National College of Ireland

A novel text mining approach using a bipartite graph projected onto two dimensions

MSc Research Project
Data Analytics

Stephen Redmond
x15021815

School of Computing
National College of Ireland

Supervisor: Eleni Rozaki

National College of Ireland
Project Submission Sheet – 2015/2016
School of Computing

Student Name:	Stephen Redmond
Student ID:	x15021815
Programme:	Data Analytics
Year:	2017
Module:	MSc Research Project
Lecturer:	Eleni Rozaki
Submission Due Date:	5/8/2017
Project Title:	A novel text mining approach using a bipartite graph projected onto two dimensions
Word Count:	5,799

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:	
Date:	20th July 2017

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
3. Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A novel text mining approach using a bipartite graph projected onto two dimensions

Stephen Redmond
x15021815
MSc Research Project in Data Analytics

20th July 2017

Abstract

The collection of text data is exploding. From blogs to news reports, to helpdesk tickets, there seems to be a never-ending supply of writings. The owners of these data see methods to group texts and look for clusters of topics. Because of the size of the data, solutions that scale on clustered computer solutions are ideal. The traditional term vector approach can lead to the curse-of-dimensionality. Simple solutions are better than complex because it is often necessary to explain the model to either business users or even regulators. This paper demonstrates a method of keyword mining using the graph-of-words technique and classification by projecting the bipartite graph of terms and documents onto two dimensions. This method can be scaled using a cluster computing technology such as Apache Spark, and the results are easily surfaced to users.

1 Introduction

Words shall be known by the company they keep (Mackin; 1978).

Worldwide collections of text documents are exploding. Texts such as academic libraries, news archives, online blogs, even collections of helpdesk tickets, grow all the time. Because of the volume of the data, the owners of the document collections look for ways to categorise the text automatically. With the ever increasing size of the problem, there is a growing need for solutions that will scale, for example, using Big Data systems.

Keyword extraction techniques are essential. The most frequently applied method of extracting keywords is TF/IDF (Robertson; 2004). It is a bag-of-words approach that does not take into consideration the relationship of words within documents (Rousseau and Vazirgiannis; 2013). It also needs to be calculated by counting words across the entire corpus. Such shared-state calculations have been shown to cause problems in Big Data systems (Zhai et al.; 2012).

In order to compare documents, a document-term matrix model is often used. This matrix is sparse and becomes very wide as the amount of text expands. It is suggested in Dhillon (2001) that the matrix can have up to 99% zero values and Dhillon, therefore, proposes a bipartite graph approach. Another issue with such wide dimensions is the potential for the curse-of-dimensionality (Huang et al.; 2005), where distance calculation becomes complicated.

Such wide and sparse vector spaces are not easily demonstrated to stakeholders, such as business leaders or regulators. Attempts to visualise high dimensional data include self-organising maps (Kohonen; 1998) and stochastic neighbour embedding (Hinton and Roweis; 2002). These are useful to an extent, but lose some information that would be of benefit to stakeholders, and can be expensive to calculate.

MapReduce, Hadoop's method of sharing tasks across many computers (Lin and Dyer; 2010) can be a useful technology to use to process significant amounts of text. More recently, in-memory approaches, such as Apache Spark (Zaharia et al.; 2012), deliver much-improved performance. Spark also affords other graph and data processing techniques.

More complicated machine learning methods have been applied to text classification. In Lai et al. (2015), a recurrent convolutional neural network is used to build the classifier. However, in regulated industries, such as banking or insurance, there is a potential difficulty using these complex models as regulators, and other business stakeholders may need to be shown the model details.[1]

This leads to the question: what scalable method can be used to classify significant numbers of documents but is also straightforward enough to explain to a business stakeholder.

This paper presents a research project to develop a text classifier that can be used with any body of documents and can scale well on a distributed computing platform. The result will be the novel method of using graphs to both extract keywords and model the document and term relationship, and then project that bipartite graph onto two dimensions. The extraction of the keywords can be scaled using Apache Spark, and the classification model can be surfaced visually to stakeholders.

The rest of this document is arranged as follows: Section 2 presents a review of the literature. Section 3 discusses the research methodology. Section 4 presents the implementation of the text mining approach. Section 5 presents an evaluation of the approach using several public datasets. Section 6 presents the conclusion.

2 Related Work

In this section, the literature is reviewed around several areas. Current practices around text mining are looked at. This is followed by a description of graph approaches to clustering and classification. Next, there is a discussion of Big Data technologies and their application in analytic problems. Finally, methods of presenting clusters to business users are examined.

2.1 Text mining

In this subsection, some of the literature around text mining techniques and their application is looked at.

Text classification is a frequently required task. Latent Dirichlet Allocation (LDA) (Blei et al.; 2003) is regularly applied. LDA has difficulties working with large corpora as it needs to maintain state across the entire set of documents. This state maintenance is a problem on a Big Data system such as Hadoop using MapReduce (Zhai et al.; 2012).

[1]The Financial World Wants to Open AIs Black Boxes: `https://www.technologyreview.com/s/604122/the-financial-world-wants-to-open-ais-black-boxes/`

Many techniques make use of the document-term matrix (Salton; 1962) using the TF/IDF score (Robertson; 2004). The document-term matrix is, by its nature, very sparse with up to 99% of the entries being zero (Dhillon; 2001). As the size of the corpus increases, the number of document and terms also increases, and the curse-of-dimensionality (Huang et al.; 2005) becomes a consideration.

Naive Bayes, k-Nearest Neighbour (k-NN) and Support Vector Machines (SVM) are popular methods for text classification. There have been recent efforts to add additional complexity in the search for higher accuracy. Greene and Cunningham (2006) discuss the issue of diagonal dominance in SVM with sparse and high-dimensional data such as with text. They examine several options, including a kernel k-means, but that adds additional complexity. Another approach is a hybrid of k-NN and SVM demonstrated by Wan et al. (2012) whereby a k-NN model is trained using the support vectors of the documents instead of the term vectors. This is an excellent reduction approach when dealing with large volumes of text, but results cannot be explained in detail to a stakeholder.

More advanced approaches, such as recurrent convolutional neural networks (Lai et al.; 2015), are computationally expensive and can also be difficult to explain to stakeholders, such as regulators in banking or insurance, who may need to see the workings of the model, which are hidden in neural networks. Huang et al. (2005) investigate the use of methods such as multi-dimensional scaling (MDS) and self-organising maps (SOM), but these are also computationally expensive and can lead to information loss.

In this subsection, it has been seen that LDA is an often used classification approach, but there are difficulties when scaling to Big Data platforms. The issue of scale can also be a problem for the document-term matrix approach as it faces the curse-of-dimensionality. Methods such as using clusters of support vectors, multi-dimensional scaling or self-organising maps are useful but can lead to information loss or are computationally expensive.

In the next subsection, graph-based approaches, which can be useful for the task of text mining, are examined.

2.2 Graph approaches

The previous subsection has shown that when dealing with a large text corpus then there are potential issues with the scaling of conventional methods. The width of the document-term matrix risks the curse-of-dimensionality. In this subsection, graph approaches that look to solve the problems with scale are described.

In Colace et al. (2014), bigrams (pairs of co-occurring words) in a graph are used instead of the traditional bag-of-words, to address the curse-of-dimensionality issue. LDA is applied to the graph to extract the key bigrams. The approach may not scale though as, although LDA can be implemented on a Big Data platform (Nagwani; 2015), there are issues with it because of having to store the shared state (Zhai et al.; 2012).

Bleik et al. (2013) identify that there can be different words used between various documents, so they map the words to a vocabulary that is controlled for the subject matter. They implement a semantic hierarchy to add information to the resultant graph. Once built, two kernel functions were written, one based on edge similarity and one on the cosine distance of edge weights. These are applied using both an SVM and a k-NN algorithm. Their models compare favourably to a document-term matrix-based model. It is a method of some merit, but it requires a vocabulary that must be derived from subject matter experts and may not generalise well.

Wu et al. (2015) demonstrate the extensible nature of graphs when extracting subgraphs of documents. The documents are then treated as bag-of-graphs, and a multi-graph classifier is applied. The results are good but, again, scaling may be an issue.

Because of the sparsity of the document-term matrix, (Dhillon; 2001) suggests a bipartite graph. A bipartite graph contains a set of nodes D, a set of nodes T, and a set of edges E. Any node in D can have edge connections to multiple nodes in T but no connections to any other node in D. Similarly, nodes in T will only have edge connections to D. This type of graph is ideal for modelling the relationship between documents and terms. Dhillon proposes that by co-clustering the documents and terms, then separations can be discovered using a spectral partitioning approach based on the weights assigned to the edges between documents and terms. Dhillon suggests either term frequencies or other term-weighting calculations can be candidates to be used as weights. The approach is for clustering only and can be computationally expensive as Spectral separation is an np-complete calculation.

A bipartite graph is used by (Rossi et al.; 2014) to generate a text classifier. Weights are assigned to each term node as to how it relates to the class of the document, and some good results are achieved versus other techniques such as k-NN and SVM. It is recognised that spurious terms may affect the model. It is unclear how the algorithm will scale as it keeps a running score of the class weights. However, it does show that a classifier can be generated using a bipartite graph.

Collocation of words is important in linguistics. For example, (Mackin; 1978) discusses the issues around teaching foreign language students to learn the correct usage of collocation. Documents are also given context by the words they contain and the intuition in Dhillon (2001), and one of the bases of this current work, is that clustering documents will induce the clustering of terms, and clustering terms will induce the clustering of documents.

Collocation is not considered by the bag-of-words, TF/IDF, approach. An alternative to TF/IDF is proposed by Rousseau and Vazirgiannis (2013). Rather than only counting the term frequencies in a document, a graph-of-words is created which also maintains information about ordering and collocation of terms in the document. This technique produces a term weight (TW) for each term in the document. A TW/IDF score is calculated, using the TW in the usual calculation instead of TF, resulting in good results in testing. In later work, Rousseau and Vazirgiannis (2015) show that the TW technique can be applied along with the concept of k-core to extract keywords from a single document. This method is potentially useful as it is a corpus-independent method of extracting the key terms from a document.

In this subsection, several approaches to using graph techniques when mining text documents have been described. Good results are seen when compared against the document-term matrix model. Although, scalability may still be an issue. The keyword extraction method of Rousseau and Vazirgiannis (2015) is interesting from that point of view because it is corpus independent and so could be easily parallelized on a clustered platform.

The next subsection will focus on Big Data technologies and consider how they can be applied for scaling the task of categorising text.

2.3 Big Data technologies

In the previous subsection approaches to text mining using graphs were presented. Although there is little to be found in the literature on scaling such methods on Big Data platforms, general Big Data technologies, and how they have been applied to other analytic problems on large data, are discussed in this subsection.

Apache Spark was introduced by Zaharia et al. (2012). It performs better than MapReduce because data can be kept in memory and shared across several jobs. Spark is designed to share its in-memory datasets across a cluster of computers in a resilient, fault-tolerant way. White (2012) compares Spark to Hadoop as a distributed technology that does not rely on MapReduce. It has many similar types of processes, but it relies on memory more than disk and hence speeds up operations. White suggests that Spark is a better option than MapReduce when considering the experience of the user. MapReduce is still useful for batch type tasks, and Spark is better when the speed of results is paramount.

Some users who prefer a more declarative approach to writing queries (Armbrust et al.; 2015), which has led to the creation of tools such as Apache Hive, allowing more SQL-like queries to be written. Armbrust et al. have created Spark SQL which gives that ability to write declarative queries against an in-memory DataFrame. Spark DataFrames are further extended by GraphFrames (Dave et al.; 2016), which allows graph-based analytics.

This subsection has examined Big Data techniques that might be considered in text mining. In the final subsection, some methods for visually representing a model are presented.

2.4 Graph and cluster presentation

Big data technologies, such as Hadoop and Apache Spark, are shown in the previous subsection. This subsection examines the presentation of graph and cluster models to a business user.

Gibson et al. (2013) present a survey of two-dimensional graph layout techniques. It is noted that the plotting of a graph helps make better sense of the relationship. The ForceAtlas2 algorithm of Jacomy et al. (2014) is shown to be one of the best-performing methods. This algorithm is best known as being used in the Gephi visualisation tool, as well as the sigma.js JavaScript library. While it can be used to visually cluster data based on node sizes and edge weights, it is not suitable for co-clustering of documents and terms as it clusters all objects at the same time.

The visualisation method of Holten and Van Wijk (2009) seeks to focus on the edges of a graph to show the graph relationships. This gives insight into the relationships between nodes, and often the graph is projected onto two dimensions, such as a map. However, the graph itself is not a bipartite graph, and the projection is based on existing data about the nodes such as latitude and longitude.

Kohonen (1998) provides an excellent method of clustering data and then presenting that result of that clustering to users. Because it is a type of artificial neural network, the method to create the map is quite complicated to create and subsequently difficult to explain to a business user. The method also requires calculations across the entire corpus.

In this subsection, methods have been described that could be used for the rendering of graph and cluster models to a business user.

3 Methodology

The purpose of this research is to develop a method of mining texts that can be applied to a significant number of documents, and the resulting model should be easily explainable to business stakeholders.

The research methodology will follow the Knowledge Discovery in Databases (KDD) approach proposed by Fayyad et al. (1996). The main steps of the KDD process are shown in Figure 1

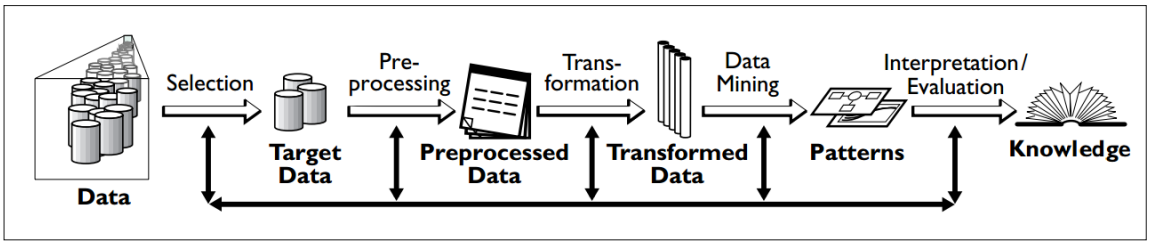

Figure 1: Steps of the KDD process (Fayyad et al.; 1996)

Each of the KDD steps is detailed in the following subsections.

3.1 Selection

Several text corpora that had existing labels or topics identified were selected for this research.

3.1.1 BBC News corpora

In Greene and Cunningham (2006), two sets of documents have been prepared and made available for researchers[2].

The smaller set, curated from the BBC Sport website, consists of 737 articles with five known topics: tennis, rugby, football, cricket and athletics.

The second set, curated from the BBC news website, has 2,225 articles, also having five topics: technology, sport, politics, entertainment and business.

3.1.2 20 Newsgroups

The curated 20 Newsgroups dataset (Mitchell; 1997) was downloaded from the UCI KDD Archive[3]. The corpus consists of approximately 20,000 separate documents representing posts to twenty different Usenet newsgroups.

3.2 Preprocessing

Because of the work of Greene and Cunningham (2006) the BBC datasets have already been cleaned and can be used without further preprocessing. Both sets come in a zip file, which contains a subfolder for each of their five topics.

[2]BBC Datasets: `http://mlg.ucd.ie/datasets/bbc.html`

[3]20 Newsgroups on UCI KDD Archive: `https://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.data.html`

The 20 Newsgroups data set also has a zip file containing the topic subfolders, one for each newsgroup. As an additional processing step, header records, such as the posted newsgroup and the date, which may over-influence the calculated category of a file, are removed.

3.3 Transformation

In the transformation step, the data are transformed into a form that is useful for the data mining process by performing some standard operations.

3.3.1 Data Cleanse

As mentioned above, the BBC datasets were already cleaned. Non-useful parts, such as header records, were deleted from the 20 Newsgroup dataset.

3.3.2 Typical text analytics processes

The typical processes used in text analytics were applied to the corpora (Uysal and Gunal; 2014). This includes steps such as conversion of all the text to lower case; stemming, where common stems of words are swapped into the document; and stop word removal, where common words, such as articles and other function words, are removed.

3.3.3 Keyword Extraction

The keywords were extracted using the techniques of Rousseau and Vazirgiannis (2013). The input of this is a text document, and the output is a list of the keywords with a normalised weighting between 0 and 1, depending on their importance in that document. The method is described in Algorithm 1. Because this is not dependent on any other document, it can be run across multiple nodes on a Spark cluster.

Algorithm 1 Extract the keywords from a document using graph-of-words

Input : Text document
Output: Document name, Terms and Term Weights

Clean document (lower case, stem, remove stop-words)

Split document into an array of words in sequence

foreach *term in the array* **do**
    Add unique triple term + next term to graph
    Add unique triple term + next-next term to graph
    `/* continue depending on selected window size */`
end

foreach *Node in graph* **do**
    if *Degree or InDegree is greater than cutoff limit* **then**
        add document name, term name and normalised term weight to return
    end
end

A chart showing some terms extracted from a document with their degree score is shown in Figure 2. In this case, the window size was 4, so the cutoff limit is 6. Words above the cut-off limit are retained as keywords.

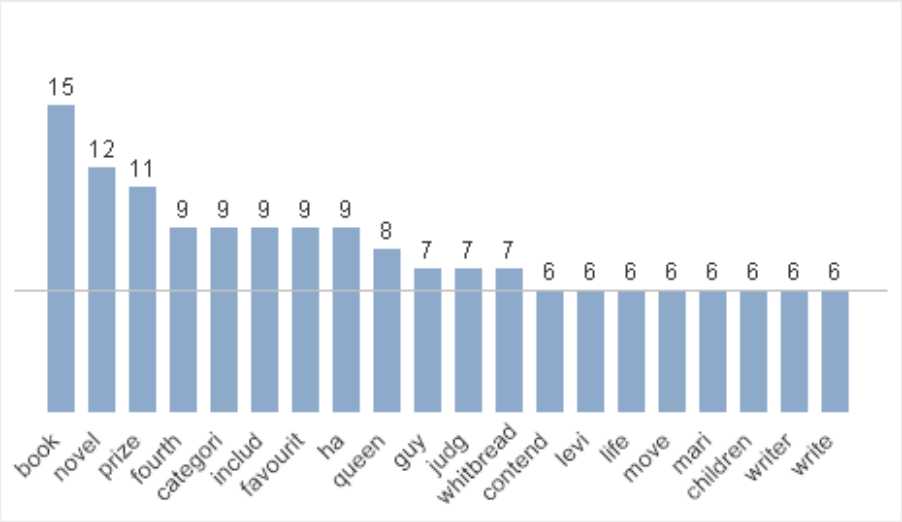

Figure 2: Example of terms extracted using graph-of-words. The cutoff limit is 6, so only terms with a score above the line are retained as keywords.

The Term weight is calculated by dividing each degree score by the maximum degree score in that document. In the example in Figure 2, *book* would get 1.0, *novel* would get 0.8 and *whitbread* would get 0.466

3.3.4 Bipartite graph generation

The results of the keyword extraction across multiple text files were collected into a bipartite graph of documents, key terms and edges with the weighting score applied. An image of documents connected to terms in a graph is shown in Figure 3

3.3.5 Graph data extraction

As an output of the building of the model, the current positions of the nodes and edges are extracted to a data visualisation tool. This visualisation will allow users to explore the model.

3.4 Data Mining

There are two kinds of data mining afforded by processing the bipartite graph - an unsupervised clustering model and a supervised classifier model. Both methods rely on assigning an (x,y) position to each node in the graph and then repositioning the nodes based on the weighted centroid of its connected nodes.

For any term or document, the centroid of its connected nodes corresponds to their centre-of-gravity. The weighted centroid is influenced by the edge weights of the connections, as shown in Figure 4.

The calculation for the x of the centroid is given by the sum of each x times the edge weight (w) for each of the connected nodes, divided by the sum of the edge weights:

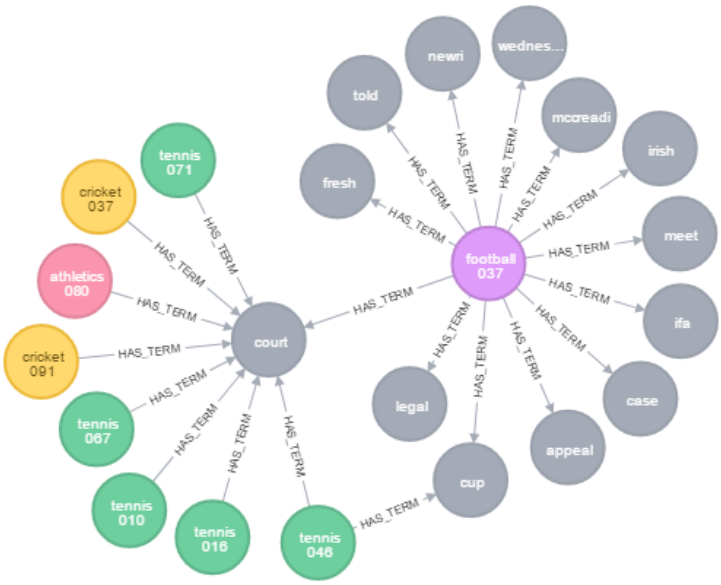

Figure 3: Example of documents connected to terms in a bipartite graph. Document nodes, such as *football 037* or *tennis 046*, only ever connect to term nodes, such as *count* or *cup*, never other document nodes.

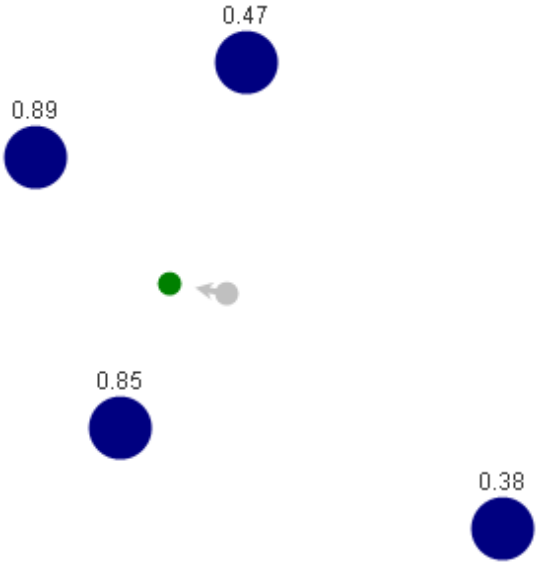

Figure 4: Difference between centroid (grey) and weighted centroid (red). The centroid is at the centre-of-gravity of the four nodes. The weighted-centroid is closest to the nodes on the left with the greater weights.

$$x_c = \sum_{i=1}^{n} x_i w_i / \sum_{i=1}^{n} w_i \quad (1)$$

The calculation for the y of the centroid is given by a similar calculation:

$$y_c = \sum_{i=1}^{n} y_i w_i / \sum_{i=1}^{n} w_i \quad (2)$$

The steps to generate the clustering model are shown in Algorithm 2.

Algorithm 2 Generate cluster model

Input : Graph of nodes and edges extracted using graph-of-words
Output: Cluster model

Randomly assign initial (x,y) positions to document nodes

`/* Co-clustering process */`
while *entropy change is greater than stop value* **do**
    Assign term nodes to weighted centroid of the term's documents
    Assign document nodes to weighted-centroid of the document's terms
    Calculate entropy change
end

Export cluster model

The process to build the classifier is similar. The labels for the documents are already known, so the starting position for the document nodes can be specified, based on those labels. Only one iteration of the co-clustering algorithm is necessary. The steps to build the classification model are shown in Algorithm 3.

Algorithm 3 Generate classification model

Input : Graph of nodes and edges extracted using graph-of-words
Output: Classification model

Assign each document a two-dimensional spatial position (x,y) by positioning the document nodes on the circumference of a circle, grouped into their topics. The arc distance between each topic is relative to the size of each group.

Assign term nodes to weighted-centroid of the term's documents

Assign document nodes to weighted-centroid of the document's terms

Build kd-tree (Bentley; 1975) is built using the document node data.

Export classification model (term nodes + kd-tree)

Once the model has been built, then it can be used to predict against a new document with an unknown topic. The predictor can return either an absolute category or a fuzzy percentage. The steps to perform the prediction is shown in Algorithm 4.

Algorithm 4 predict the category of a new document

Input : Document key terms extracted using graph-of-words
Output: Absolute or fuzzy topic

Each keyword is matched to the term nodes.

The weighted-centroid of the matched nodes is calculated.

With the kd-tree, run a k-nearest neighbour algorithm to calculate topics that are nearest that centroid.

if *Absolute result is required* **then**
| return topic with the highest k-NN score
else
| return all topics with percentage match
end

3.5 Interpretation / Evaluation

So as to evaluate the classifier, the datasets are split into training and test sets on a typical ratio of 7:3. Evaluation of the predictor is via the standard *F1* score on the results of the application of the trained classifier to the test set.

4 Implementation

In implementing this research project, three main steps were applied. Firstly, keywords were extracted using the graph-of-words approach. Next, clustering was attempted using the random positions and the repositioning to weighted centroids. Finally, a classifier was built using known start positions and repositioning using weighted centroids. These steps are detailed in this section.

4.1 Keyword extraction using graph-of-words

The graph-of-words Term Weight extraction method of Rousseau and Vazirgiannis (2013) was implemented. This method extracts connected terms from a document based on a sliding window of fixed length, either 3 or 4 terms wide. When the graph is constructed, the Term Weight is evaluated by counting either the degree or the indegree. Logically, every term in the document will have a count of either the window size minus one (indegree) or twice that value (degree).

In a later work, Rousseau and Vazirgiannis (2015) discuss extracting keywords using k-core. The k-core algorithm (Scott; 2017) removes vertices with less than degree k. For this work, k is based on window size, and any term that scores above this cutoff limit will be considered as a keyword.

In the next subsection, the clustering of the terms and documents using a centroid based calculation is described.

4.2 Clustering with bipartite graphs and weighted centroids

The previous subsection discussed extraction of keywords. This subsection details the generation of a bipartite graph and co-clustering documents and terms.

For this bipartite graph, there is one kind of node for documents, another kind for terms. Edges connect terms to documents, weighted with the Term Weight that was previously extracted.

The novel method of this research is to give each document and term node a location in two-dimensional space (x,y).

For the unsupervised clustering task, the starting locations of each document is assigned using a random value. Several iterations are carried out in which the location of the terms are first set to the weighted centroids of their associated documents; then the location of all the documents are set to the weighted centroids of their associated terms.

As the process is iterated, the terms and documents will begin to co-cluster. If too many iterations are executed, then the nodes will tend to converge. Therefore, a stop condition is needed. This condition is achieved by measuring the entropy between two iterations using a Kullback-Leibler divergence (Kullback and Leibler; 1951) and comparing to a parameter.

An image of the output of the process after eight iterations is shown in Figure 5. In this image, the terms have been coloured light grey, and the documents have been coloured using the known topic labels.

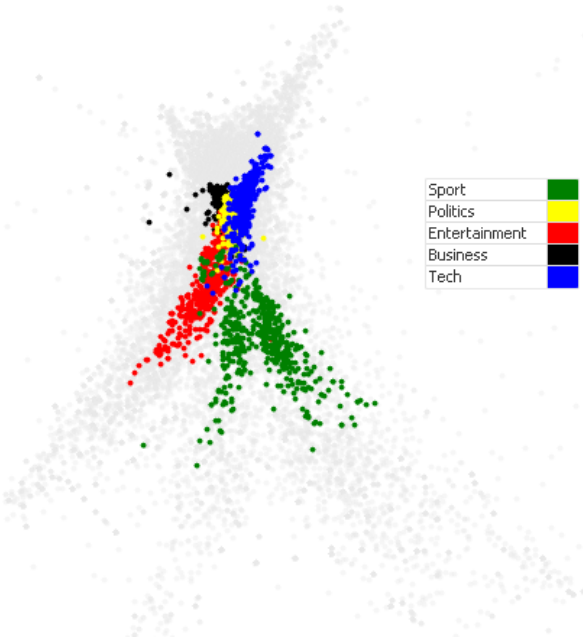

Figure 5: After several iterations of the process, clusters emerge. The terms have been coloured light grey while the documents have been coloured post hoc using document topics.

The image shows that clusters of documents have emerged, but these would not necessarily be easily extracted using a conventional clustering algorithm. However, as the data has been mapped to two dimensions, the results can be surfaced using a visualisation tool, such as QlikView[4], which may afford interactive discovery and assignment of clusters.

In the next subsection, it will be shown how using a similar positioning technique can be used to create a text classifier.

[4]QlikView: `http://www.qlik.com/us/products/qlikview`

4.3 Implementing a text classifier

In the previous subsection, the technique of co-clustering the nodes in a bipartite graph is demonstrated. In this subsection, applying the method to text classification is discussed.

For classification, initial locations of the document nodes can be established based known topics, instead of randomly. Rather than requiring multiple iterations, only one relocation of the terms relative to connected documents, followed by one relocation of the documents relative to connected terms, is necessary. An example a classification model built on the BBC Sport dataset is shown in Figure 6.

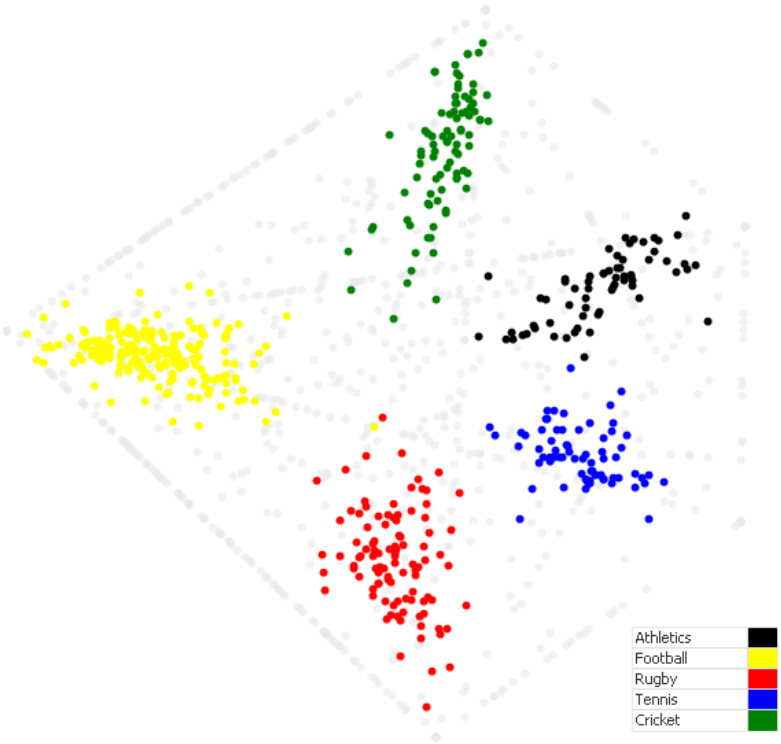

Figure 6: Image of a model built on the BBC Sport dataset.

With the documents having been assigned their final locations, a kd-tree Bentley (1975) is built using the two-dimensional location values of the nodes.

When classifying new documents, the key terms are established as before, and then locations mapped from the terms in the graph. A weighted centroid calculation is performed, and the nearest-neighbours are calculated from the kd-tree. The topics of the nearest-neighbours are used to establish the classification.

5 Evaluation

The method of evaluation of the classifier performance is the standard *F1* score, based on the *precision* score and the *recall* score. Several tests were executed on different corpora.

Precision (p) is a measure of the ratio of the correct predictions of a class versus the number of predictions of that class:

$$p = \frac{number\ of\ class\ predictions\ correct}{number\ of\ class\ predictions} \tag{3}$$

Table 1: F1 scores calculated using three methods

Dataset	k-NN	SVM	Graph
BBC Sport	0.9778	0.9526	0.9598
BBC News	0.9562	0.9720	0.9206

Table 2: F1 scores when using all twenty labels

Dataset	k-NN	SVM	Graph
20 Newsgroups	0.2065	0.7775	0.3280

Recall (r) is the measure of the ratio of the correct predictions of a class versus the number actual correct class values:

$$r = \frac{number\ of\ correct\ predictions}{actual\ number\ in\ class} \tag{4}$$

The *F1* score is the harmonic mean between the *precision* and *recall* and is given by the formula:

$$F1_{(p,r)} = \frac{2pr}{p+r} \tag{5}$$

When dealing with multiple classes, as in the case of a text categorisation problem, a weighted average is calculated across the categories.

5.1 Basic text classifier

A classifier model was created using the BBC News and BBC Sport datasets. To compare performance, RapidMiner was used to create classifier models, on the same datasets, using both k-NN and SVM (linear kernel) algorithms.

The F1 scores achieved with the three different models are displayed in Table 1.

For the two different corpora, the results are good across the three methods. The high F1 score indicates that the graph method is quite accurate on these datasets.

5.2 Identification of topic groups

The twenty-newsgroup set is described in section 3.1.2. Attempting to classify across this corpus using each label as a classifier proves to be unsuccessful, as shown in Table 2, although SVM performs adequately.

When considering this dataset, a low score is not unexpected. Many of the newsgroups are on similar topics and may have similar keywords. Surfacing the results allows a user to interact visually and discover groupings.

The results of visually inspecting the model are demonstrated in Figure 7. The first panel displays the whole model with mixed-up topics. In the second panel it is seen that for the newsgroup *soc.religion.christian*, it does not overlap with *talk.religion.misc* and *alt.atheism* as expected. In the third panel, *soc.religion.christian* is seen to actually overlap with the *talk.politics* group. Groups *rec.sport.baseball* and *rec.sport.hockey* show little overlap.

Based on exploring the surfaced model, new topic group sets emerge, as shown in Table 3.

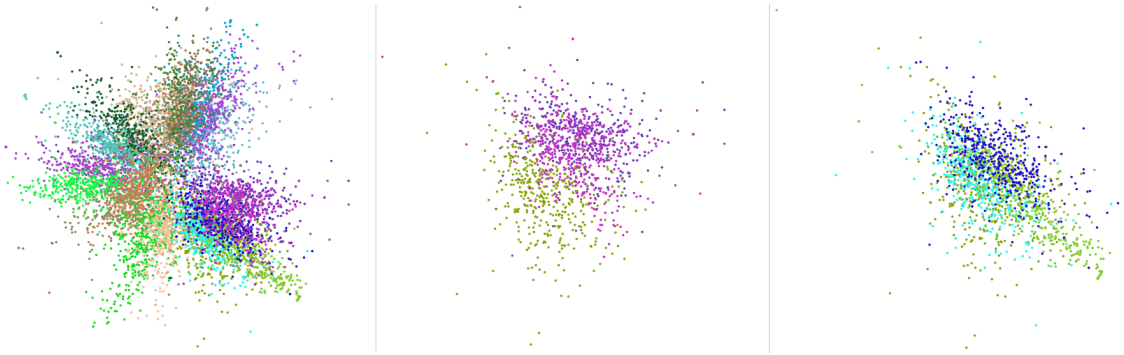

Figure 7: Surfacing the model to identify topic groups.

Table 3: The 20 Newsgroups grouped into topics

Science	**Computer**	**Politics**
sci.crypt	comp.graphics	talk.politics.guns
sci.electronics	comp.sys.ibm.pc.hardware	talk.politics.mideast
sci.med	comp.os.ms-windows.misc	talk.politics.misc
sci.space	comp.sys.mac.hardware	soc.religion.christian
	comp.windows.x	
Misc	**Motors**	**Hockey**
misc.forsale	rec.motorcycles	rec.sport.hockey
	rec.autos	
Religion		**Baseball**
alt.atheism		rec.sport.baseball
talk.religion.misc		

Table 4: F1 scores with the new topic groups

Dataset	k-NN	SVM	Graph
8 Topic Groups	0.3712	0.8477	0.6381

With the new groups, the F1 scores improve in both the SVM and the graph method, as shown in Table 4. This better score is achieved because of the improved separability of the topics. By using surfacing the model and exploring the topics, these discoveries can be made, and the results improved. This ability is a key benefit of the method.

5.3 Model Scalability

While the accuracy of the model is important, the execution speed of the building and application of the model is also a consideration, as a typical workflow will require multiple iterations of training and testing.

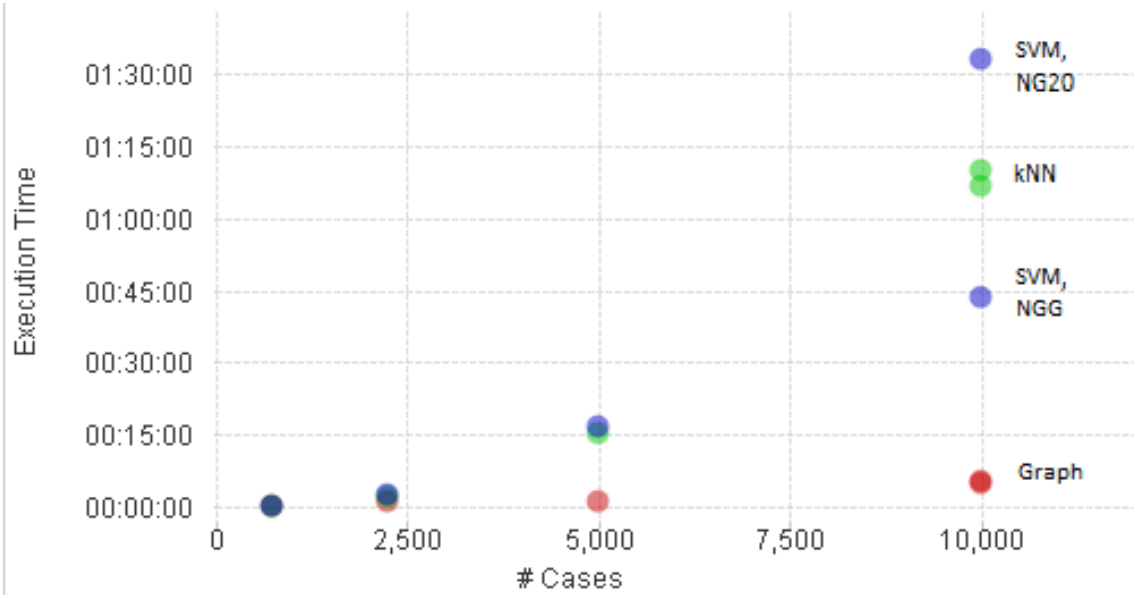

Figure 8: Execution time for train/test cycles for different corpus sizes. The graph method scales linearly, while the k-NN and SVM scale exponentially. SVM scaling is also dependent on the number of classes.

As shown in Figure 8, the graph method's execution time is linearly related to the number of documents in the corpus, and it performs quite quickly with the largest dataset. The k-NN method's execution time increases exponentially based on the number of documents. The SVM method's time is also exponential but is based on both the number of documents and the number of classes. The SVM method has a dependency on the number of classes because the One-Versus-All classification method is used.

6 Conclusion and Future Work

6.1 Conclusion

This paper has presented the research around developing a text classifier that can be used with any body of documents and can scale on a distributed computing platform. The result is the novel method of using graphs to both extract keywords and model the document and term relationship, and then project that bipartite graph onto two dimensions. The extraction of the keywords can be scaled using Apache Spark, and the classification model can be surfaced visually to stakeholders.

The unsupervised clustering method confirms the observation of (Dhillon; 2001) that term clustering induces clustering of documents and document clustering induces clustering of terms. While the method alone does not always produce useful clusters, it did suggest the supervised classification method.

The classifier model produced good results on some of the corpora used. Where the results were lower, surfacing the model allows a user to discover overlapping topics that could be merged.

The classifier build process is linearly dependent on the number of documents classified. Because the keyword extraction process is corpus independent, the extraction can be scaled on a clustered system such as Apache Spark.

This method is a valuable addition to the body of knowledge as it creates a visually interactive text classifier that is explainable to stakeholders, which is important in many industries, such as banking or insurance, where it is required to explain the model being used. This ability becomes additionally important with the continual arrival of new regulations, such as GDPR[5].

The approach also demonstrates a method of presenting bipartite data in a way which increases the understanding of the relationships and co-clustering of those data.

6.2 Future Work

Nassirtoussi et al. (2015) introduce an interesting semantic abstraction technique using hypernyms of words. This approach would be interesting to apply along with the graph-of-words keyword extraction method. By replacing the keywords with their hypernyms, it would be interesting to see how the model emerges.

The model built is presented on a two-dimensional plane with a circle used to calculate the starting position. Future research could look at using a three-dimensional model and different starting shapes to see if model performance changes. The model could be extended to many more dimensions, but anything more than three would lose the ability for the user to visualise effectively.

An additional clustering method could be introduced, perhaps using a Markov chain relating documents to each other by the probability of their two-step connection. The resultant clusters can then be visualised using this method in this paper.

Documents and terms are not the only kinds of data that can be modelled in a bipartite graph. An interesting example that might be explored further is the relationship between customers and products that they buy. The result may deliver a useful product recommendation system.

[5]GDPR: `http://www.eugdpr.org/`

Acknowledgements

Many thanks to my supervisor, Eleni Rozaki, for her constant support and guidance during this research project.

Author's Publications

The main concept of this thesis has been accepted for presentation and publication at the Fifth International Conference on Advances in Computing, Communication and Information Technology - CCIT 2017, 2-3 September 2017.

Title: Using bipartite graphs projected onto two dimensions for text classification

References

Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., Meng, X., Kaftan, T., Franklin, M. J., Ghodsi, A. and Zaharia, M. (2015). Spark sql: Relational data processing in spark, *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, ACM, New York, NY, USA, pp. 1383–1394.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching, *Communications of the ACM* **18**(9): 509–517.

Blei, D. M., Ng, A. Y. and Jordan, M. I. (2003). Latent dirichlet allocation, *Journal of machine Learning research* **3**(Jan): 993–1022.

Bleik, S., Mishra, M., Huan, J. and Song, M. (2013). Text categorization of biomedical data sets using graph kernels and a controlled vocabulary, *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **10**(5): 1211–1217.

Colace, F., Santo, M. D., Greco, L. and Napoletano, P. (2014). Text classification using a few labeled examples, *Computers in Human Behavior* **30**: 689 – 697.

Dave, A., Jindal, A., Li, L. E., Xin, R., Gonzalez, J. and Zaharia, M. (2016). Graphframes: an integrated api for mixing graph and relational queries, *Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems*, ACM, p. 2.

Dhillon, I. S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning, *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 269–274.

Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. (1996). The kdd process for extracting useful knowledge from volumes of data, *Communications of the ACM* **39**(11): 27–34.

Gibson, H., Faith, J. and Vickers, P. (2013). A survey of two-dimensional graph layout techniques for information visualisation, *Information Visualization* **12**(3-4): 324–357. Copyright - SAGE Publications Jul 2013; Document feature - Tables; ; Last updated - 2013-08-05.

Greene, D. and Cunningham, P. (2006). Practical solutions to the problem of diagonal dominance in kernel document clustering, *Proc. 23rd International Conference on Machine learning (ICML'06)*, ACM Press, pp. 377–384.

Hinton, G. and Roweis, S. (2002). Stochastic neighbor embedding, *NIPS*, Vol. 15, pp. 833–840.

Holten, D. and Van Wijk, J. J. (2009). Force-directed edge bundling for graph visualization, *Computer graphics forum*, Vol. 28, Wiley Online Library, pp. 983–990.

Huang, S., Ward, M. O. and Rundensteiner, E. A. (2005). Exploration of dimensionality reduction for text visualization, *Coordinated and Multiple Views in Exploratory Visualization, 2005.(CMV 2005). Proceedings. Third International Conference on*, IEEE, pp. 63–74.

Jacomy, M., Venturini, T., Heymann, S. and Bastian, M. (2014). Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software, *PLOS ONE* **9**(6): 1–12.

Kohonen, T. (1998). The self-organizing map, *Neurocomputing* **21**(1): 1–6.

Kullback, S. and Leibler, R. (1951). The george washington university and-washington, dc, *The Annals of Mathematical Statistics* **22**(1): 79–86.

Lai, S., Xu, L., Liu, K. and Zhao, J. (2015). Recurrent convolutional neural networks for text classification., *AAAI*, Vol. 333, pp. 2267–2273.

Lin, J. and Dyer, C. (2010). Data-intensive text processing with mapreduce, *Synthesis Lectures on Human Language Technologies* **3**(1): 1–177.

Mackin, R. (1978). On collocations: Words shall be known by the company they keep, *Honour of AS Hornby* pp. 149–165.

Mitchell, T. M. (1997). Machine learning. 1997, *Burr Ridge, IL: McGraw Hill* **45**(37): 870–877.

Nagwani, N. (2015). Summarizing large text collection using topic modeling and clustering based on mapreduce framework, *Journal of Big Data* **2**(1): 1.

Nassirtoussi, A. K., Aghabozorgi, S., Wah, T. Y. and Ngo, D. C. L. (2015). Text mining of news-headlines for forex market prediction: A multi-layer dimension reduction algorithm with semantics and sentiment, *Expert Systems with Applications* **42**(1): 306–324.

Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for idf, *Journal of documentation* **60**(5): 503–520.

Rossi, R. G., de Andrade Lopes, A., de Paulo Faleiros, T. and Rezende, S. O. (2014). Inductive model generation for text classification using a bipartite heterogeneous network, *Journal of Computer Science and Technology* **29**(3): 361–375.

Rousseau, F. and Vazirgiannis, M. (2013). Graph-of-word and tw-idf: new approach to ad hoc ir, *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, ACM, pp. 59–68.

Rousseau, F. and Vazirgiannis, M. (2015). Main core retention on graph-of-words for single-document keyword extraction, *European Conference on Information Retrieval*, Springer, pp. 382–393.

Salton, G. (1962). Some experiments in the generation of word and document associations, *Proceedings of the December 4-6, 1962, fall joint computer conference*, ACM, pp. 234–250.

Scott, J. (2017). *Social network analysis*, 4th edn, Sage, London.

Uysal, A. K. and Gunal, S. (2014). The impact of preprocessing on text classification, *Information Processing Management* **50**(1): 104 – 112.

Wan, C. H., Lee, L. H., Rajkumar, R. and Isa, D. (2012). A hybrid text classification approach with low dependency on parameter by integrating k-nearest neighbor and support vector machine, *Expert Systems with Applications* **39**(15): 11880 – 11888.

White, T. (2012). *Hadoop: The definitive guide*, O'Reilly Media, Inc., Sebastopol, CA.

Wu, J., Pan, S., Zhu, X. and Cai, Z. (2015). Boosting for multi-graph classification, *IEEE Transactions on Cybernetics* **45**(3): 416–429.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S. and Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing, *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, USENIX Association, pp. 2–2.

Zhai, K., Boyd-Graber, J., Asadi, N. and Alkhouja, M. (2012). Mr. LDA: A flexible large scale topic modeling package using variational inference in mapreduce, *ACM International Conference on World Wide Web*.