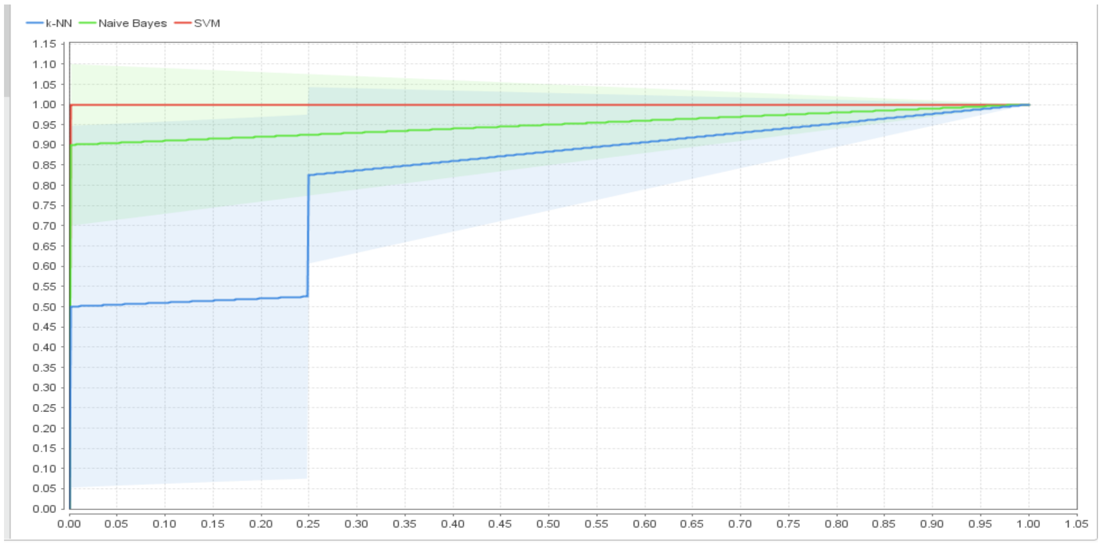


Figure 5.1 Receiver Operator Characteristic

5.2 Conclusion

In relation to the original business objective it's was essential to produce both a clustering model and a classification that could predict all three class labels from a test dataset with a reasonable level of accuracy and this is the case when using K-NN as it only makes one mistake in predicating our 9 texts. The clustering was best preformed under an agglomerative method with single linkage but still failed to separate each three categories of text in to their own cluster, possibly due to the similarities of the two of the categories. None the less I believe this assignment was a successful one and look forward to doing it again someday.

6. Appendix

Pre-processing

```xml
<?xml version="1.0" encoding="UTF-8"?><process version="7.6.001">
  <context>
    <input/>
    <output/>
    <macros/>
  </context>
  <operator activated="true" class="process" compatibility="7.6.001" expanded="true"
name="Process">
    <parameter key="logverbosity" value="init"/>
    <parameter key="random_seed" value="2001"/>
    <parameter key="send_mail" value="never"/>
    <parameter key="notification_email" value=""/>
    <parameter key="process_duration_for_mail" value="30"/>
    <parameter key="encoding" value="SYSTEM"/>
    <process expanded="true">
      <operator activated="true" class="text:process_document_from_file"
compatibility="7.5.000" expanded="true" height="82" name="Process Documents from
Files" width="90" x="45" y="34">
        <list key="text_directories">
          <parameter key="WinterGames" value="C:\Users\Stephen\Documents\Semester
7\textAnalytics(2018)\Assignment\WinterGames"/>
          <parameter key="SummerGames" value="C:\Users\Stephen\Documents\Semester
7\textAnalytics(2018)\Assignment\SummerGames"/>
          <parameter key="HorrorReviews" value="C:\Users\Stephen\Documents\Semester
7\textAnalytics(2018)\Assignment\HorrorReviews"/>
        </list>
        <parameter key="file_pattern" value="*"/>

        <parameter key="extract_text_only" value="true"/>

        <parameter key="use_file_extension_as_type" value="true"/>

        <parameter key="content_type" value="txt"/>

        <parameter key="encoding" value="SYSTEM"/>

        <parameter key="create_word_vector" value="true"/>

        <parameter key="vector_creation" value="Term Frequency"/>

        <parameter key="add_meta_information" value="true"/>

        <parameter key="keep_text" value="false"/>

        <parameter key="prune_method" value="absolute"/>

        <parameter key="prune_below_percent" value="3.0"/>

        <parameter key="prune_above_percent" value="30.0"/>

        <parameter key="prune_below_absolute" value="3"/>

        <parameter key="prune_above_absolute" value="19"/>

        <parameter key="prune_below_rank" value="0.05"/>

        <parameter key="prune_above_rank" value="0.95"/>

        <parameter key="datamanagement" value="double_sparse_array"/>

        <parameter key="data_management" value="auto"/>

        <process expanded="true">

          <operator activated="true" class="text:tokenize" compatibility="7.5.000"
expanded="true" height="68" name="Tokenize (2)" width="90" x="45" y="34">

            <parameter key="mode" value="non letters"/>

            <parameter key="characters" value=".:"/>

            <parameter key="language" value="English"/>

            <parameter key="max_token_length" value="3"/>

          </operator>
        <operator activated="false" class="text:generate_n_grams_characters"
compatibility="7.5.000" expanded="true" height="68" name="Generate n-Grams
(Characters)" width="90" x="45" y="238">

          <parameter key="length" value="3"/>

          <parameter key="keep_terms" value="false"/>

        </operator>

        <operator activated="true" class="text:transform_cases" compatibility="7.5.000"
expanded="true" height="68" name="Transform Cases" width="90" x="179" y="34">

          <parameter key="transform_to" value="lower case"/>

        </operator>

        <operator activated="false" class="text:generate_n_grams_terms"
compatibility="7.5.000" expanded="true" height="68" name="Generate n-Grams
(Terms)" width="90" x="45" y="340">

          <parameter key="max_length" value="2"/>

        </operator>

        <operator activated="true" class="text:filter_stopwords_english"
compatibility="7.5.000" expanded="true" height="68" name="Filter Stopwords
(English)" width="90" x="313" y="34"/>

        <operator activated="true" class="text:filter_stopwords_dictionary"
compatibility="7.5.000" expanded="true" height="82" name="Filter Stopwords
(Dictionary)" width="90" x="447" y="34">

          <parameter key="file" value="C:\Users\Stephen\Documents\Semester
7\textAnalystics(2018)\Assignment\StopWords\stopwords.txt"/>

          <parameter key="case_sensitive" value="false"/>

          <parameter key="encoding" value="SYSTEM"/>

        </operator>

        <operator activated="true" class="text:filter_by_length" compatibility="7.5.000"
expanded="true" height="68" name="Filter Tokens (by Length)" width="90" x="112"
y="136">

          <parameter key="min_chars" value="2"/>

          <parameter key="max_chars" value="25"/>
        </operator>

        <operator activated="false" class="text:filter_tokens_by_pos"
compatibility="7.5.000" expanded="true" height="68" name="Filter Tokens (by POS
Tags)" width="90" x="179" y="238">

          <parameter key="language" value="English"/>

          <parameter key="expression" value="N.*| V.*"/>

          <parameter key="invert_filter" value="false"/>

        </operator>

        <operator activated="true" class="text:stem_dictionary" compatibility="7.5.000"
expanded="true" height="82" name="Stem (Dictionary)" width="90" x="246" y="136">

          <parameter key="file" value="C:\Users\Stephen\Documents\Semester
7\textAnalystics(2018)\Assignment\synonyms\synonyms.txt"/>

        </operator>

        <operator activated="false" class="text:stem_porter" compatibility="7.5.000"
expanded="true" height="68" name="Stem (Porter)" width="90" x="179" y="340"/>

        <operator activated="true" class="text:stem_lovins" compatibility="7.5.000"
expanded="true" height="68" name="Stem (Lovins)" width="90" x="380" y="136"/>

        <connect from_port="document" to_op="Tokenize (2)" to_port="document"/>

        <connect from_op="Tokenize (2)" from_port="document" to_op="Transform
Cases" to_port="document"/>

        <connect from_op="Transform Cases" from_port="document" to_op="Filter
Stopwords (English)" to_port="document"/>

        <connect from_op="Filter Stopwords (English)" from_port="document"
to_op="Filter Stopwords (Dictionary)" to_port="document"/>

        <connect from_op="Filter Stopwords (Dictionary)" from_port="document"
to_op="Filter Tokens (by Length)" to_port="document"/>

        <connect from_op="Filter Tokens (by Length)" from_port="document"
to_op="Stem (Dictionary)" to_port="document"/>

        <connect from_op="Stem (Dictionary)" from_port="document" to_op="Stem
(Lovins)" to_port="document"/>
```

```
        <connect from_op="Stem (Lovins)" from_port="document" to_port="document 1"/>

        <portSpacing port="source_document" spacing="0"/>

        <portSpacing port="sink_document 1" spacing="0"/>

        <portSpacing port="sink_document 2" spacing="0"/>

      </process>

    </operator>

    <operator activated="true" class="store" compatibility="7.6.001" expanded="true" height="68" name="Store" width="90" x="112" y="289">

      <parameter key="repository_entry" value="//TextMinning/lab3/WordListProject"/>

    </operator>

    <operator activated="true" class="concurrency:cross_validation" compatibility="7.6.001" expanded="true" height="145" name="Validation" width="90" x="246" y="34">

      <parameter key="split_on_batch_attribute" value="false"/>

      <parameter key="leave_one_out" value="false"/>

      <parameter key="number_of_folds" value="10"/>

      <parameter key="sampling_type" value="stratified sampling"/>

      <parameter key="use_local_random_seed" value="false"/>

      <parameter key="local_random_seed" value="1992"/>

      <parameter key="enable_parallel_execution" value="true"/>

      <process expanded="true">

        <operator activated="true" class="concurrency:parallel_decision_tree" compatibility="7.6.001" expanded="true" height="82" name="Decision Tree" width="90" x="112" y="34">

          <parameter key="criterion" value="gain_ratio"/>

          <parameter key="maximal_depth" value="20"/>

          <parameter key="apply_pruning" value="true"/>

          <parameter key="confidence" value="0.25"/>
          <parameter key="apply_prepruning" value="true"/>

          <parameter key="minimal_gain" value="0.1"/>

          <parameter key="minimal_leaf_size" value="2"/>

          <parameter key="minimal_size_for_split" value="4"/>

          <parameter key="number_of_prepruning_alternatives" value="3"/>

        </operator>

        <operator activated="false" class="naive_bayes" compatibility="7.6.001"
expanded="true" height="82" name="Naive Bayes" width="90" x="179" y="340">

          <parameter key="laplace_correction" value="true"/>

        </operator>

        <operator activated="false" class="k_nn" compatibility="7.6.001"
expanded="true" height="82" name="k-NN" width="90" x="179" y="238">

          <parameter key="k" value="1"/>

          <parameter key="weighted_vote" value="false"/>

          <parameter key="measure_types" value="MixedMeasures"/>

          <parameter key="mixed_measure" value="MixedEuclideanDistance"/>

          <parameter key="nominal_measure" value="NominalDistance"/>

          <parameter key="numerical_measure" value="EuclideanDistance"/>

          <parameter key="divergence" value="GeneralizedIDivergence"/>

          <parameter key="kernel_type" value="radial"/>

          <parameter key="kernel_gamma" value="1.0"/>

          <parameter key="kernel_sigma1" value="1.0"/>

          <parameter key="kernel_sigma2" value="0.0"/>

          <parameter key="kernel_sigma3" value="2.0"/>

          <parameter key="kernel_degree" value="3.0"/>

          <parameter key="kernel_shift" value="1.0"/>

          <parameter key="kernel_a" value="1.0"/>
```

```
<parameter key="kernel_b" value="0.0"/>
</operator>
<connect from_port="training set" to_op="Decision Tree" to_port="training set"/>
<connect from_op="Decision Tree" from_port="model" to_port="model"/>
<connect from_op="Decision Tree" from_port="exampleSet" to_port="through
1"/>
<portSpacing port="source_training set" spacing="0"/>
<portSpacing port="sink_model" spacing="0"/>
<portSpacing port="sink_through 1" spacing="0"/>
<portSpacing port="sink_through 2" spacing="0"/>
<description align="left" color="green" colored="true" height="80" resized="true"
width="248" x="37" y="137">In the training phase, a model is built on the current
training data set. (90 % of data by default, 10 times)</description>
<description align="center" color="yellow" colored="false" height="105"
resized="false" width="180" x="336" y="246">Type your comment</description>
</process>
<process expanded="true">
<operator activated="true" class="apply_model" compatibility="7.6.001"
expanded="true" height="82" name="Apply Model" width="90" x="45" y="34">
<list key="application_parameters"/>
<parameter key="create_view" value="false"/>
</operator>
<operator activated="true" class="performance" compatibility="7.6.001"
expanded="true" height="82" name="Performance" width="90" x="179" y="34">
<parameter key="use_example_weights" value="true"/>
</operator>
<connect from_port="model" to_op="Apply Model" to_port="model"/>
<connect from_port="test set" to_op="Apply Model" to_port="unlabelled data"/>
```

```
          <connect from_op="Apply Model" from_port="labelled data"
to_op="Performance" to_port="labelled data"/>

          <connect from_op="Performance" from_port="performance"
to_port="performance 1"/>

          <connect from_op="Performance" from_port="example set" to_port="test set
results"/>

          <portSpacing port="source_model" spacing="0"/>

          <portSpacing port="source_test set" spacing="0"/>

          <portSpacing port="source_through 1" spacing="0"/>

          <portSpacing port="source_through 2" spacing="0"/>

          <portSpacing port="sink_test set results" spacing="0"/>

          <portSpacing port="sink_performance 1" spacing="0"/>

          <portSpacing port="sink_performance 2" spacing="0"/>

          <description align="left" color="blue" colored="true" height="103" resized="true"
width="315" x="38" y="137">The model created in the Training step is applied to the
current test set (10 %).&lt;br/&gt;The performance is evaluated and sent to the
operator results.</description>

        </process>

        <description align="center" color="transparent" colored="false" width="126">A
cross-validation evaluating a decision tree model.</description>

      </operator>

      <operator activated="true" class="store" compatibility="7.6.001" expanded="true"
height="68" name="Store (2)" width="90" x="514" y="238">

        <parameter key="repository_entry" value="DecisionTreeProject"/>

      </operator>

      <connect from_op="Process Documents from Files" from_port="example set"
to_op="Validation" to_port="example set"/>

      <connect from_op="Process Documents from Files" from_port="word list"
to_op="Store" to_port="input"/>

      <connect from_op="Store" from_port="through" to_port="result 4"/>
      <connect from_op="Validation" from_port="model" to_op="Store (2)"
to_port="input"/>
      <connect from_op="Validation" from_port="example set" to_port="result 1"/>
      <connect from_op="Validation" from_port="test result set" to_port="result 2"/>
      <connect from_op="Validation" from_port="performance 1" to_port="result 3"/>
      <connect from_op="Store (2)" from_port="through" to_port="result 5"/>
      <portSpacing port="source_input 1" spacing="0"/>
      <portSpacing port="sink_result 1" spacing="0"/>
      <portSpacing port="sink_result 2" spacing="0"/>
      <portSpacing port="sink_result 3" spacing="0"/>
      <portSpacing port="sink_result 4" spacing="0"/>
      <portSpacing port="sink_result 5" spacing="0"/>
      <portSpacing port="sink_result 6" spacing="0"/>
    </process>
  </operator>
</process>
```