STAT5703 – Data Mining

Research Project on Text Mining of Research Papers

Submitted by:

Abiola Smith

Brendan Maher

Deepesh Khaneja

April 1, 2017

Problem Statement

With the easy access to internet, an immeasurable amount of digital data is available online and particularly in unstructured format from the increasing number of social media platforms, online repositories and websites. Discovering utile information and interesting knowledge from unstructured data is a challenge and has attracted a great deal of attention resulting into more advanced techniques called text mining. Students, researchers, academia on an average spend plenty of time in searching for research papers from online repositories. Search engines facilitate the information of interest and precisely, research papers are found by entering some keywords like title, author, publication, abstract.

This fetched result is based on these parameters and neglect some of the useful information that could classify the results and paper searching. With the availability of thousands of research papers on multiple websites, magazines and journals, the search result produced using snowball effect provides extra information which may be of no use. Therefore, it takes hours to search papers for specific topic. Lack of structural format reinforces text mining that deals with interesting or not so easily retrievable patterns in the documents which are not apparent. Online scientific paper repositories require the user to fill out a form detailing title, author, article number and other keywords before uploading any document. If a user excludes some keywords then it would be difficult and time consuming to search the paper no matter which search engine is being used.

There is a need for models dedicated to topics on research in the fields of engineering, statistics and programming languages, which make use of data mining algorithms to decrease the response time. It should be on the landing page of research journal websites, built-0in to the search feature for locating specific articles. The primary objective is to reduce the time spent on searching for papers I.e. the response time in locating them.

Literature Review

Text Mining also termed as Knowledge Discovery is the process of retrieving some interesting patterns or trends from the already known, perceivable and understandable massive and unstructured datasets [4]. Data mining and text mining have some similarities but the difference lies in the approach of handling data where the former deals with structured format. With the challenge of dealing with unstructured data, the commercial value of text mining has increased compared to data mining as all the documents in an organization are in unstructured, text format. There are innumerable online knowledge repositories on the internet, which are updated frequently with minimum effort. The applications of text mining evolve from these unstructured text repositories to extract useful information. The summary of some literary works using text mining are discussed further in this section.

The precise framework of text mining consists of two components involving text preprocessing includes text refining to semi structured or structured intermediate format and second component involves filtering out the knowledge by deducing some patterns or trends [5]. In paper [5], the authors summarized some commonly used text mining techniques like summarization, clustering, Naives Bayesian Classifier, Nearest Neighbor method, Visualization and Information Extraction which are the subparts of data mining. They also mentioned the application of using text mining in classifying news as text, analysis of the market trends to better operate the business and economy and analysis of junk emails using classification technique.

Text Mining has diverse applications for text classification and one of the applications we followed is searching research papers using clustering and text mining [6]. The authors in [6] developed a search engine for classification of research papers optimally dealing with database involving programming and operating systems. The approach followed is to create a knowledge database and using it in the architecture for construing patterns relevant to each sentence. A filter and wrapper is designed to classify the research papers during searching phase and the whole architecture is implemented on MySql database. They also followed K-means, the most common technique to cluster groups.

In paper [7], the authors used Wikipedia's search interface to find the links of the articles and group them based on some similarity measure using clustering for a particular keyword. They conducted four different experiments to cluster links depending on the HTML content of the document and labelled them in a tag for searching meaningful results. The primary objective to use text mining is the returned results are not listed categorically by Wikipedia having similarity and the links of similar data are also not in the juxtaposition of search results. The methodology follows the implementation of K-means to group search results along with feature extraction method- TF-IDF (Term Frequency- inverse Term Frequency) matrix to find important terms and group them from a given document. The labelling of the groups is performed by Latent Dirichlet Allocation(LDA). The experiments in [7] were conducted to determine the efficient clustering algorithm, also to analyze the TF-IDF scores distribution to decide number of clusters based on top terms and lastly the accuracy of the clustering techniques was analyzed for labelling.

The general framework for any text mining application in any paper consists of data collection, data preprocessing including tokenization, stemming, filtering, removal of stop words and hashtags. The processed data is used for feature extraction using TF-IDF, Support Vector Machines and if required sophisticated classification algorithms are used to train and predict data before pattern Analysis.

In another literary work [8], the authors used a hybrid algorithm for the automatic classification of research papers. The primary objective of the study was to reduce the time for searching any research journal from the online repository and this research project follows the same objective. The hybrid algorithm uses unsupervised learning algorithm i.e. K- means to cluster the data and classify them using classifier before uploading to any repository. The input research papers are passed to feature extraction unit using TF-IDF concept and applied clustering algorithm during training phase. The fetched results are passed to classifier to obtain labels for research papers. The results from the hybrid algorithm were compared with KNN (k-nearest neighbor) and proved much more efficient in predicting the classes of research papers.

Text Mining Overview

Text Mining is the process of extracting previously unknown, understandable, potential and practical patterns or knowledge from a collection of unstructured text data or corpus. (Zhang et al, 2015)

Good text mining methods use just enough information to build a general model and ignore the data that cannot be utilized to make accurate predictions. This is achieved by looking for major features that define data from a training set and disregarding features that are rare or unusual. Such rare features are not useful in a general model.

In today's organization, up to 80% of information is stored in the form of unstructured text and is usually stored in several repositories in several formats – this presents a huge challenge for mining and analysis. This is also complicated by the fact that text is high dimensional as well - a vector representation of a document uses one dimension per unique word.

Applications of Text Mining

Some areas in which Text Mining is used include

Business

- Surveys
  Responses to open-ended questions are classified into various categories based on the nature of the text and the results used to inform corporate decisions
- Intelligence
  Companies perform frequent checks of competitors' websites to gather information related to their operations, e.g. strategic partnerships, new product releases and investments
- Intellectual Property
  Comparing new product descriptions with filed patents to identify infringement
- Customer Management
  Using customer feedback to inform improvements in customer relationship and to identify actions that would lead to better customer satisfaction and engagement
- Social media analysis – preferences, behaviors, sentiments
- E-Recruitment
  Resume filtering, job search

Medicine and Law

- Medical research
- Legal research

Society

- Monitoring public opinion
- Shopping
- Academic research
- Automated grading
- Topic Modelling
- Digital libraries

Text Mining Techniques and Related Algorithms

Document clustering (phrase clustering)

Document clustering involves grouping documents according to the keywords they contain. Keywords generally describe the contents of the documents. A cluster of documents share a set of keywords that co-occur in the document text. Documents in a cluster may possibly be related to a common topic and are represented by a sequence of keywords and associated weights. On the other hand, if keywords are the objects in the collection, then each keyword is represented by a set of documents in which the keyword occurs. A cluster of keywords is created from the keywords that have overlapping lists of documents.

One of the main applications of Clustering is to find related documents for any given subject. For a large document collection, a clustering algorithm may create hundreds of clusters in an initial pass and must be run several times to generate a useful result. We can specify controlling parameters for the clustering algorithm such as:

- minimum and maximum size of clusters
- matching threshold value for including documents in a cluster
- the degree of overlap between clusters
- maximum number of clusters

Some algorithms used for document clustering are:

- K-means
- Hierarchical clustering
- Frequent item-set based clustering
- Seeded clustering
- Ontology based clustering
- Concept-based clustering
- Constraint-based clustering

K-means and Hierarchical clustering are demonstrated in our example.

Document classification (text classification, document standardization, topic modelling)

This technique refers to sorting a collection of documents into pre-defined categories. Applications of document classification are seen in:

- Email (spam) filtering
- Sentiment analysis
- Language identification
- Genre classification

Algorithms commonly used in document classification include:

- K-nearest neighbor
- Naïve Bayesian
- Rocchio
- Centroid based
- Support Vector Machine
- Decision tree algorithm
- LDA
- LSI

Natural language processing (spelling correction, sentiment analysis, lemmatization, grammatical parsing, and word sense disambiguation)

This technique, a form of Artificial Intelligence that can understand human language.

The following algorithms are commonly used in NLP applications.

- C-Support Vector Classification
- Decision Tree Classifier
- k-nearest neighbor classifier
- Neural networks

Information extraction (relationship extraction / link analysis)

This technique is used to identify relationships and connections between various types of entities.

The Text Mining Process

Regardless of the technique employed to conduct the analysis of the corpus, the Text Mining process entails several steps which are outlined below.

1. Data Collection / Extraction
   Extraction of the text from its original source such as word documents, blogs, web pages and repositories

2. Text Processing / Pre-processing
   This step involves:

   a. Removal of common words, known as stop words (and, the, a, at, all, etc.), removal of numbers, capitalization punctuation
   b. Removal of other words that may appear frequently in the text but may not be valuable in the analysis being performed
   c. Stemming of words – removal of endings such as “ing”, “ed”, “es”, etc.
   d. Combining words that should stay together, for example “qualitative research”, “research methods”
   e. Removal of white spaces that were left after words were removed

3. Data Division
   The data is divided into training and testing sets, like the separation done in other forms of data mining

4. Feature Extraction
   In this step, a document frequency matrix is created which contains document number in the rows and features or words in the columns. The value in the cells corresponds to the number of times a word appears in the document.

5. Feature /Pattern Selection
   The matrix generated in the previous step will have a long list of features (Curse of dimensionality), some of which may not be useful in the analysis. The matrix is also very sparse as many words only appear in a few documents. To reduce the dimensionality and create a more useful result, methods such as Chi Squared, Term Frequency and Document Frequency are used to select the features with the highest frequency.

6. Model Training
   In this step, the algorithm is trained using the matrix produced in the previous step. The algorithm selected will be determined by the mining technique.

7. Model Evaluation
   The model is evaluated by applying the algorithm to the test data

Applying Text Mining Clusters to a given dataset

The dataset is a csv file which contains 494 abstracts from various journals. The data is treated using the tidytext package since automatically any punctuation is gotten rid of, and all single words or word pairs are split up and made lowercase.
Next all Unicode is removed, and numbers are dropped as well as words shorter than three letters and common generic words from stopwords("english").
Last the words are stemmed of prefixes and suffixes so to collapse synonyms.

From a tidytext of single words, and word counts over 10 from the whole set, gives a word cloud showing **imag, method, comput, algorithim, data**, to be the most common words in the corpus.

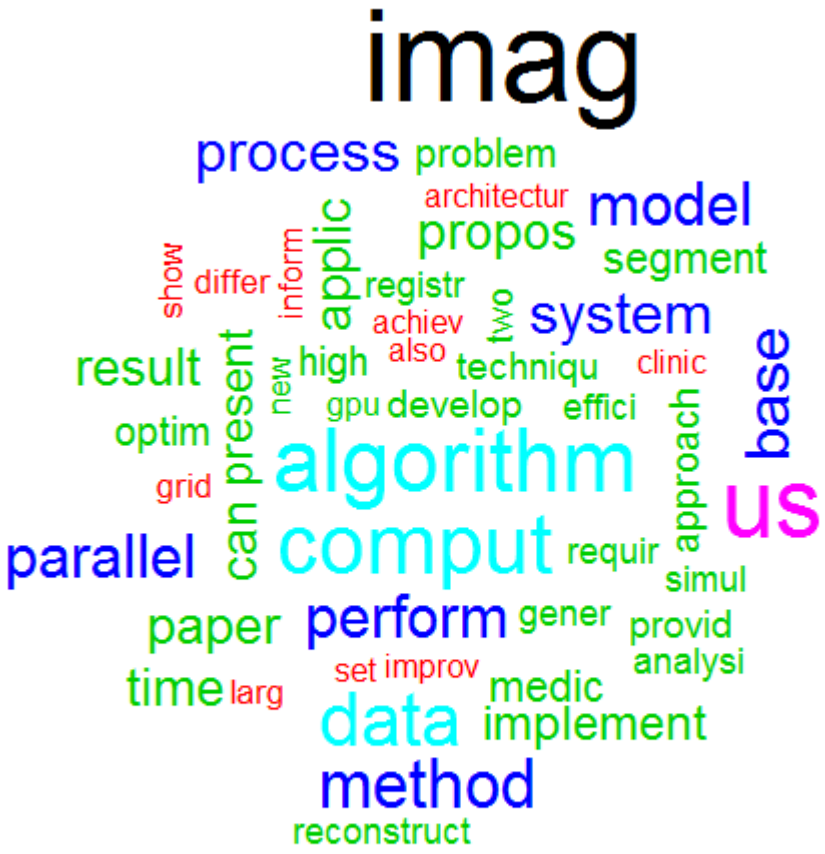

Figure 1: Word cloud of the terms occurring most frequently in the corpus.

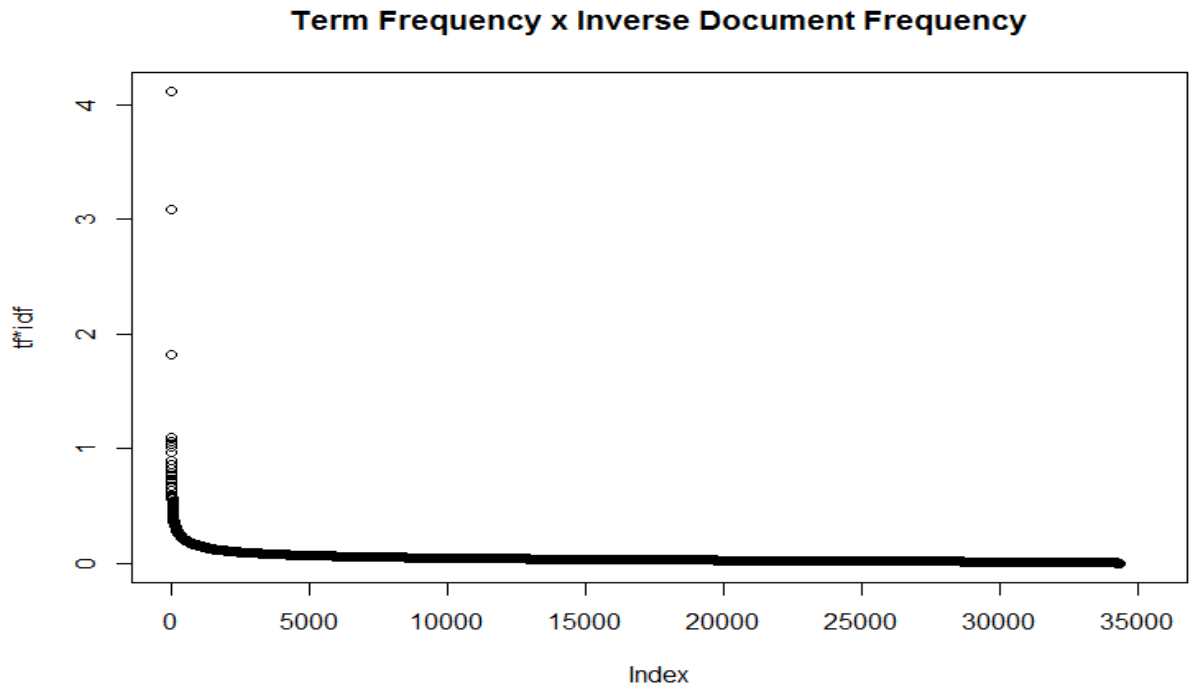

Figure 2: Quickly Decreasing TFxIDF values

From a tidytext of single words, a count of each word for each document is made. The term frequency per document (tf) is calculated for common words. The inverse document frequency (idf, being the ln of the total docs over docs containing the term), is calculated for unique terms. The tf times the idf gives terms which are common but not too common so making a plot shows values concentrating at low frequencies, and terms which are both frequent and unique to be rare. By taking terms by the tf times idf, gives just 40 terms over one half.

id	abs	n	tf	idf	tf_idf	id	abs	n	tf	idf	tf_idf
245	stripe	10	0.666667	6.173786	4.115857	331	shoulder	7	0.102941	6.173786	0.635537
269	dimm	16	0.5	6.173786	3.086893	207	epr	7	0.097222	6.173786	0.600229
245	width	6	0.4	4.564348	1.825739	33	grid	4	0.266667	2.24196	0.597856
236	agent	18	0.24	4.564348	1.095444	17	histologi	4	0.108108	5.480639	0.592502
269	tree	10	0.3125	3.401197	1.062874	86	cell	10	0.2	2.95491	0.590982
245	storag	5	0.333333	3.129264	1.043088	269	throughput	5	0.15625	3.775891	0.589983
85	beamform	21	0.168	6.173786	1.037196	93	pilot	8	0.123077	4.787492	0.58923
245	disk	3	0.2	5.075174	1.015035	218	restor	10	0.114943	5.075174	0.583353
323	rlde	5	0.15625	6.173786	0.964654	458	colonoscopi	8	0.112676	5.075174	0.571851
52	carotid	10	0.144928	6.173786	0.894752	231	mobil	5	0.135135	4.227876	0.571335
58	tuu	8	0.140351	6.173786	0.866496	102	fring	5	0.090909	6.173786	0.561253
245	multiprogram	2	0.133333	6.173786	0.823172	245	choic	2	0.133333	4.094345	0.545913
85	fir	16	0.128	6.173786	0.790245	222	gp	5	0.087719	6.173786	0.54156
86	death	8	0.16	4.787492	0.765999	345	mmm	6	0.098361	5.480639	0.539079
245	secondari	2	0.133333	5.480639	0.730752	159	quantiz	6	0.109091	4.787492	0.522272
33	workflow	3	0.2	3.534729	0.706946	245	arrai	2	0.133333	3.871201	0.51616
385	simbo	15	0.114504	6.173786	0.706922	245	optim	5	0.333333	1.539057	0.513019
245	overlai	2	0.133333	5.075174	0.67669	141	worm	6	0.082192	6.173786	0.507435
52	arteri	12	0.173913	3.871201	0.673252	245	overhead	2	0.133333	3.775891	0.503452
116	seed	5	0.15625	4.094345	0.639741	409	prime	12	0.081081	6.173786	0.500577

Word associations were also found between terms. In the plot below the results display the pairs and groups of terms that have correlation greater than 15% through values over 0.5 of tf times idf.

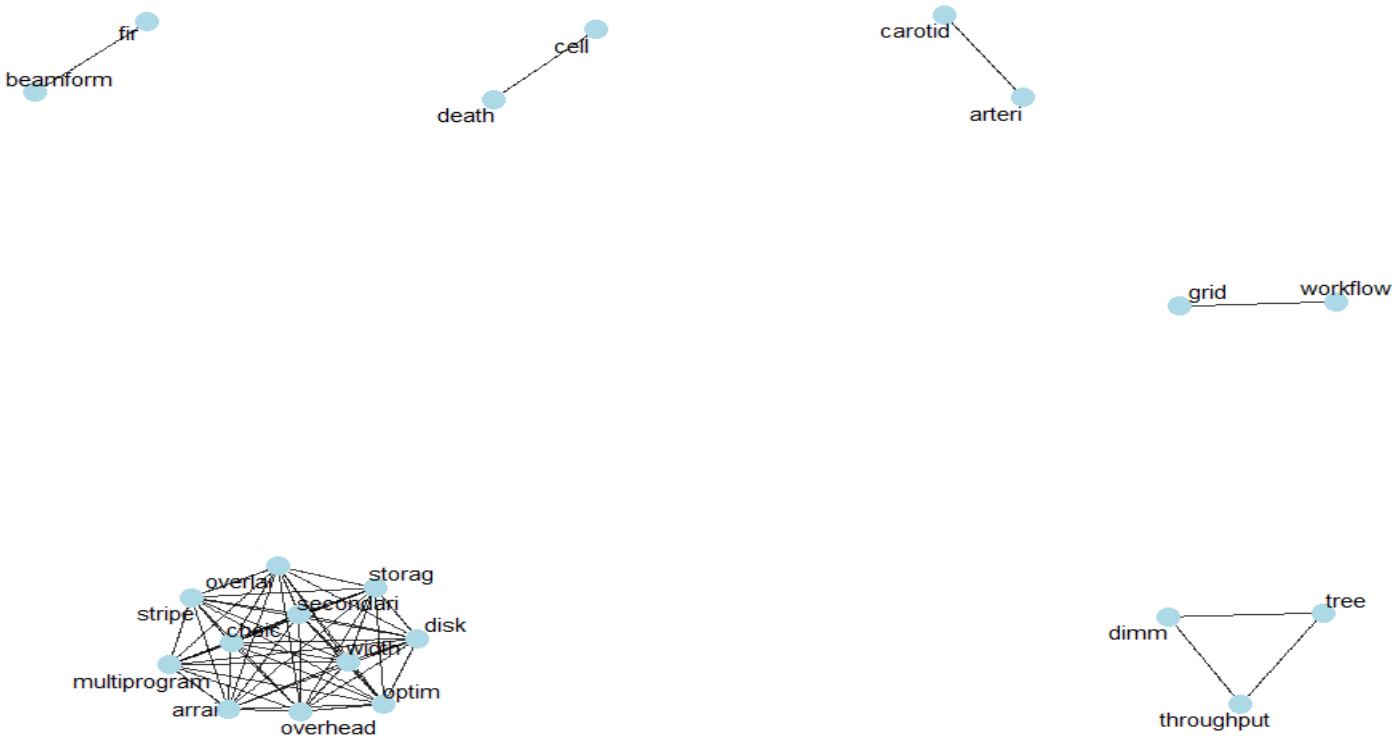

Figure 3: Word correlations throughout the corpus

A document term matrix can now be made, where each document term pair has a tf times idf value. The document term matrix is very sparse. Removing sparse terms to have less then 0.70% leaves a matrix with 20 non-sparse terms. Clusters are now made either bottom up by hierarchy or top down by k-means.

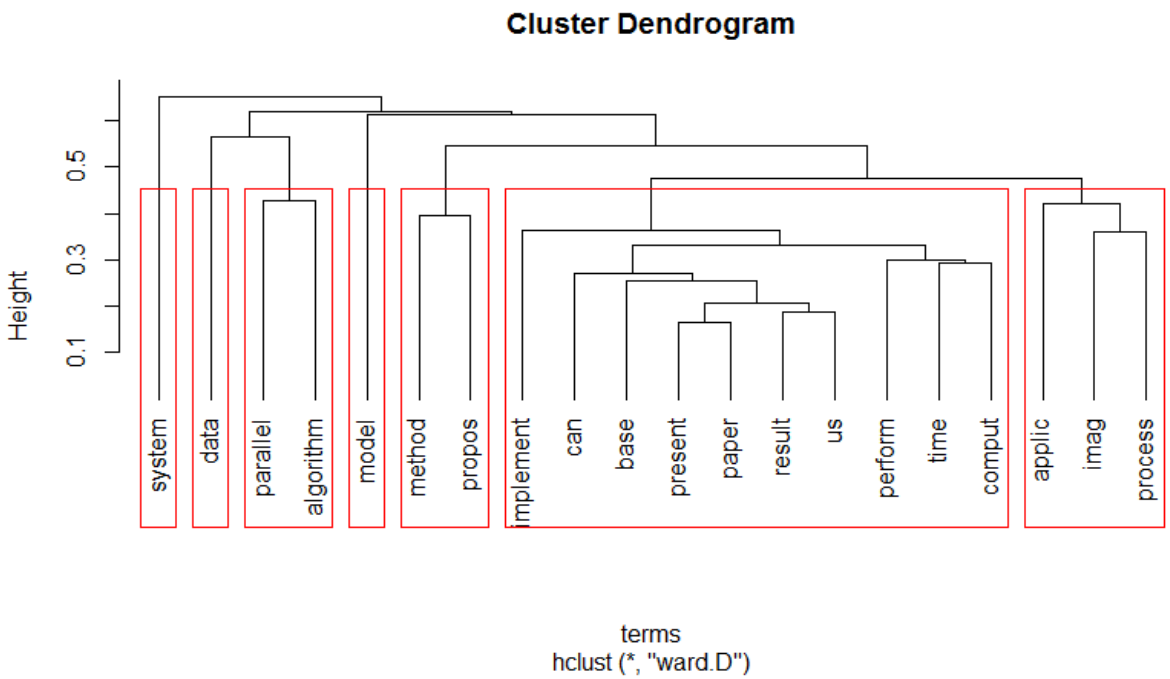

Figure 4: Dendrogram showing clusters of most frequently occurring terms

Taking values bottom up by hclust and clustering by term, shows a dendrogram with 7 clusters. Taking values top down by k-means and clustering by term shows a cluster plot with 51% variability explained. This plot has 3 clusters.

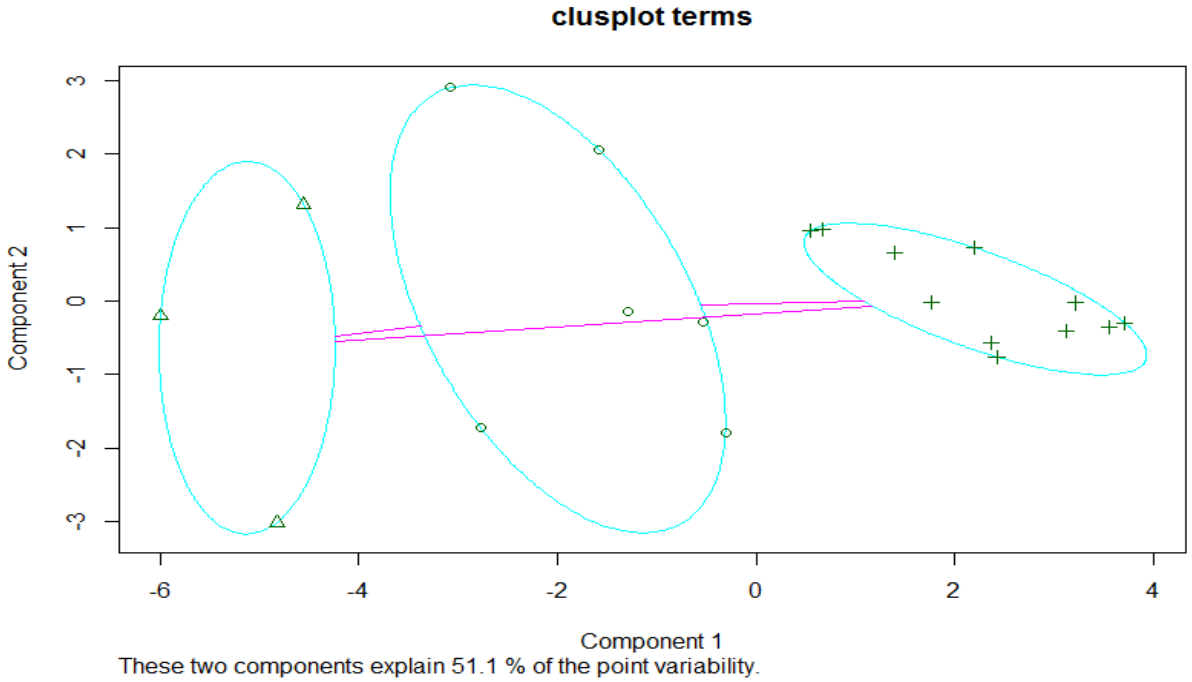

Figure 5: K-means clusters of the most frequently occurring terms

Taking values bottom up by hclust and clustering by document, shows a dendrogram with 7 clusters.

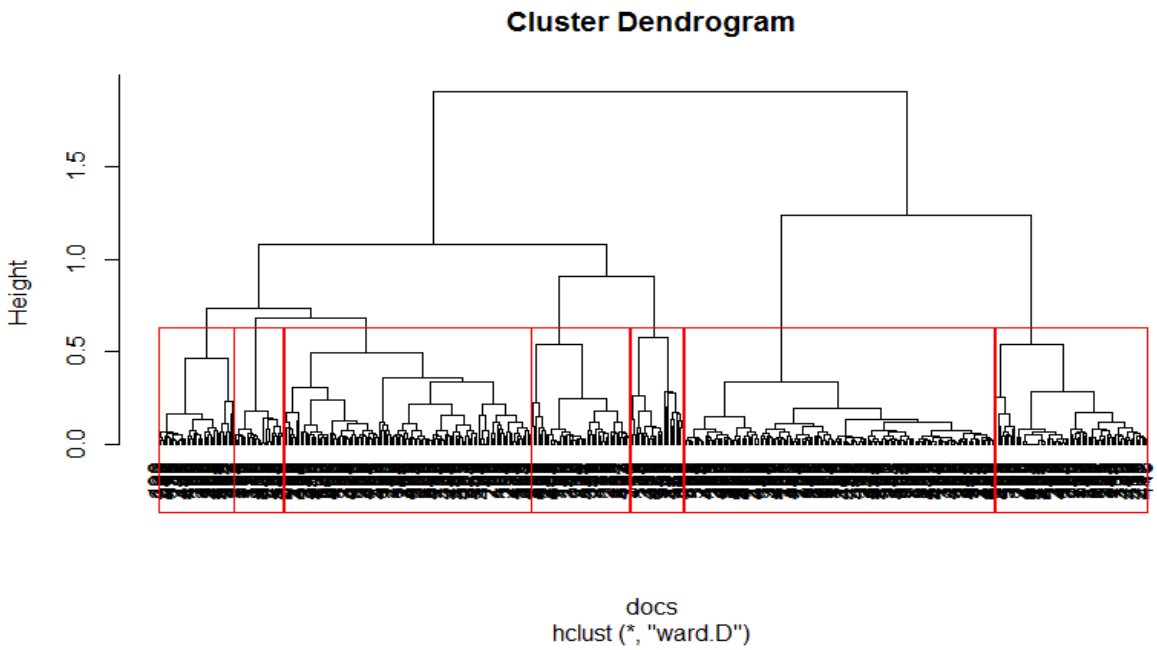

Figure 6: Dendrogram showing clusters of documents based on keyword frequency

Taking values top down by k-means and clustering by document, shows a cluster plot with 86% variability explained. This plot shows 3 clusters.

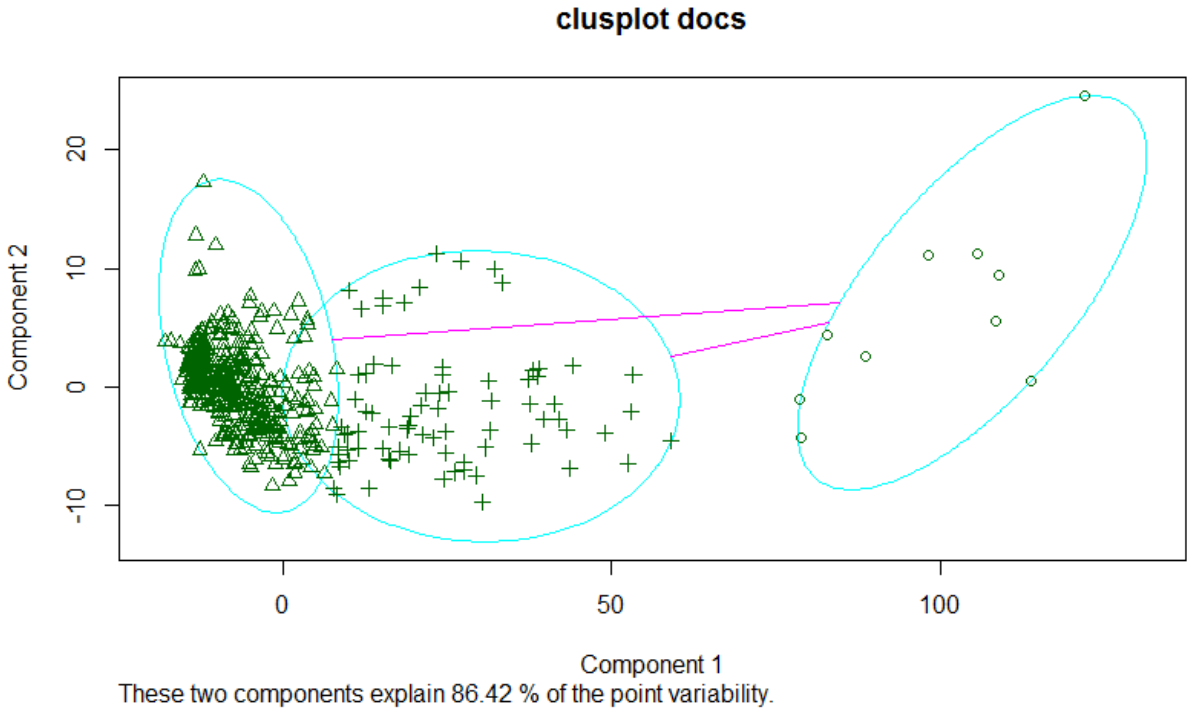

Figure 7: K-means plot showing clusters of documents based on keyword frequency

Taking the top term of each cluster by the document term matrix, gives ***model*** from Biomedical Signal Processing and Control, and ***parallel*** from Journal of Parallel and Distributed Computing, and ***imag*** from Computer Methods and Programs in Biomedicine. This step gives the topic key words for each cluster.

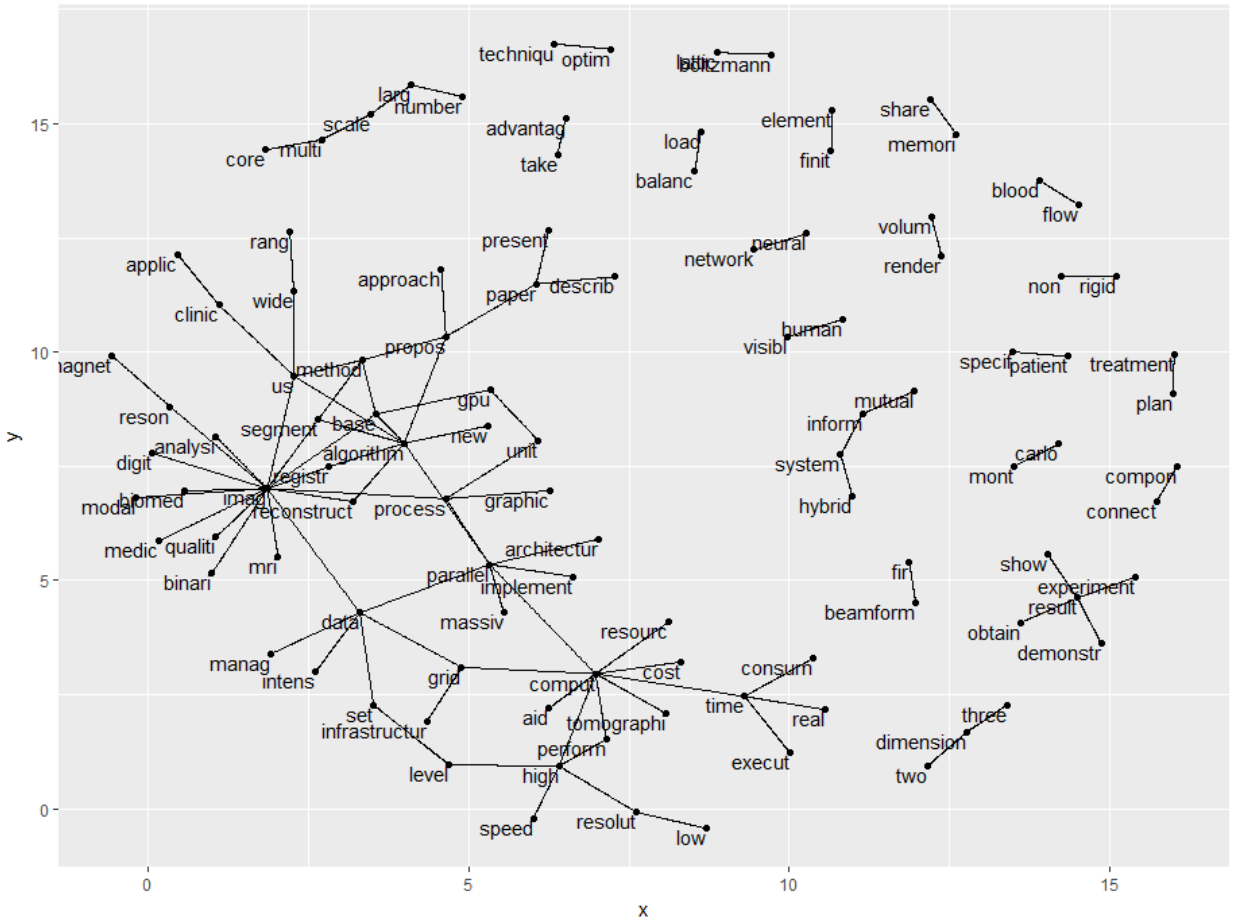

Figure 8: Plot showing the most frequently occurring keywords from each cluster

From a tidytext of word pairs, counting across the whole dataset gives 98 pairs of terms that have over 10 occurrences, with ***imag*** and ***algorithm*** and ***comput*** being the words that have the highest correlations.

For weights, term frequency times inverse document frequency works best. For clustering, dendrogram read better since they don't have to deal with high dimensionality. For word associations, neighbor pair frequencies seem to work just as well as correlations.

The R code used to produce the above results is provided in the Appendix of this paper. (R version 3.3.3 is required to run the code correctly).

References

1. Carlos A.S.J. Gulo and Thiago R.P.M. R´ubio. 2015. ***Text Mining Scientific Articles using the R Language***. Proceedings of the 10th Doctoral Symposium in Informatics Engineering - DSIE'15
2. Khorsheed, M.S. & Al-Thubaity, A.O. 2013. Comparative evaluation of text classification techniques using a large diverse Arabic dataset. ***Language Resources & Evaluation***. 47: 513.
3. Konchady, Manu. 2006. ***Text Mining Application Programming***. Cengage Charles River Media.
4. Prabha, S. Duraiswamy, K. Sharmila, M. 2014. Analysis of Different Clustering Techniques in Data and Text Mining. ***International Journal of Computer Science Engineering***. 3(2)
5. Ramzan Talib, Muhammad Kashif Hanif, Shaeela Ayesha and Fakeeha Fatima. 2016. Text Mining: Techniques, Applications and Issue. ***International Journal of Advanced Computer Science and Applications***, 7(11)
6. Y. Zhang, M. Chen and L. Liu, "A review on text mining," *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, 2015[5] Techniques on Text Mining
7. M. Sukanya and S. Biruntha, "Techniques on text mining," *2012 IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, Ramanathapuram, 2012
8. E. A. Calvillo, A. Padilla, J. Muñoz, J. Ponce and J. T. Fernandez, "Searching research papers using clustering and text mining," *CONIELECOMP 2013, 23rd International Conference on Electronics, Communications and Computing*, Cholula
9. K. D. C. G. Kapugama, S. A. S. Lorensuhewa and M. A. L. Kalyani, "Enhancing Wikipedia search results using Text Mining," *2016 Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, Negombo, 2016
10. Kaur, Er. Rajvir, Er. Nishi. "Automated Classification Of Research Papers Using Hybrid Algorithm". *International Journal of Hybrid Information Technology* 8.6 (2015): 95-106. Web
11. http://tidytextmining.com/tidytext.html
12. https://rstudio-pubs-static.s3.amazonaws.com/31867_8236987cf0a8444e962ccd2aec46d9c3.html
13. https://github.com/carlosalexsander/ECAC_Project

Appendix A

```
libs=c("readr","dplyr","tidytext","tm","SnowballC","wordcloud","cluster","tidyr","widyr","igraph","ggraph");
rbind(t(lapply(libs,require,character.only=TRUE)),libs);                    # libraries used
References=read_csv("F:/ STAT5703ResearchProjectTextMining/CSV_All_Texts.csv");   # load data so change directory accordingly
ref=data_frame(id=References$id,abs=References$Abstract);                   # only the abstracts are being studied
refs=unnest_tokens(ref,abs,abs);                                            # for single terms
refs$abs=sub("[^\x20-\x7E]","",refs$abs);                                   # remove Unicode
refs=anti_join(refs,data_frame(abs=stopwords("en")));                       # remove common English
refs=filter(refs,!grepl("\\d",refs$abs));                                   # remove numbers
refs=filter(refs,nchar(refs$abs)>2);                                        # remove words less then 3 letters
refs$abs=wordStem(refs$abs);                                                # remove prefixes and suffixes
with(count(refs,abs),wordcloud(abs,n,max.words=50,colors=c(2,3,4,5,6,9)));  # word cloud of terms over all documents
refc=count(refs,id,abs,sort=TRUE);                                          # term counts per documents
refc=arrange(bind_tf_idf(refc,abs,id,n),desc(tf_idf));                      # calculate tf, idf, tf*idf
plot(refc$tf,refc$idf,main="terms vs inverse of documents");View(filter(refc,tf_idf>0.5));   # tf vs idf plot
plot(refc$tf_idf,ylab="tf*idf",main="Term Frequency x Inverse Document Frequency");   # plot tf*idf and view tf*idf over 0.5
refr=pairwise_cor(filter(refc,tf_idf>0.5),abs,id,n,sort=TRUE);              # term correlations from values over 0.5
refr=graph_from_data_frame(filter(refr,correlation>0.15));                  # graph correlations within docs over 0.15
ggraph(refr,layout="fr")+geom_edge_link(aes(edge_alpha=correlation),show.legend=FALSE)+geom_node_point(color="lightblue",size=5)
    +geom_node_text(aes(label=name),repel=TRUE)+theme_void();
refm=removeSparseTerms(cast_dtm(refc,id,abs,tf_idf),0.7);                   # make document term matrix of values tf*idf
reft=dist(t(refm));refth=hclust(reft,method="ward.D");reftk=kmeans(reft,3);  # cluster by term differences
plot(refth,hang=-1,xlab="terms");rect.hclust(refth,k=7,border="red");        # plot bottom up clusters of terms
clusplot(as.matrix(reft),reftk$cluster,lables=2,main="clusplot terms");     # plot top down clusters of terms
refd=dist(refm);refdh=hclust(refd,method="ward.D");refdk=kmeans(refd,3);    # cluster by document differences
plot(refdh,hang=-1,xlab="docs");rect.hclust(refdh,k=7,border="red");         # plot bottom up clusters of documents
clusplot(as.matrix(refd),refdk$cluster,lables=2,main="clusplot docs");      # plot top down clusters of documents
refo=top_n(group_by(tidy(refm),document),1,count);                          # get the top dtm terms of each doc cluster
refo=inner_join(refo,data_frame(document=refo$document,journal=References$Journal[as.numeric(refo$document)]));
View(top_n(group_by(inner_join(refo,data_frame(clus=refdk$cluster,document=names(refdk$cluster))),clus),1,count));
View(cbind(References[1:10,2:4],refdk$cluster)[1:10,]);                     # A sample of document cluster designations
refp=unnest_tokens(ref,abs,abs,"ngrams",n=2);                               # for term pairs
refp=separate(refp,abs,c("a1","a2"),sep=" ");                               # split for removals
refp$a1=sub("[^\x20-\x7E]","",refp$a1);                                     # remove Unicode of leading terms
refp$a2=sub("[^\x20-\x7E]","",refp$a2);                                     # remove Unicode of following terms
refp=anti_join(refp,data_frame(a1=stopwords("en")));                        # remove common English of leading
refp=anti_join(refp,data_frame(a2=stopwords("en")));                        # remove common English of following
refp=filter(refp,!grepl("\\d",refp$a1)&!grepl("\\d",refp$a2));              # remove numbers
refp=filter(refp,(nchar(refp$a1)>2)&(nchar(refp$a2)>2));                    # remove words less then 3 letters
refp$a1=wordStem(refp$a1);refp$a2=wordStem(refp$a2);                        # remove prefixes and suffixes
refg=graph_from_data_frame(filter(count(refp,a1,a2),n>10));                 # graph term relations
ggraph(refg,layout="fr")+geom_edge_link()+geom_node_point()+geom_node_text(aes(label=name),vjust=1,hjust=1)
```

Appendix B

Shown below, from the R code used, is what the data created looks like

```
.....................................................................................
# A tibble: 494 × 7
      id                                                                    Title...
   <int>                                                                    <chr>...
1      1 ISP: An Optimal Out-of-Core Image-Set Processing Streaming Architecture for...
.....................................................................................
Source: local data frame [35,401 x 6]
Keywords <chr>, Recommend <int>
# A tibble: 50,488 × 2
      id   abs
   <int> <chr>
1      1  imag
.....................................................................................
Source: local data frame [35,401 x 6]
Groups: id [480]
      id      abs     n        tf      idf    tf_idf
   <int>    <chr> <int>     <dbl>    <dbl>     <dbl>
1    245   stripe    10 0.6666667 6.173786 4.1158574
.....................................................................................
IGRAPH DN-- 22 124 --
+ attr: name (v/c), correlation (e/n)
+ edges (vertex names):
 [1] width  ->stripe     storag  ->stripe    disk  ->stripe     multiprogram->stripe...
.....................................................................................
<<DocumentTermMatrix (documents: 480, terms: 20)>>
Non-/sparse entries: 4342/5258
Sparsity           : 55%
Maximal term length: 9
Weighting          : term frequency (tf)
.....................................................................................
Call:hclust(d = reft, method = "ward.D")
Cluster method   : ward.D
Distance         : euclidean
Number of objects: 20
.....................................................................................
model    system    method      data  parallel   perform algorithm      imag ...
    1         1         2         1         2         3         2         2 ...
.....................................................................................
Call:hclust(d = refd, method = "ward.D")
Cluster method   : ward.D
Distance         : euclidean
Number of objects: 480
.....................................................................................
245 269 236  85 323  52  58  86  33 385 116 331 207  17  93 218 458 231 102 222 345 159 ...
1   1   2   3   1   2   2   2   1   3   2   2   3   2   2   2   3   2   1   2   2   2 ...
.....................................................................................
Source: local data frame [469 x 4]
Groups: document [?]
   document  term      count                                                  journal
      <chr> <chr>      <dbl>                                                    <chr>
1       331 model 0.07807341                                 Computers & Geosciences
.....................................................................................
# A tibble: 27,434 × 3
      id      a1    a2
   <int>   <chr> <chr>
1      1    imag popul
.....................................................................................
IGRAPH DN-- 106 98 --
+ attr: name (v/c), n (e/n)
+ edges (vertex names):
 [1] algorithm ->base          algorithm ->us            base ->method         binari ...
.....................................................................................
```