

Copyright © 2005 University of Stellenbosch
All rights reserved.



Declaration

I, the undersigned, hereby declare that the work contained in this assignment is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature:

I.C. Mogotsi

Date:



Abstract

WHAT DID THEY COVER? A Cluster Analysis of News Stories Published in the *Botswana Daily News*, January - December 2004

I.C. Mogotsi

Department of Information Science

University of Stellenbosch

Assignment: MPhil(IKM)

April 2006

In this study, a cluster analysis of news stories published in the *Botswana Daily News* during the period January - December 2004 was undertaken. The study was exploratory in nature and sought to find out what topics were predominant during the study period. The approach we adopted can be divided into three phases, namely data collection, document pre-processing, and cluster analysis. The data used in the study was downloaded from the *Botswana Daily News* website using a simple program developed specifically for that purpose. Document pre-processing was concerned with transforming the raw documents into a format that could be directly operated upon by the various clustering algorithms. The documents themselves were represented using the vector space model, with the *tf.idf* term weighting scheme. We experimented with three clustering approaches, namely, direct k-way clustering, k-way clustering through repeated bisections, and agglomerative clustering. Agglomerative clustering performed poorly, and we thus discarded its results. Direct k-way clustering and k-way clustering through repeated bisections produced similar results, though the former performed better in terms of external isolation and internal cohesion of the clusters produced. Consequently, we only retained the results from direct k-way clustering, and subsequently performed a quarterly analysis of our corpus using only the direct k-way clustering algorithm. Analysis of the complete corpus identified a number of topics that were prevalent over the study period. Interestingly, a quarterly analysis of the corpus revealed other topics whose prevalence appears to have been limited to certain parts of the year.

Opsomming

WAT HET HULLE GEPUBLISEER?

'n Trosanalise van nuusberigte gepubliseer in die *Botswana Daily News*, Januarie-Desember 2004

I.C. Mogotsi

Departement Inligtingwetenskap

Universiteit van Stellenbosch

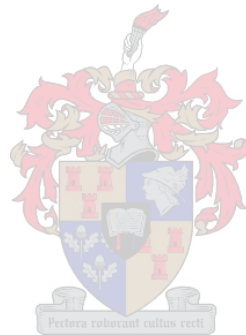
Werkstuk: MPhil(IKM)

April 2006

In hierdie studie is 'n trosanalise van nuusberigte onderneem wat in die *Botswana Daily News* gedurende die tydperk Januarie-Desember 2004 gepubliseer is. Die studie was ondersoekend van aard en het gepoog om vas te stel watter onderwerpe oorheersend tydens die oorsigtydperk was. Die benadering wat gevolg is, kan in drie fases ingedeel word, naamlik data-insameling, dokumentvoorverwerking en trosanalise. Die data wat in die studie aangewend is, is van die *Botswana Daily News* se webwerf afgelaai met behulp van 'n eenvoudige program wat spesifiek vir hierdie doel ontwerp is. Dokumentvoorverwerking verwys na die omvorming van die rou dokumentasie tot 'n formaat wat direk gemanipuleer kon word deur die verskeie trosvormingsalgoritmes. Die dokumente self is verteenwoordig met gebruik van die vektorruimtemodel met die gewilde *tf.idf* termbeswaringskema. Daar is met drie trosvormingbenaderings geëksperimenteer, naamlik direkte k-rigting trosvormings, k-rigting trosvormings deur herhaalde halvering en agglomerasiewe trosvormings. Laasgenoemde trosvormings het swak vertoon en hierdie uitslae is dus verwerp. Direkte k-rigting trosvormings en k-rigting trosvormings deur herhaalde halveerders het soortgelyke resultate opgelewer, alhoewel laasgenoemde beter vertoon het in terme van die eksterne isolasie en die interne kohesie van die trosse wat verkry is. Slegs die resultate wat uit die direkte k-rigting trosvormings verkry is, is behou en 'n kwartaallikse analise van die korpus is daarna onderneem deur slegs die direkte k-rigting trosvormingsalgoritme te gebruik. 'n Analise van die totale korpus het 'n aantal onderwerpe aangetoon wat algemeen oor die studietydperk voorgekom het. Interessant genoeg het 'n kwartaallikse analise van die korpus weer ander onderwerpe aangetoon wat oënskynlik tot sekere dele van die jaar beperk was.

Dedication

To my father, OK, and late mother, LM.

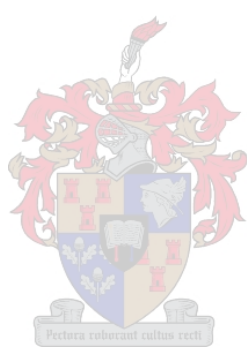


Acknowledgements

My enrolment on the MPhil (IKM) programme – and, by extension, this study – was made possible by generous funding availed by the University of Botswana. The Faculty of Humanities graciously provided me with amenities (office space, computer, telephone) that greatly facilitated this research. During the course of my studies at the University of Stellenbosch, I benefited immensely from interaction with the various course tutors, as well as fellow students on the programme; in particular, I would like to mention Dr Ben Fouchè (formerly Professor of Information Science at the University of Stellenbosch) and Mr Mwala Sheba, a colleague and classmate, for their words of encouragement over the years. Many thanks to Professor Johann Bacher who allowed me to use, and cite, his lecture notes entitled ‘Cluster Analysis’. I would also like to record my gratitude to my supervisor, Dr Martin van der Walt, for the expert guidance he provided during the course of this study. Martin painstakingly read and corrected various drafts of this report; sadly, I have to take full responsibility for any lingering errors, whether they be of omission, or commission. The report itself was typeset with L^AT_EX, using the University of Stellenbosch thesis package developed by Danie Els. On the administrative side, the assistance provided by Ms Netta Strauss, Ms Alma van der Spuy, and Ms Winifred Fourie was invaluable in making my transactions with the University smooth. Finally, I would like to thank my wife Annah, and my children Ame, Neo, and Thata, for their support over the decades.

Contents

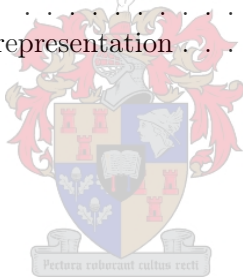
Declaration	ii
Abstract	iii
Opsomming	iv
Dedication	v
Acknowledgements	vi
Contents	vii
List of Figures	ix
List of Tables	x
1 Introduction	1
1.1 Background	1
1.2 Research Aims And Objectives	2
1.3 Motivation	2
1.4 Cluster Analysis	3
1.5 Methodology	4
1.6 Report Structure	4
2 Foundations	6
2.1 Introduction	6
2.2 Document Modelling	7
2.3 Index Term Selection	13
2.4 Cluster Analysis	16
2.5 Hierarchical Clustering	18
2.6 Partitional Clustering	20
2.7 Evaluating and Interpreting Cluster Analysis	22
2.8 Summary	23



3	Methods	24
3.1	Introduction	24
3.2	Document Preprocessing	26
3.3	Index Term Selection	30
3.4	Creating Document Vectors	31
3.5	Criterion Functions in CLUTO	33
3.6	Document Clustering	35
3.7	Summary	36
4	Data Analysis and Interpretation	37
4.1	Introduction	37
4.2	Data Description	37
4.3	Understanding the Output from gCLUTO	39
4.4	Side Effects of Stemming	42
4.5	Direct K-Way Clustering	42
4.6	K-Way Clustering Through Repeated Bisections	48
4.7	Agglomerative Clustering	51
4.8	Comparing Direct and Bisecting K-way Clustering	54
4.9	Quarterly Analysis	56
4.10	Summary	59
5	Summary, Conclusions and Further Work	61
5.1	Introduction	61
5.2	Summary of Work Undertaken	61
5.3	Summary of Findings	63
5.4	Further Work	67
	References	70
	A Source Code Listings	73
	B SMART Stop List	112
	C Botswana Daily News Non-publishing Days	114
	D Direct K-Way Clustering 10-Cluster Solution	115
	E Quarterly Analysis CLUTO output	117

List of Figures

2.1	The Cosine correlation coefficient	12
2.2	Internal cohesion and external isolation	16
2.3	Dissection of a set of objects into sectors	17
2.4	Dendogram illustrating hierarchical clustering	19
2.5	Measuring distance between clusters	19
3.1	Main stages of the KDT process	25
3.2	Document Preprocessing Activities	26
3.3	A sample sparse matrix	31
3.4	CLUTO sparse matrix representation	32



List of Tables

2.1	A simple term-by-document matrix representation of two documents	8
2.2	The k-means Algorithm	21
2.3	The Bisecting k-means Algorithm	21
3.1	Document preprocessing utilities developed for use in the current study.	29
3.2	The mathematical definition of CLUTO's clustering criterion functions	34
3.3	gCLUTO clustering options for direct k-way clustering	36
4.1	Summary corpus statistics	38
4.2	Sample gCLUTO output, showing ISim and ESim, and their deviations	40
4.3	Sample gCLUTO output, showing descriptive and discriminating features.	41
4.4	Direct k-way 15 cluster solution	45
4.5	Direct k-way 10 cluster taxonomy	46
4.6	Direct k-way 15 cluster taxonomy	46
4.7	Direct k-way 20 cluster taxonomy	47
4.8	15 cluster bisecting k-way clustering solution	50
4.9	15-cluster complete linkage agglomerative clustering solution	53
4.10	A comparison of the bisecting and direct k-way 15 cluster solutions	55
4.11	Agreement of randomly selected stories with their cluster labels	56
4.12	Quarterly news analysis with the direct k-way clustering technique: first quarter	57
4.13	Quarterly news analysis with the direct k-way clustering technique: second quarter	57
4.14	Quarterly news analysis with the direct k-way clustering technique: third quarter	58
4.15	Quarterly news analysis with the direct k-way clustering technique: fourth quarter	58
5.1	Taxonomy of news stories published in the <i>Botswana Daily News</i> , January - December 2004.	65

5.2 Summary of Results From Quarterly Analysis of Corpus 66

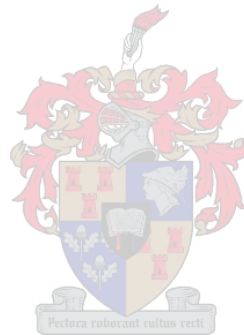
D.1 Direct K-Way Clustering 10-Cluster Solution 116

E.1 Quarterly Analysis Results: Quarter 1 119

E.2 Quarterly Analysis Results: Quarter 2 121

E.3 Quarterly Analysis Results: Quarter 3 123

E.4 15 cluster bisecting k-way clustering solution 125



Chapter 1

Introduction

1.1 Background

In this study, we use techniques from the nascent field of text mining to investigate news articles published in a state-owned daily newspaper over a period of one year. The related field of data mining, which “[lies at] the intersection of statistics, machine learning, data management and databases, pattern recognition, artificial intelligence, and others”, is concerned with “extracting useful information from large data sets” (Hand *et al.*, 2001:xxvi). The growth of interest in data mining in recent years is attributed to a number of factors, among them the drop in the cost of computer processing power and disk storage, and the concomitant explosion in the quantity of data maintained by businesses (Two Crows Corporation, 1999; Dilly, 1995). While data mining was initially applied almost exclusively to the analysis of structured data (as found, for instance, in relational databases), its use for the analysis of textual data is also growing (see, for instance, Berry (2004); within the literature, the title of “data mining” is reserved for the former, with the latter referred to as “text mining”. In essence, text mining combines techniques from natural language processing and information retrieval with the more traditional data mining techniques, and uses them to analyse textual data.

In any case, data (and text) mining uses “computer learning techniques to automatically analyse and extract knowledge from data contained within a database” (Roiger and Geatz, 2003:4). Computer learning can itself be classified as being “supervised” or “unsupervised”. In supervised learning, the output classes are known; the task of the learner is to determine the class to which a new, previously unclassified instance belongs i.e. “the scheme operates under supervision by being provided with the actual outcome for each of the training examples” (Witten and Frank, 2000:38); this is the case in classification, prediction and estimation. In contrast, in unsupervised learning the output class is not known in advance; the task of unsupervised learning, then, is to identify patterns when no specific output values are given (Russel and Norvig, 2003).

This is the case with clustering, which we use in the current study.

In reality, cluster analysis is “a generic name for a variety of mathematical methods, numbering in the hundreds, that can be used to find out which objects are similar” (Romesburg, 1984:2). We discuss cluster analysis in depth in chapter 2; however, taking “news stories” as the objects in our study, we can immediately see the connection between what we seek to achieve in our research, which we discuss next, and the use of cluster analysis.

1.2 Research Aims And Objectives

The aim of this study is to explore the main themes addressed in the news stories published in the *Botswana Daily News*, a daily newspaper owned and published by the Botswana Government, during the period January - December, 2004. Being exploratory in nature, the study eschews the “verification model” in which the user tests his/her own hypothesis (the null hypothesis) against the data, in favour of the “discovery model”, characteristic of data mining, in which “the system automatically discovers important information hidden in the data” (Dilly, 1995). The intention, then, is to learn from data, with the learning being unsupervised. Specifically, the study seeks (i) to use cluster analysis to identify natural groupings in the collection of stories published in the *Botswana Daily News* during the said period, and (ii) to develop meaningful and informative descriptions for the identified clusters. Though positivistic in our ontological outlook, we are mindful of the warning given by Hair *et al.* (1995) that while cluster analysis is usually thought to be ‘structure seeking’, it actually imposes a structure through the selected methodology i.e. different clustering approaches will (potentially) identify different cluster configurations.

1.3 Motivation

News analysis is interesting from a number of perspectives. For businesses seeking to outsmart the competition, competitive intelligence (CI) can be a veritable weapon. CI is “. . . the process by which companies inform themselves about every aspect of their rivals’ activities and performance” (West, 2001:12) through “organised, structured information gathering, analysis and processing . . .” (Cook and Cook, 2000:5). All data gathered for use as part of the CI process is derived from public sources (McGonagle and Vella, 1998), such as the *Botswana Daily News*. Such data can be about the activities of other companies, or other factors within the business environment, such as new laws and government policies. News stories also reflect societal interests at any point in time, although, of course, the decision of what to write about and what to publish lies with reporters, editors and other influential stakeholders. In the case of a publicly funded newspaper, it may be interesting to see what public

funds are being used to report on. But why use cluster analysis? The volume of information published in a newspaper annually is such that manual analysis techniques would not be able to cope; this is particularly true in the case of a daily. In the context of structured data as found in, for instance, relational databases, analysis of large volumes of data is undertaken using data mining techniques. Similarly, knowledge discovery in text corpora uses text mining techniques. This study proposes to use cluster analysis, a machine learning approach with roots in statistics, to undertake the requisite analysis.

1.4 Cluster Analysis

Cluster analysis has been applied in a wide variety of disciplines, including psychiatry, medicine, market research, education, archaeology, classification of stratigraphical pollen spectra, soil profiling, classification of aesthetic judgements on painters, to name but a few (Everitt, 1993; Gordon, 1981). The main attractions of cluster analysis are that (i) it is conceptually easy to understand, (ii) it is quite straightforward to apply, and (iii) it works well with different data types (Berry and Linoff, 1997).

Despite its obvious appeal, cluster analysis does pose some interesting challenges; the following are summarised from Dunham (2003:127) and Berry and Linoff (1997): (i) dealing with outliers can be difficult; (ii) in the case of dynamic data, such as the data considered in the current study (i.e. document databases), cluster membership may change as data is added or removed from the database; (iii) since the labelling of clusters is not known *a priori*, it is not always easy to interpret the results of cluster analysis, though domain expertise does help, and other approaches, such as association rule mining, may also be brought to bear on the identified clusters; (iv) there is no single, correct way to solve a clustering problem; for instance, as Dunham indicates, clustering a group of homes on the basis of geographic distance between the homes will produce a different result from clustering on the basis of the sizes of homes; additionally, in each case changing the number of clusters will significantly change the results of analysis; and (v) deciding what attributes should be used for clustering is not necessarily trivial.

There is a propensity in the extant literature to view cluster analysis solely as exploratory. However, cluster analysis can be either exploratory or confirmatory. According to Bacher (2002), in exploratory cluster analysis “the number of clusters is unknown (i.e. the number of clusters has to be estimated), the characteristics of clusters (such as cluster centres in k-means) are unknown (i.e. clusters have to be interpreted, and finding a substantive interpretation can be difficult), and the fit to data is maximised; in contrast, in confirmatory cluster analysis “the number of clusters is known (i.e. the number of clusters shall not be estimated), the characteristics of clusters are known (even if only partially), the clusters already have a substantive interpretation, and the fit

to data may be poor”.

Motivation for the use of cluster analysis in Information Retrieval can be traced back to Van Rijsbergen (1979)’s Cluster Hypothesis according to which “closely associated documents tend to be relevant to the same request”. The hypothesis suggests that (i) documents in a collection could be clustered *a priori* and incoming queries matched against the clusters as opposed to individual documents, which would have the benefit of speeding the retrieval process as well as finding documents indexed with terms not in the user’s query; (ii) clustering can be used to enhance browsing by grouping together documents in the result of a query (Sahami, 1998). Much of the literature on document clustering, though, seems to be concerned with developing and fine tuning clustering algorithms using standardised test data, as opposed to applying existing algorithms to real data, as we do in this study.

1.5 Methodology

Following Karanikas and Theodoulidis (2002), our methodology consists of three phases, namely, (i) collection of relevant documents, (ii) pre-processing documents and (iii) actual text mining operations i.e. documents clustering itself in this case. English language news articles from the *Botswana Daily News*, which are available online at the Botswana Government website, will be used in the current study. The first phase of our study will thus be concerned with downloading the required data from the website. The document pre-processing phase is concerned with getting the data ready for the document clustering algorithms. The vector-space model will be used for document representation. Finally, in the text mining phase, document clustering proper will be undertaken. In order to achieve methodological triangulation, and thus strengthen the findings, this study will use two document clustering approaches, the first of which will be partitional, the second hierarchical. We discuss our methodology further in Chapter 3.

1.6 Report Structure

The preferred structure of a thesis varies from discipline to discipline, as well as from institution to institution. However, the structure advocated by Perry (1995), which is in fact the basic structure of an empirical research report in the quantitative tradition, is particularly amenable to empirical studies regardless of the “home discipline” of the study, or even the paradigm (qualitative or quantitative) followed. The framework, which we follow closely, consists of five chapters: Chapter 1 - Introduction; Chapter 2 - Research Issues / Literature Review; Chapter 3 - Methodology; Chapter 4 - Data Analysis; and Chapter

⁰<http://www.gov.bw/cgi-bin/news.cgi>

5 - Conclusions and Implications. The main purpose of the current chapter, Chapter 1, was to introduce the study. In Chapter 2 we present the vector space model and cluster analysis, which together form the foundation upon which this study rests. The methodology adopted in the study is presented in Chapter 3, while in Chapter 4 we discuss the ways in which we analysed our data. Finally, in Chapter 5, we summarise work undertaken in the current study, present our findings, and suggest avenues for further research.



Chapter 2

Foundations

2.1 Introduction

In this chapter, we develop the theoretical foundation upon which the remainder of this thesis is based. Cluster analysis seeks to group together objects that are similar. In our case, the objects in question are textual documents. Thus, we first review the way in which documents are commonly modelled in information retrieval, before proceeding to discuss cluster analysis in some detail; this approach allows us to keep in perspective what we seek to cluster even as we explore the various aspects of cluster analysis. While a number of information retrieval models are discussed in the literature, the vector space model (Salton and McGill, 1983) is arguably the most ubiquitous. The main attractions of this model are its simplicity and elegance; documents are modelled as vectors (Baeza-Yates and Ribeiro-Neto, 1999), making it possible to apply concepts from linear algebra. In particular, the similarity between documents is usually measured in terms of the angles between the resulting vectors. The process of moving from raw documents to document vectors includes a number of steps, such as tokenisation, removal of stop words, stemming, and term weighting, all of which are discussed in this chapter. In this chapter, we also provide an overview of cluster analysis, providing both an intuitive perspective, and a formal statement of the clustering problem. Within the literature, a number of clustering schemes are discussed. In order to achieve methodological triangulation, this study proposes to use hierarchical and partitional clustering approaches; accordingly, each of these schemes is discussed in some depth. We conclude the chapter with a discussion of evaluation (without which any cluster analytic study would be incomplete). As our study deals exclusively with textual documents, all references to documents in this chapter, and indeed in the rest of this thesis, should be understood as referring to textual documents.

2.2 Document Modelling

2.2.1 Information Retrieval Models

In order to manipulate documents (or their surrogates) on a computer, it is necessary to model them in some way. Document modelling falls under the purview of information retrieval. A number of models are discussed in the extant literature. In general, though, an information retrieval model consists of four components (Baeza-Yates and Ribeiro-Neto, 1999:23): (i) a set of logical views (or representations) for the documents in the collection; (ii) a set of logical views (or representations) for queries; (iii) a framework for modelling document representations, queries, and their relationships; and (iv) a ranking function which defines an ordering among the documents with regard to the query. Traditionally, a document has been viewed logically as a set of index words; typically nouns, because they are believed to carry the semantics of the document, are used (Baeza-Yates and Ribeiro-Neto, 1999). Baeza-Yates and Ribeiro-Neto also make the point that increasingly, especially in the context of web search engines, all the words in the document are used to index it, in which case the logical document view is referred to as full text. While Korfhage (1997) does discuss the pros and cons of considering queries to be documents, we agree with Subrahmanian (1998:147) that, ultimately, “a query is nothing but a small document”, so that “the result of a query is a set of documents similar to it”. Thus logically, both documents and queries can be viewed as collections of index terms. As we have already intimated, the particular model used in this study is the vector space model, which we now turn to.

2.2.2 The Vector Space Model

Most research work in Information Retrieval is based on the vector space model (Tombros, 2002). In this model, each document is represented as a simple vector using distinct terms from the document collection; the terms themselves could be all the words from the document collection (full text logical view), selected index terms, or even stems arrived at using some stemming algorithm (Baeza-Yates and Ribeiro-Neto, 1999). In the vector space model, “all the structure and linear ordering of words within the context is ignored”; for this reason, it is sometimes referred to as a “bag of words” approach (Manning and Schutze, 1999:237). We introduce the vector space model through the use of an example which we borrow from Sahami (1998).

Consider the following documents:

Document 1: “Computing is not about computers any more. It is about living.”

Document 2: “To live is to compute.”

These documents would be represented using the simple term-by-document matrix depicted in Table 2.1. The entries in the ‘term’ column represent all

Term	Document 1	Document 2
About	2	0
Any	1	0
Compute	0	1
Computers	1	0
Computing	1	0
Is	2	1
It	1	0
Live	0	1
living	1	0
More	1	0
Not	1	0
To	0	2

Table 2.1: A simple term-by-document matrix representation of two documents

[Source:Sahami (1998:19)]

the distinct terms in the document collection, and the numbers in columns 1 and 2 are counts of the occurrences of the terms in each of the sample documents: for instance, the term ‘is’ appears twice in document 1 and once in document 2, while ‘living’ appears once in document 1 and not at all in document 2.

Individual documents from Table 2.1 may be represented as column vectors:

$$d_1 = \begin{pmatrix} 2 \\ 1 \\ 0 \\ 1 \\ 1 \\ 2 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} ; \text{ and } d_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 2 \end{pmatrix}$$

In the vector space model, terms are weighted in some way. In the case of vectors d_1 and d_2 above, the weighting is simply the raw frequencies of the index terms in each document. Other term weighting schemes are possible; Lan *et al.* (2005) evaluate ten such schemes in the context of text categorisation using support vector machines. For instance, one might use binary vectors where instead of representing word counts, vector entries represent the presence (binary weight 1) or absence (binary weight 0) of terms in the document; using

this approach document 1 in the previous example would be represented by the vector (depicted here as a row, rather than column, vector to conserve space):

$$d_1 = (1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0)$$

One obvious weakness with the binary weighting scheme is that it ignores the fact “that terms that occur frequently within a document may reflect its meaning more strongly than terms that occur less frequently, and should thus have higher weights” (Jurafsky and Martin, 2000:651), which the raw term frequencies do capture. Raw frequencies, however, do not take into cognisance the differences in document lengths; that term t appears more frequently in document d_1 than it does in document d_2 may merely be due to d_1 being longer than d_2 , and not be a reflection of what the documents are about per se. The usual way of tackling this issue is through normalising the document vectors i.e. converting all the vectors to a standard length, by dividing each of the vector’s dimensions by the overall length of the vector.

A popular term weighting approach (see, for instance, Jurafsky and Martin (2000); Baeza-Yates and Ribeiro-Neto (1999)), which we use in this study, is the so called *tf.idf* measure, where the term frequency *tf* is multiplied by the inverse document frequency factor *idf*. The *tf* factor is a measure of the frequency of a particular term in a particular document; conceivably, the raw term frequency can be used, but for reasons alluded to earlier, some normalisation is usually undertaken. Intuitively, “terms that are limited to a few documents are useful for discriminating those documents from the rest of the collection”, while “terms that occur frequently across the entire collection are less useful in discriminating among documents” (Jurafsky and Martin, 2000:651). The *idf* factor is a measure that attempts to capture this intuition, and it is computed according to the following formula:

$$idf_i = \log \left(\frac{N}{n_i} \right)$$

where N is the total number of documents in the collection, and n_i is the number of documents containing term i . Notice that when n_i is small *idf* is large, and when n_i is large *idf* is small, aptly capturing the ‘intuition’ expressed earlier. The use of the logarithm is intended to tone down the effect of the large number of documents in many a collection (Jurafsky and Martin, 2000). Thus, the weight $w_{i,j}$ of term i in document j is given by:

$$w_{i,j} = tf_{i,j} \times \log \left(\frac{N}{n_i} \right)$$

2.2.3 Measuring Similarity

The concept of similarity lies at the heart of information storage and retrieval; in Korfhage's opinion (1997:125), it is "probably the single key concept behind information storage and retrieval". Earlier, we identified "a ranking function which defines an ordering among the documents with regard to the query" (Baeza-Yates and Ribeiro-Neto, 1999:23) as the fourth component of an information retrieval model. Such ranking is based on the similarity of the various documents to the query. Similarity also lies at the heart of cluster analysis. Indeed, "all clustering algorithms use a similarity measurement between vectors to compare patterns of expression" (Sturn, 2000:14). Simply put, "interobject similarity is a measure of correspondence or resemblance between objects to be clustered" (Hair *et al.*, 1995:429); of course, the objects in question could quite easily be - and in the case of our study, actually are - documents. Rather than discuss similarity separately under information retrieval models and cluster analysis, this section will attempt to be broad-based and address aspects of similarity as they arise in both contexts.

In practice, one can measure either the similarity, or the dissimilarity of objects; the term proximity is often used to cover both similarity and dissimilarity (Gordon, 1981; Everitt, 1993; Hand *et al.*, 2001). Van Rijsbergen (1979) uses another term, namely association, which he sees as meaning exactly the same thing as similarity, with the exception that the former is reserved for discrete-state (i.e. categorical) variables. We will return to the similarity / dissimilarity question shortly, and stay, for now, with the broader proximity notion. Broadly, three types of proximity measures can be identified, namely, association measures, correlation measures and distance measures (Hair *et al.*, 1995). Association measures are used with non-metric (i.e. nominal or ordinal) variables. A simple association measure could be a count, possibly expressed as a percentage, of the number of variables in which two objects (expressed as vectors with the same dimensions) agree (Hair *et al.*, 1995), i.e. the number of times the characteristics of one case tallied with those of another. van Rijsbergen (1979) refers to such an association measure as a simple matching coefficient, and points out that in the case of textual documents, the "shared characteristics" would typically be index terms.

Correlation measures "... represent similarity by the correspondence of patterns across the characteristics [of the objects being compared] ... A correlational measure does not look at the magnitude of the values but instead the patterns of values" (Hair *et al.*, 1995:430); thus, such measures are concerned with whether corresponding variables in different cases (or tuples) both go up (or down) without necessarily considering the magnitudes of the said variables. As Hair *et al.* (1995) point out, "correlational measures are rarely used [in cluster analysis] because emphasis on most applications of cluster analysis is on the magnitude of the objects, not the patterns of values" (430).

Distance measures represent similarity as the proximity between obser-

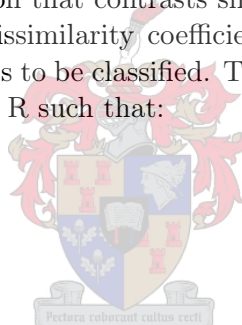
vation points in an n dimensional space; the closer two points are in an n -dimensional place, the more similar they are. The Euclidean distance measure is the most popular proximity measure for continuous, numeric data (Jain *et al.*, 1999). Given two vectors $u = (u_1, u_2, \dots, u_n)$ and $v = (v_1, v_2, \dots, v_n)$, the Euclidean distance d between them is calculated as:

$$d(u, v) = \sqrt{|u_1 - v_1|^2 + |u_2 - v_2|^2 + \dots + |u_n - v_n|^2}$$

The Euclidean distance is a straight-line distance measure i.e. it measures the straight-line distance between two points. Distance measures may be non-Euclidean, as is the case with the Manhattan (city-block approach), and the Minkowski approaches. We concur with Hand *et al.* (2001) that given the huge number of distance measures available, the challenge lies not with defining one, but with determining which is the most appropriate for the research problem at hand. This is an important point to keep in mind, for clustering algorithms use the proximity values, as opposed to the original data values, to try to uncover general relationships between the variables Gordon (1981). We close this subsection with a short discussion that contrasts similarity and dissimilarity.

Gordon (1981) defines dissimilarity coefficients in the following manner. Let S denote the set of objects to be classified. Then, a dissimilarity coefficient is a function d from $S \times S$ to \mathbb{R} such that:

1. $d_{ij} \geq 0 \forall i, j \in S$
2. $d_{ii} = 0 \forall i \in S$
3. $d_{ii} = d_{ji} \forall i, j \in S$



Distance measures which additionally obey the triangular inequality (i.e. $d_{ij} \leq d_{ik} + d_{kj} \forall i, j, k \in S$) are called *metric*. Notice that, according to this definition the distance measures we discussed above are, in fact, dissimilarity measures. Note also that bigger distances between objects implies bigger dissimilarities, or, conversely, smaller similarities. Transforming from dissimilarity s_{ij} to dissimilarity d_{ij} can be accomplished with an appropriate formula; the examples that Gordon (1981) gives are:

$$s_{ij} = \frac{1}{1 + d_{ij}}$$

and

$$s_{ij} = c - d_{ij}$$

for some constant c .

We have so far been discussing proximity measures generally. We will now turn to the use of proximity measures in information storage and retrieval.

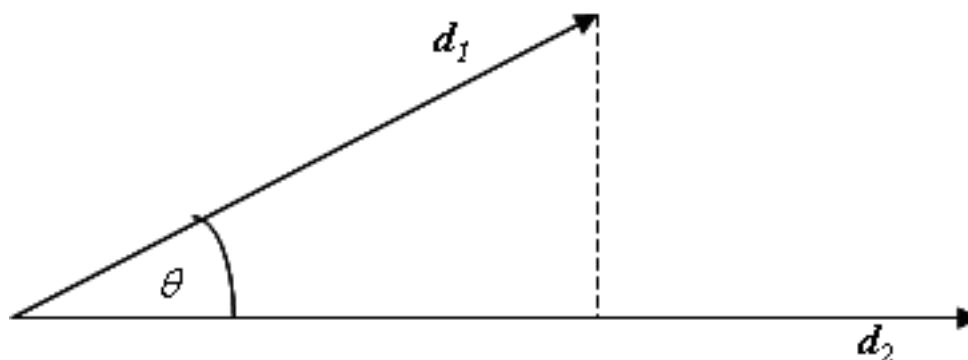


Figure 2.1: The Cosine correlation coefficient

[Source: modified slightly from Baeza-Yates and Ribeiro-Neto (1999:28)]

2.2.4 Commonly Used Proximity Measures In Information Retrieval

Distance measures are frequently used in cluster analysis. However, in Information Retrieval, the most popular measure of similarity is the cosine coefficient (Sahami, 1998). It is given by the formula

$$\cos\theta = \frac{d_1 \bullet d_2}{\|d_1\| \bullet \|d_2\|}$$

where d_1 and d_2 are the vectors representing the documents being compared.

As can be seen from the above formula, the cosine similarity measure is an angular measure; it measures the cosine of the angle between the document vectors (recall that we consider a query to be a document). Notice looking at Figure 2.1 that the cosine measure, being a correlation measure, ignores the magnitude (i.e. length) of the vectors in favour of the angle between them.

Korfhage (1997) argues that angular measures, such as the cosine coefficient, are *extrinsic* i.e. they represent a view of the document space from some origin. Consequently, if such origin were to be moved (the example that Korfhage uses here is indexing the documents quite differently), then the angles between documents might also change quite significantly. Distance measures, on the other hand, are said to be *intrinsic* i.e. “based solely on the group of documents under consideration”; thus, “from a given point in the document space all directions are considered equal”, and “similarity depends only on how far a given document is from the point” (Korfhage, 1997:85). Korfhage also posits that because angular measures do not consider distance, documents lying on the same vector will be considered similar even if they lie far apart in the document space. Other measures of similarity often used in Information Retrieval are the simple matching coefficient, Dice’s coefficient, Jaccard’s coefficient, and

the Overlap coefficient; the last three may be considered normalized versions of the simple matching coefficient (Van Rijsbergen, 1979). These measures are not discussed further in this report; the appropriate formulae can, however, be found in Van Rijsbergen (1979).

2.2.5 Other Information Retrieval Models

We have pointed out that the vector space model is the most popular of all information retrieval models. However, “a large variety of alternative ranking methods have been compared to the vector model but the consensus seems to be that, in general, the vector model is superior, or almost as good as any known alternatives. Furthermore, it is simple and fast. For these reasons, the vector model is a popular retrieval model nowadays” (Baeza-Yates and Ribeiro-Neto, 1999:30). In our study, we only use the vector space model. We thus do not discuss other information retrieval models. Nevertheless, Baeza-Yates and Ribeiro-Neto (1999) do provide a comprehensive coverage of information retrieval models, including, among others, the vector space model, as well as the Boolean and probabilistic models.

Our discussion of document modelling assumed that we had somehow selected the index terms representing our documents; in the next section, we discuss index term selection, highlighting the activities that are typically undertaken in this process.

2.3 Index Term Selection

2.3.1 Lexical Analysis



Index term selection begins with tokenisation, which breaks a document into its constituent terms. In general, in the English language, a white space (i.e. space or tab) indicates the ending of one word and the start of another (Manning and Schutze, 1999). However, as Manning and Schutze themselves point out, in reality things are much more complicated than that. In particular, Baeza-Yates and Ribeiro-Neto (1999) draw attention to four cases that require special attention, namely, digits, hyphens, punctuation marks, and the case of letters (lower and upper case). While numbers on their own may be too vague to be of any use, in some contexts, such as dates, they will be quite useful. Hyphens can be used to break lines in order to improve text justification, to indicate that two otherwise distinct ‘tokens’ should be treated as one (e.g. non-lawyer), as well to indicate the correct grouping of words (Manning and Schutze, 1999). Words may be surrounded not by spaces, but by punctuation marks, such as commas, semicolons, apostrophes and full stops. The punctuation marks themselves will serve different functions in different circumstances, and so cannot be naively discarded; a full stop may indicate the end of a sentence, or the use of an abbreviation. Nevertheless, the usual practice is to remove punctuation marks

entirely in lexical analysis (Baeza-Yates and Ribeiro-Neto, 1999). The case of letters may convey important information; for instance, the capital ‘b’ in Banks suggest a proper noun, perhaps a name of a human being. Nevertheless, all tokens are typically converted to lowercase in a process called case folding.

Synonyms and *homonyms* raise interesting challenges in information retrieval. Synonyms are words that have the same (or nearly the same) meanings. Homonyms, on the other hand, are “words that are written the same way, but are (historically or conceptually) really two different words with different meanings which seem unrelated” (Manning and Schutze, 1999:110), such as ‘bank’ for river bank, and ‘bank’ in reference to a financial institution. In manual indexing approaches, synonyms and homonyms are addressed through the use of a thesaurus. In the bag-of-words approach we use in this study, synonyms will simply become different dimensions of the vector space, while homonyms will be collapsed into one dimension.

An important consideration which governs the choice of index terms is the frequency of individual terms within the document collection. According to Van Rijsbergen (1979), Luhn, writing in the 1950s, postulated that the frequency of a term within, say, an article, is related to the term’s significance within the article; in other words, the higher the term’s frequency in the article, the more likely is it that the article is about the term. However, it is also true that in any given text, the most frequent words are *glue words*, such as prepositions, that, while their absence would render the document unreadable, give no indication of what the document is about; indeed, “many studies have shown that the most common 250 to 300 words in English may account for 50% or more of a given text” (Korfhage, 1997:134). Korfhage identifies two problems associated with such common words. Firstly, because these words have such high frequencies, they tend to dwarf other, possibly content-bearing words, in any retrieval system based on word frequencies. Secondly, although these words are virtually useless for document discriminating, a lot of processing is likely to be wasted on them. At the other extreme, are words that are so rare as to contribute only insignificantly to the content of the document. Zipf’s Law, which we discuss next, is often used to determine the upper and lower cut-off points. Other common activities are the removal of stop words, as well as stemming.

2.3.2 Zipf’s Law

Zipf’s Law suggests that an approximate relationship exist between the rank r of a term and that term’s frequency f in a given text such that

$$f \times r \approx k$$

for some constant k (Korfhage, 1997); rank refers to the position of the term under consideration when all the terms in the text are ordered according

to their frequency of occurrence in the text. When using Zipf's Law, "a certain arbitrariness is involved in determining the cut-offs. There is no oracle which gives their values. They have to be established by trial and error" (Van Rijsbergen, 1979). Sahami (1998) used Zipf's Law to eliminate all terms that occurred fewer than 10 times in the document collection.

2.3.3 Stemming

Stemming seeks to collapse "morphological variants of a lexical item . . . into a single root form" (Jurafsky and Martin, 2000); for instance, running and run will both be reduced to run. Although stemmers are usually associated with suffix removal, Korfhage (1997) argues that there is really no reason why they should not do prefix stripping as well. Indeed, Baeza-Yates and Ribeiro-Neto (1999:168) formally define a stem as "the portion of a word which is left after the removal of its affixes" (i.e., prefixes and suffixes). To motivate stemming, consider the example developed by Korfhage (1997). The words computer, computers, computing, compute, computes, computed, computational, computationally, and computable express closely related concepts, making it desirable that searching with one of the terms should return documents containing the other terms; stemming makes this possible by reducing all these terms to a single root. Thus, in the context of information retrieval, stemming has the effect of broadening searches, which may sometimes be problematic. For instance, Jurafsky and Martin (2000) recount the case of a web search engine that, as a consequence of conflating stocks and stockings to the same root form, returned salacious sites about stockings in response to queries about stock prices.

Commonly, stemming algorithms use rules of the form (Belew, 2000):

Rule 2.1 (.*)SSES -> /1SS

As Belew explains, this rule requires that the ending SSES be transformed to SS; for instance hisses will be transformed to hiss. The complete stemming algorithm will contain many such rules. While a number of stemming algorithms have been developed over the years, Korfhage (1997:29) observes that the Lovins and Porter stemmers have been "well tested" and are thus "widely accepted". Porter's algorithm appears to be the more popular of the two "because of its simplicity and elegance" (Baeza-Yates and Ribeiro-Neto, 1999).

2.3.4 Stop List

Stop words are words that occur frequently in the document collection, and so have no value in retrieval (Large *et al.*, 1999). A list of stop words, a "negative dictionary" (Korfhage, 1997) of sorts, is usually used to exclude common words from any processing. Typically, *grammatical* or *function* words, such as

articles, prepositions, and conjunctions are included in the list of stop words (Manning and Schütze, 1999; Baeza-Yates and Ribeiro-Neto, 1999). Baeza-Yates and Ribeiro-Neto (1999) further suggest that in order to increase the index compression benefits of stop word elimination, other words, such as verbs, adverbs and adjectives, should also be considered for inclusion in the stop list. It is important to note, though, that stop word removal does also have its downside. Jurafsky and Martin (2000) indicate, for instance, one common stop list would reduce the phrase *to be or not to be* to the single word *not*, making it virtually impossible to find documents containing the original phrase using keyword-based searching.

2.4 Cluster Analysis

The aim of cluster analysis is to “[decompose] a (usually multivariate) data set into groups so that the points in one group are similar to each other and are as different as possible from the points in other groups” (Hand *et al.*, 2001:203); the terms sometimes used to denote these concepts are, respectively, internal cohesion and external isolation (Everitt, 1993). Internal cohesion is a measure of within-cluster similarities, while external isolation measures between-cluster dissimilarities; clusters are internally cohesive if points within a cluster are similar to each other, and they are externally isolated if each cluster is clearly dissimilar from other clusters. These concepts are illustrated in Figure 2.2. The clusters in Figure 2.2 (a) exhibit both internal cohesion and external isolation, while in Figure 2.2 (b) the clusters are externally isolated without being internally cohesive.

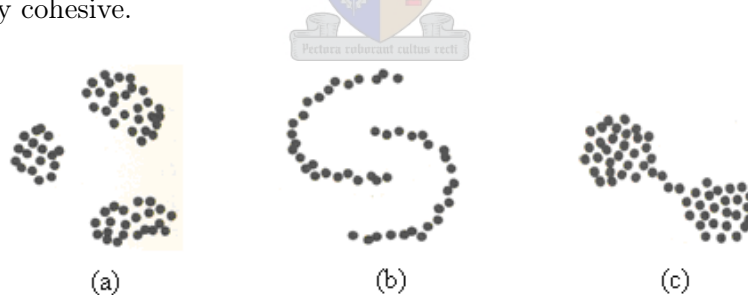


Figure 2.2: Internal cohesion and external isolation

(a) clusters are cohesive and isolated; (b) clusters are isolated but not cohesive; (c) two cohesive clusters linked by several intermediate points.

[Source: Gordon (1981:5)]

While the two clusters in Figure 2.2(c) are internally cohesive, they are linked by several points not falling in either of the clusters. Cases (data points) not belonging to any particular cluster are sometimes referred to as the entropy group (Hair *et al.*, 1995). It is important to distinguish clustering from either

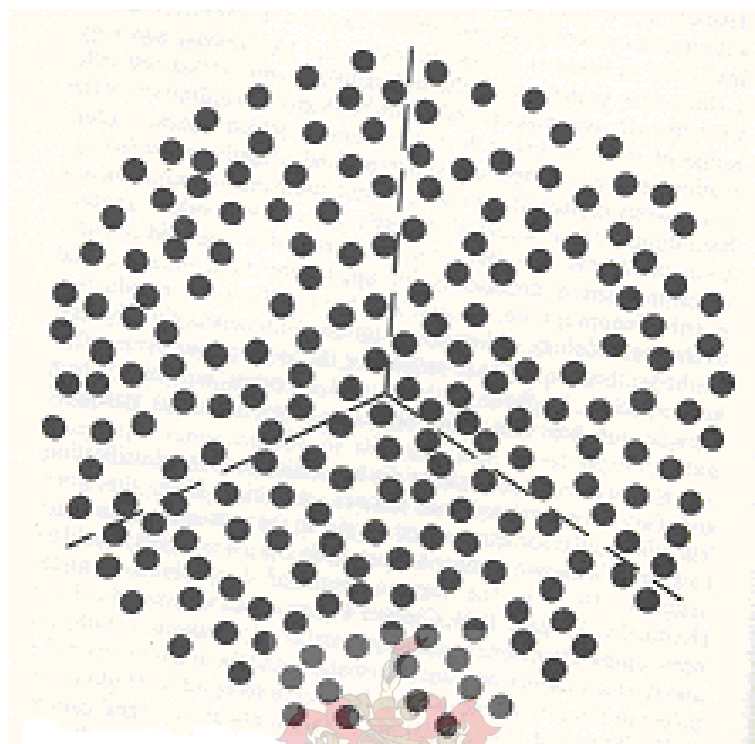


Figure 2.3: Dissection of a set of objects into sectors

[Source: Gordon (1981:4)]

segmentation, which simply partitions data on the basis of convenience (Hand *et al.*, 2001), or classification (in which previously unseen cases are assigned to previously defined groups). Clustering is also different from dissection, which simply divides a homogenous data set into different parts (Everitt, 1993). The illustration of dissection in Figure 2.3 comes from Gordon (1981). Many clustering algorithms will actually divide the data into groups, raising the possibility of misinterpretation as the investigator typically does not know the structure of the data *a priori* (Everitt, 1993). Thus, when undertaking cluster analysis, the researcher needs to always bear in mind that while cluster analysis is usually thought to be ‘structure seeking’, as Hair *et al.* (1995) point out, it actually imposes a structure through the selected methodology i.e. different clustering algorithms will (potentially) identify different cluster configurations.

Formally, the clustering problem may be stated as follows (Dunham, 2003:127):

Given a database $D = t_1, t_2, \dots, t_n$ of tuples and an integer value k , the *clustering problem* is to define a mapping $f : D \rightarrow 1, \dots, k$ where each t_i is assigned to one cluster $K_j, 1 \leq j \leq k$. A cluster, K_j , contains precisely those tuples mapped to it; that is, $K_j = \{ t_i | f(t_i) = K_j, 1 \leq i \leq n, \text{ and } t_i \in D \}$.

The mapping f merely determines the cluster (i.e. cluster 1, cluster 2, ... cluster k) to which each tuple should be assigned. Of course, in the context of document clustering, the tuples t_1, t_2, \dots, t_n are individual documents (typically expressed as document vectors), while the integer k represents the number of clusters to be formed. A number of clustering algorithms are presented in the literature; for document clustering, hierarchical clustering algorithms, as well as the k-means algorithm, appear to be the most popular (Steinbach *et al.*, 2000; Belew, 2000). Consequently, our discussion of document clustering will be limited to these two approaches.

2.5 Hierarchical Clustering

Hierarchical clustering algorithms get their name from the *dendrogram*, an example of which is shown in Figure 2.4, they construct as they work their way through the data. They can further be divided into two groups, namely, *agglomerative* and *divisive* hierarchical clustering algorithms. Looking at Figure 2.4, in the case of agglomerative methods, we start with six clusters, consisting of the individual data points A, B, C, D, E and F. In the next stage, clusters A and B are merged to form cluster H; the next time around clusters C and D are merged to form cluster I. At each stage, the two nearest clusters are merged to form one large cluster; thus, with each pass of the algorithm, the number of clusters is reduced by one. Notice that initially, each individual point is considered a cluster and that ultimately, one cluster, made up of all the data points, is formed. It is this merging of clusters that give agglomerative algorithms their name. Divisive algorithms, which also get their name from the way they work, move in the opposite direction i.e. initially, all the data points are considered to be in one large cluster; the cluster is then divided based on the notion of dissimilarity, with the most dissimilar points split off to form clusters of their own. Looking again at Figure 2.4, agglomerative hierarchical clustering algorithms would move from the bottom to the top, while their divisive counterparts would move from the top to the bottom. Clusters produced through that hierarchical clustering are nested; for instance, cluster L contains clusters F and K, with cluster K in turn subsuming clusters H and J. This makes hierarchical clustering algorithms ideal for creating taxonomies; “up” represents the direction more “generality”, while down represents the direction of more “specificity”.

Cluster merging joins together the two closest clusters. There are a number of ways in which the distance between clusters may be calculated; the popular ones are *single linkage*, *complete linkage*, *average linkage*, *Ward’s method*, and the *centroid method* (Hair *et al.*, 1995). Single linkage uses the minimum distance between two clusters, calculated as the shortest distance between any data point in the first cluster to any other data point in the second cluster; complete linkage only differs from single linkage in that it uses the maximum

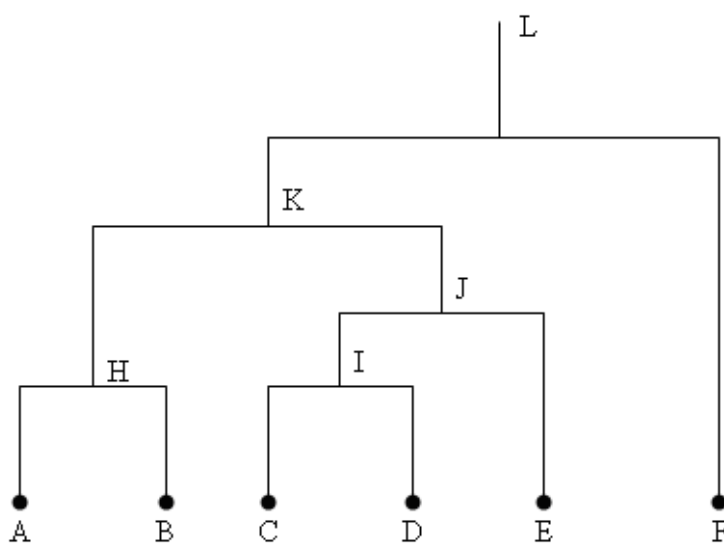


Figure 2.4: Dendrogram illustrating hierarchical clustering

[Source: modified from Dunham (2003:131)]

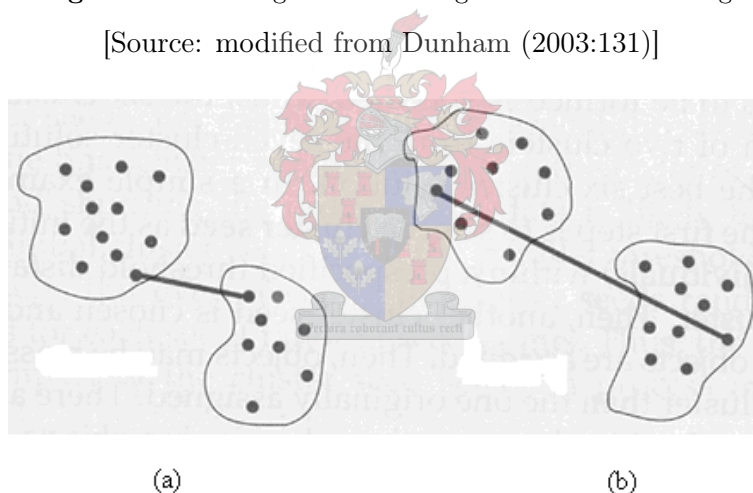


Figure 2.5: Measuring distance between clusters

(a) single linkage and (b) complete linkage. [Source: Hair *et al.* (1995:439)]

(as opposed to minimum) distance between any data point in the first cluster and any data point in the second cluster. Single and complete linkage are illustrated in Figure 2.5.

Single and complete linkage approaches are sometimes referred to as the *nearest* and *furthest neighbour* approaches, respectively. Although conceptually easy to understand, single and linkage measures are not without problems. In single linkage since one small distance is enough to force two otherwise different clusters to merge, the tendency is for the resulting clusters to form long

chains, with points at opposite ends of the chain very dissimilar; in complete linkage only one large distance is enough to prevent clusters from merging, and so clusters tend to be small and merged together very late with a great error value (Sturn, 2000; Hair *et al.*, 1995). Both single and complete linkage procedures calculate the distance between two clusters based on extreme (smallest or largest) values; average linkage, Ward's method, and the centroid method use all the data points within a cluster.

In the average linkage procedure the distance between two clusters is calculated as the average distance from all individuals in one cluster to all individuals in the other cluster. The method works as equally when the objects form natural distinct "clumps", as with elongated, "chain" type clusters (Sturn, 2000). A variation of the average link procedure weights the distance calculation by the size of the clusters. Sturn recommends using this method when the cluster sizes are suspected to be greatly uneven. In the centroid method in which the distance between two clusters is simply calculated as the distance between their centroids, with the centroid itself calculated as mean of the variables in the cluster variate. Conceptually, the centroid is middle of the cluster, and need not be an actual datum (Dunham, 2003). Dunham further explains that in some algorithms an actual data point is chosen to represent the cluster; such a data point is called a *medoid*. In these algorithms, the medoid, rather than the centroid, distance is used, so that the distance between two clusters is essentially the distance between their medoids. Similar to the average link procedure, the centroid (or medoid) calculations may be weighted by the cluster sizes. Finally, Ward's method calculates the distance between two clusters as "the sum of squares between the two clusters summed over all variables". Ward's method uses an analysis of variance approach, and as such, tends to be very efficient, partitioning the data into small clusters of roughly equal size (Sturn, 2000).

2.6 Partitional Clustering

Partition-based clustering seeks to "partition a data set into k disjoint sets of points such that the points within each set are as homogenous as possible" (Hand *et al.*, 2001:296). A commonly used partition-based clustering algorithm is k -means algorithm, depicted in Table 2.2.

In the k -means algorithm, the number of clusters to be created (step 1) must be stated by the user *a priori* the clustering process itself (step 3). Roiger and Geatz (2003) suggest running the algorithm several times with alternative values of k , and eventually selecting the best value for k . Notice also that in step 2, the initial cluster centres are specified by choosing K instances from the data at random. Bacher (2002) suggests two other ways in which the initial cluster centres may be selected, namely, (i) specifying starting values for the K cluster centres (as opposed to specific cases), and (ii) using starting values

- | |
|--|
| <ol style="list-style-type: none"> 1. Choose a value for K, the total number of clusters to be determined. 2. Choose K instances (data points) within the dataset at random. These are the initial cluster centres. 3. Use simple Euclidean distance to assign the remaining instances to their closest cluster centre. 4. Use the instances in each cluster to calculate a new mean for each cluster. 5. If the new mean values are identical to the mean values in the previous iteration the process terminates. Otherwise, use the new means as cluster centres and repeat steps 3 - 5. |
|--|

Table 2.2: The k-means Algorithm

[Source:Roiger and Geatz (2003:84)]

- | |
|---|
| <ol style="list-style-type: none"> 1. Pick a cluster to split. 2. Find 2 sub-clusters using the k-means algorithm (bisecting step). 3. Repeat step 2, the bisecting step, ITER times and take the split that produces the highest overall similarity. 4. Repeat steps 1, 2 and 3 until the desired number of clusters is reached. |
|---|

Table 2.3: The Bisecting k-means Algorithm

[Source:Steinbach *et al.* (2000)]

obtained from a hierarchical clustering procedure. Though the basic k-means algorithm is “simple and effective” (Roiger and Geatz, 2003:84), it does have a number of weaknesses (Berry and Linoff, 1997): (i) it is not suited to handling overlapping clusters; (ii) the clusters are easily pulled off centre by outliers; and (iii) each record is either in or out of a cluster; there is no notion of some records being more or less likely than others to really belong to the cluster to which they have been assigned. When clustering text, the K means algorithm is normally modified to use the cosine measure to measure similarity, in which case it is referred to as the spherical k-means algorithm. Another variation of the simple k-means algorithm is the bisecting k-means algorithm; the complete algorithm is given in Table 2.3. The initial cluster includes all documents; subsequently, choosing the cluster to split may be based, for instance, on it being the largest or the one with the least similarity or even combining the two criteria (Steinbach *et al.*, 2000).

2.7 Evaluating and Interpreting Cluster Analysis

Cluster evaluation and interpretation are an integral part of any cluster analytic study. When using the k-means algorithm, we seek to find the best value of K , while in hierarchical clustering the aim is to determine the level in the hierarchy that contains the best clusters (Berry and Linoff, 1997:210). A good clustering solution should yield clusters that exhibit external isolation and internal cohesion. Everitt (1993) provides a useful, if rather short, list of questions that evaluation tries to answer: Are the clusters real or merely artefacts of the algorithms? Do other solutions exist which are better? Can the clusters be given a convincing interpretation? Since, by definition, the labelling of clusters is not known *a priori*, interpreting the results of cluster analysis is not a trivial undertaking; domain expertise, and other text mining approaches, particularly association rule mining, may prove useful (Berry and Linoff, 1997). Although Bacher (2002) insists that clusters should have a substantive interpretation, Berry and Linoff point out that, particularly in commercial applications, cluster analysis can be profitable even if only some - perhaps just one - of the clusters are successfully interpreted!

Evaluating cluster analysis consists of the following steps (Bacher, 2002): (i) determination of the number of clusters, (ii) substantive interpretation of clusters, (iii) test of stability and (iv) test of internal validity, relative validity and external validity. In the k-means algorithm, for instance, the user is expected to specify the number of clusters into which the data are to be partitioned. To determine the number of clusters when using hierarchical clustering algorithms, Bacher suggest an inspection of the produced dendrogram, or, alternatively, an inspection of the agglomeration levels; in the former, the user is guided by the “small hills” that are in fact the clusters proper, while in the latter sharp changes between levels are suggestive of boundaries between clusters. According to Bacher, a clustering solution is stable if small modifications in the data and methods do not change the results. Bacher points out, however, that in the case of k-means clustering, the technique and the distance measure are fixed, with the starting partition being the only factor that can be varied. Suppose the initial partitions were randomly generated; then a simple way of changing the starting partitions would entail generating different sets of starting partitions and comparing the resulting solutions. Comparing clustering solutions may entail comparing cluster centres, or the clusters to which each case has been assigned (Bacher, 2002). In the case of hierarchical clustering, Bacher suggests testing stability by varying the technique (e.g. single linkage, complete linkage) and the similarity / dissimilarity measures used.

Internal validity attempts to assure internal cohesion and external isolation. In the case of k-means clustering, for instance, the variance may be used to measure within-cluster similarity (i.e. internal cohesion) (Berry and

⁰Though, we would argue, not necessarily in that order!

Linoff, 1997). A further requirement of internal validity is that “the classification should fit the data” i.e. “the classification should be able to explain the variation in the data” (Bacher, 2002:18). Relative validity is concerned with ensuring that indeed the data can be clustered (rather than just bisected) in some way, and also requires that the selected classification should be better than others.

2.8 Summary

In this chapter, our aim was to present the theoretical scaffolding for the rest of the study. Our study uses cluster analysis to explore a body of text, namely, news stories that appeared in the *Botswana Daily News* during the period January - December 2004. In order to do this, it is necessary that documents be modelled in some way. While a number of information retrieval models are discussed in the literature, the most frequently cited is the vector space model in which documents are viewed as term vectors in a high dimensional space. The similarity between documents can then be measured in a number of ways, with the cosine measure being the most frequently used. Selecting the terms with which to index documents involves a number of steps: tokenisation breaks the text into individual words, Zipf’s Law and stop word removal serve to select only the most useful terms to include, and stemming reduces words to their root forms. Zipf’s Law, stop word removal and stemming all serve to reduce the number of dimensions in the document vector space. This chapter also reviewed cluster analysis, with particular emphasis on document clustering. While a number of clustering algorithms have been developed over the years, with new ones being developed all the time, our focus was on agglomerative hierarchical clustering and the k-means (and its variation, the bisecting k-means) algorithm, both of which have been used widely and successfully for clustering textual data. We also provided a review of cluster evaluation techniques. In Chapter 1 we provided a brief synopsis of the methodology followed in this study; building upon this, in the next chapter, we present an overview of the actual work undertaken in this study.

Chapter 3

Methods

3.1 Introduction

In the previous chapter, we presented the theoretical basis for the work undertaken in this study. In this chapter, we discuss the methodology adopted in the study, including the actual work undertaken as part of the study, leaving the data analysis and interpretation for chapter 4. Data mining is usually considered part of the wider Knowledge Discovery in Databases (KDD) process that “attempts to extract implicit, previously unknown, and potentially useful knowledge from data” (Roiger and Geatz, 2003:148). A number of KDD process models are discussed in the literature. Some players within the data mining industry, including SPSS and Daimler-Chrysler, have come together to develop a generic KDD process model referred to as the Cross Industry Standard Process for Data Mining, or CRISP-DM for short (Chapman *et al.*, 2000); the model is generic in the sense that it can be used in any data mining project, regardless of the application domain. CRISP-DM consists of six phases, namely, (i) business understanding, (ii) data understanding, (iii) data preparation, (iv) modelling, (v) evaluation, and (vi) deployment. Other process models are quite similar to CRISP-DM; for instance, Roiger and Geatz (2003) propose a seven-phase model consisting of (i) goal identification, (ii) creating a target set, (iii) data preprocessing, (iv) data transformation, (v) data mining, (vi) interpretation and evaluation, and (vii) taking action. The point to note about these KDD process models, and others discussed in the literature, is that data (or text) mining is but a step in the KDD process; for instance, in CRISP-DM, data mining occurs at phase (iv), while in Roiger and Geatz’s KDD process model, it occurs at phase (v). We concur with Roiger and Geatz that KDD is the application of the scientific method to data mining.

The approach we take in this study is closely modelled around the three-phase Knowledge Discovery in Text (KDT) process discussed in Karanikas and Theodoulidis (2002), which is really a simplified version of the more elaborate KDD process models discussed in the literature, such as the two that we al-

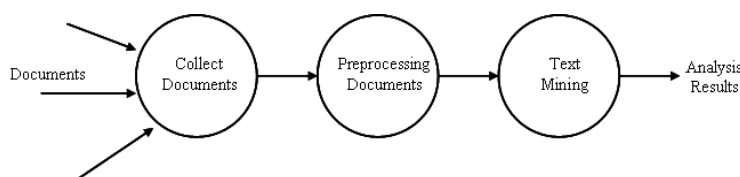


Figure 3.1: Main stages of the KDT process

[Source: Modified slightly from Karanikas and Theodoulidis (2002:4)]

luded to above. The KDT process consists of three phases, namely, (i) data collection, (ii) document pre-processing, and (iii) text mining (see Figure 3.1). The data collection phase entails first deciding what data to use in the study, and then actually collecting such data. Our study seeks to explore news stories published in the *Botswana Daily News* in the period January - December, 2004. The paper publishes daily and typically carries each story in two languages, namely English and Setswana. We limited ourselves to the English language bulletins, which are conveniently available electronically¹ as part of the online edition of the *Botswana Daily News*. We have written a fairly simple program, using the C# programming language, to download the requisite data from the *Botswana Daily News*'s website. The C# language was chosen because of its similarity with the C programming language, with which we were already familiar, and, equally important, the ease with which Internet client programs can be written in it. The actual source code for the program, together with accompanying comments, is given in Appendix A (Source Code Listings). The program, and by extension, the data collection phase, is quite straightforward and will not be discussed any further; rather, we will focus on the document pre-processing and text mining (i.e. cluster analysis) phases. Document pre-processing is concerned with getting the data ready for the cluster analysis phase, and involves a number of activities which are covered at length in this Chapter.

In our study, Phase III of Karanikas and Theodoulidis (2002)' KDT process, text mining, consisted entirely of cluster analysis. Motivated by the desire to strengthen our research findings, we experimented with three clustering approaches, namely, direct k-way clustering, k-way clustering through repeated bisections, and hierarchical agglomerative clustering. The clustering software that we use is called CLUTO (Clustering Toolkit), developed by Professor Karypis and his colleagues at the Department of Computer Science and Engineering, University of Minnesota (Karypis, 2003). We found this software attractive for three reasons. Firstly, it implements the particular clustering schemes we wished to use in our study. Secondly, it is geared towards the analysis of large, high dimensional data sets that typically arise in document clustering projects, such as this one. Finally, the software is freely available

¹<http://www.gov.bw/cgi-bin/news.cgi>

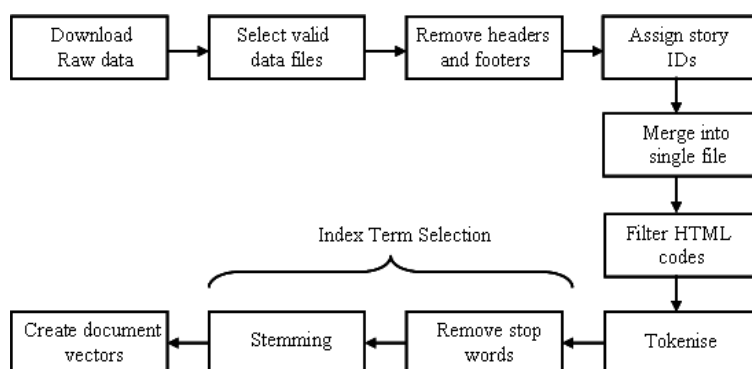


Figure 3.2: Document Preprocessing Activities

from its developers . A complete coverage of CLUTO can be found in Karypis (2003).

3.2 Document Preprocessing

In the three-phase KDT process of Karanikas and Theodoulidis (2002:4), phase II, the document preprocessing phase, “. . . includes any kind of transformation processes of the original documents retrieved”. In our case, we begin with raw web pages downloaded from the *Botswana Daily News* website, and end up with document vectors, as discussed in Chapter 2, for use in the cluster analysis phase. The complete process is illustrated in Figure 3.2. We developed a number of console based programs to facilitate this phase of our study. Unlike in the data collection phase where we used the C# programming language, in the document preprocessing phase all our programming was done in the C++ programming language. We considered using the C# programming language even in this phase but ultimately chose C++ because we felt more comfortable with it, largely due to our previous exposure to the C language upon which C++ is built. We used the gnu-C++ compiler in conjunction with Dev-C++ IDE and Jen’s File Editor, all of which are freely available on the Internet. The source code for all the programs is attached as Appendix A (Source Code Listings).

3.2.1 Selecting Valid Data Files

As indicated earlier, the data used for this study was downloaded from the *Botswana Daily News* section of the Botswana Government website. The C# program alluded to above simply sends a query (using the HTTP Internet protocol) to the server, to which the server responds by sending the web page for the date specified by the query. The web page so sent may contain the

¹<http://www-users.cs.umn.edu/karypis/cluto/>

actual data we are interested in, or it may simply indicate that there is no data for the specified date, as may happen when the specified date is a day, such as a public holiday, during which the paper did not publish. Thus, once we had downloaded all the data, our first task was to separate “valid” data files from the rest of the downloaded files.

Web pages indicating that no data is available for the specified data always contain the text “No such file or directory”; consequently, we used this phrase to test whether a given data file was valid or not, with the presence of the phrase interpreted as an indication that the file is not valid, and as such should be discarded. Admittedly, this test is rather naïve and could potentially be misleading; for instance, any given file that does not contain the stipulated phrase will be deemed a valid data file, even when the file in question has nothing to do with the *Botswana Daily News* or, by extension, our study! Nevertheless, used with due care, and for the particular data that we are dealing with, the test is quite adequate.

To help facilitate the selection of valid data files from the rest of the data, three programs were developed, namely *isvf* (which was used to test whether a given file is in fact a valid data file), *listvf* (which lists all valid data files in the specified directory), and *copyvf* (which copies all valid data files from one directory to another); these programs, and all the other pre-processing utilities, are described briefly in Table 3.1. The preprocessing step named “Select valid data files” in Figure 3.2 is accomplished by running *copyvf* to copy only the valid data files from the original directory to a different directory. Copying “valid” files to a different working directory achieves our aim of selecting valid data files and also ensures that the original data is left intact.

3.2.2 Removing Headers and Footers

For the sake of uniformity and consistency, web pages on a given web site typically have information that appears on every web page; in the case of the *Botswana Daily News* such information includes, among other things, links to other parts of the Botswana government website. In this study, we refer to such information as a “header” if it comes at the top of the page, or “footer” if it comes at the bottom of the page. Headers and footers need to be removed from all valid data files before any further processing can be undertaken. The program *gethf* was used to compare two files, and extract both the header and the footer, which were then separately saved to specified files. The program works by comparing the two files character by character from the beginning until a mismatch is encountered; all the characters before the mismatch collectively constitute the header. The footer is determined similarly, with processing beginning at the end of the file.

While one may be tempted to simply use any two data files from the database to create a header and footer for use in the whole study, it is important to note that headers and footers may in fact change over time. It is

for this reason that the program *comphf* was written; it can be used to pick two files from, say, January, and another two files from, say December, and compare the January headers and footers with their December counterparts to see if there has been any changes. In our study we created a header / footer pair for each month. Nevertheless, for some months, specifically March and June, we had to create two headers and two footers in order to handle the data accurately.

Once the header and footer files had been created, the next step was to remove the headers and footers from all the valid data files; for this we used the program *stripd*, which takes all the files in a given directory, strips them of the headers and footers (as specified in the appropriate header and footer files), and saves the output in the designated output directory. Sometimes during the course of our experiments, it was necessary to strip individual files; for this we used *stripf* whose target is a file, as opposed to a directory as is the case of *stripd*.

3.2.3 Assigning Story IDs and Removing HTML Codes

With the headers and footers removed, the next step was to assign each individual story some unique ID. The program *annotate* was developed for this purpose. The ID itself is the date the story was published (and this, by the way, is also the name of the data file), and a number that simply indicates the position of the story in the data file; for instance, the tenth story in the data file containing stories that appeared in the *Botswana Daily News* of 23 December 2004 would be assigned the ID 2004122310. Although we did not directly use story headlines (as they appear in the newspaper) in our study, the program *annotate* also assigns corresponding IDs to each story's headline.

At this stage, our data was still in HTML format. A simple and effective way to filter out HTML codes from a data file is to open the file using a web browser, such as Microsoft Internet Explorer, and then use the "save as" option to save the file as a text file; this approach was adopted in the current study. In order to make the process efficient, the data files were first merged into one file; this way the "file/save as" operation would be performed only once for all the data. For merging files, we developed the program *fmerge*, which expects as its input a directory name, and proceeds to merge all the files in the input directory into the single file specified as its output.

Earlier, we indicated that document preprocessing was concerned with getting the data ready for the clustering phase. In this section we discussed some, but not all, of the document preprocessing activities undertaken in this study. In the next two sections, we discuss the rest of the document preprocessing activities, namely, index term selection, and, ultimately the creation of document vectors themselves.

PROGRAM	PURPOSE
<i>grabdata</i>	Queries the <i>Botswana Daily News</i> website for news [extracts] for a specific day and saves the response it receives to a data file.
<i>isvf</i>	Checks if specified file is a valid data file. If specified file contains actual news stories, the program responds ‘yes’, otherwise it responds ‘no’.
<i>listvf</i>	Lists all the valid files in the specified directory, and also gives a count of such files.
<i>copyvf</i>	Used to copy data files from one directory to another; it can also be used to copy a single data from one directory to another, or even to the same directory but under a different file name.
<i>gethf</i>	Extracts header and footer common to specified pair of files, saving them a header and footer file respectively.
<i>comparehf</i>	Compares headers and footers from two different pairs of files.
<i>stripd</i>	Removes header and footer from all files in specified directory, saving stripped files to another directory.
<i>stripf</i>	Same as <i>stripd</i> , but only targets individual files.
<i>annod</i>	Assigns IDs to all stories in all files in a given directory.
<i>annof</i>	Assigns IDs to all stories in a single file.
<i>remhl</i>	Removes story headlines, saving them in the process.
<i>fmerge</i>	Merges all files in specified directory to create one corpus file.
<i>ForCluto</i>	Prepares text corpus for processing with CLUTO’s <i>doc2mat</i> utility.
<i>lineB</i>	Inserts line breaks as appropriate to ensure that each story ID is on a line by itself.

Table 3.1: Document preprocessing utilities developed for use in the current study.

3.3 Index Term Selection

3.3.1 Lexical Analysis

In Chapter 2, we discussed the activities that are typically undertaken to select the terms with which a given body of text is to be indexed. The process begins with tokenisation, which breaks a document into its constituent terms. CLUTO comes with a useful utility, *doc2mat*, which converts raw documents into the data matrices that can be readily used for clustering. This utility, which we used in our study, performs tokenisation in the following manner. First, all non-alphanumeric characters are replaced with spaces. This approach is in keeping with the usual practice in Information Retrieval where punctuation marks are simply ignored (Baeza-Yates and Ribeiro-Neto, 1999). Words are then assumed to be bounded by white spaces, and accordingly extracted.

In the *doc2mat* utility, no special handling of hyphens is provided; indeed, the replacement of all non-alphanumeric characters with spaces will have the effect of eliminating hyphens from the document. Since numbers (alone or mixed with letters) tend to be too vague to be of any value, our approach in this study is to disregard any would be tokens that contain digits. In *doc2mat*, we do this by setting the `skipnumeric` flag. The *doc2mat* also performs case folding, with all letters simply converted to lowercase. A particular advantage of case folding is that it results in vocabulary reduction (Belew, 2000) since words such as ‘the’, ‘The’, and ‘THE’ are all treated as the same word Manning and Schutze (1999); from the purview of cluster analysis, this translates into dimensionality reduction. Naturally, case folding results in some loss of information as, for example, ‘Banks’, with a capital ‘B’ might suggest a proper noun, while ‘banks’ with a lower case suggests the plural form of ‘bank’. Thus, as Belew (2000) points out, case folding comes at the cost of precluding any analysis predicated on proper nouns. However, case folding is fairly standard practice in Information Retrieval (Sahami, 1998).

Baeza-Yates and Ribeiro-Neto (1999) indicate that nouns are sometimes preferred as indexing terms because they are perceived to carry the semantics of the document. However, in our study we did not specifically try to identify nouns with which to index our documents; rather, we followed the more common approach that is term frequency driven. Baeza-Yates and Ribeiro-Neto also suggest the use of noun phrases for indexing purposes; this can of course be extended to include phrases in general. In principle, this does not change the vector-centric view of documents discussed in Chapter 2 as the phrases are simply viewed as compound terms. In our study, we only use terms, and not phrases, for indexing.

3.3.2 ‘Fluff’ Word Removal and Stemming

The *doc2mat* utility comes with an in-built stop list that one can use directly. However, in our study we used the stop list designed by Salton and his colleagues for use with their SMART (System for the Manipulation and Retrieval of Text) information retrieval system (Salton and McGill, 1983). It is perhaps the single most frequently cited stop word list within the information retrieval literature. We have included a copy of the SMART stop list in Appendix B. Stemming, which removes grammatical affixes, thereby converting a set of related words to a common root form (Korfhage, 1997), is used quite frequently in Information Retrieval. In this study, stemming is accomplished via Porter’s stemming algorithm, which is available as part of the *doc2mat* utility.

We indicated in Chapter 2 that Zipf’s Law can be used to exclude unwanted words from the indexing language. While some researchers (e.g. Sahami (1998) do use Zipf’s Law for this purpose, others (e.g. Steinbach *et al.* (2000) do not use it at all. In our experiments we elected not to apply Zipf’s Law. We believe that the use of a stop list is sufficient in getting rid of fluff words, particularly given the size (571 words) of the stop word list that we use. In any case, as Van Rijsbergen (1979) points out, the cut-off points used with Zipf’s Law are arbitrarily chosen, which in effect lessens the appeal of the law.

3.4 Creating Document Vectors

In any cluster analysis study, the data will ultimately have to be represented in a format that the clustering algorithm requires. The data matrices that result from the use of the vector space model tend to be sparse i.e. they have relatively few non-zero entries. Figure 3.3 is an example of a sparse document-by-term matrix in which each column represents a unique term within the corpus under consideration, with each row representing an individual document. This view is slightly different from the term-by-document view depicted in Table 2.1; the reason for this is that the clustering algorithm expects each object - in our case, each document - to be in a row by itself. In this case, the terms being used as column headings constitute the dimensions of the vector space.

$$\begin{pmatrix} 1 & 0 & 0 & 3 \\ 0 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 4 & 0 & 0 & 0 \end{pmatrix}$$

Figure 3.3: A sample sparse matrix

Storing a sparse matrix such as the one in Figure 3.3 consumes a lot of space unnecessarily. In the case of high dimensional data, such as that being dealt with in this study, the amount of space wasted can be quite significant, growing exponentially with the number of documents. A number of efficient, space saving schemes have been devised for storing sparse matrices. The schemes also help in speeding up computations involving the matrices, an important consideration in computation intensive applications such as clustering. In the scheme adopted in CLUTO, the sparse matrix of Figure 3.3 would be stored as follows:

```

4 4 5
1 1 4 3
4 2
2 1
1 4

```

Figure 3.4: CLUTO sparse matrix representation

The first row of Figure 3.4 can be considered meta-data as it informs the clustering software that the data file contains 4 rows (the first 4), 4 columns (the second 4), and 5 non-zero entries. The numbers in the other rows are in pairs, with the first part indicating the column number, and the second part being the actual value stored at that place. For instance, the second row informs the clustering software that the data value in row 1 column 1 is 1, while that in row 1 column 4 is 3. Similarly, the third row indicates that the data value in column 4 of row 2 is 2. Note that because of the meta-data row in the sparse matrix representation, row n in the matrix carries data from row $n-1$ of the original data matrix.

The *doc2mat* utility directly converts raw documents into the sparse matrix representation expected by the clustering software. It, however, expects the input data to be in a specific format. Specifically, *doc2mat* expects that each story be stored on a single line, with the story IDs, where available, coming at the beginning of each line. We used the program *ForCluto*, whose functionality is briefly summarised in Table 3.1. One program that is also alluded to in Table 3.1 that we have not made mention of so far is *lineB*. We used the program after HTML filtering to ensure that each story ID was on a line by itself; this made processing with *ForCluto* rather straightforward. We will now briefly discuss the criterion functions in CLUTO, after which we will be ready to present the actual parameters we used with the clustering algorithms.

3.5 Criterion Functions in CLUTO

Clustering seeks to group similar objects. Similarity (dissimilarity) measures, such as the cosine coefficient, are used to quantify the similarity (dissimilarity) between objects. However, even when the number of clusters is kept constant, a collection of objects can be clustered in many different ways. The challenge, then, lies in selecting the best clustering solution among those “discovered” by the clustering algorithm. To address this issue, clustering algorithms typically have criterion functions that they seek to optimise; the clustering solution that out performs all others at maximising the criterion function is the one that is kept. For illustrative purposes, CLUTO’s clustering criterion functions are listed in Table 3.2. In CLUTO, the number of different clustering solutions that must be computed, from which the best is then selected as per the specified criterion function, is set using the `ntrials` (i.e. number of trials) parameter. To illustrate, if in running one of CLUTO’s clustering algorithms we set the `ntrials` parameter to 10 and the criterion function parameter to I_1 , then CLUTO will compute 10 different clustering solutions (all with the same number of clusters k) from which the one that is the best at maximising

$$\sum_{i=1}^k \frac{1}{n_i} \left(\sum_{v,u \in S_i} sim(v,u) \right)$$

will be elected as the final solution.

In the table k is the total number of clusters, S is the total objects to be clustered, S_i is the set of objects assigned to the i th cluster, n_i is the number of objects in the i th cluster, v and u represent two objects, and $sim(v,u)$ is the similarity between two objects.

After performing a number of experiments comparing the different criterion functions implemented in CLUTO, Zhao and Karypis (2002) concluded that for direct k -way clustering, the criterion functions I_2 and H_2 always produced the best clustering solutions, while for k -way clustering via repeated bisections the best overall clustering solutions are obtained when using criterion function H_2 . Consequently, in our experiments we used criterion functions I_2 and H_2 for direct k -way clustering and k -way clustering via repeated bisections, respectively. Zhao and Karypis (2002) characterise criterion functions as *internal*, *external*, and *hybrid*. Internal criterion functions focus on the similarity of documents with each cluster without regard to documents in other clusters (internal cohesion), while external criterion functions are concerned with how the various clusters are different from each other (external isolation); hybrid functions combine two or more other criterion functions, forming one new criterion function. Criterion function I_2 is internal, and seeks to maximise the pairwise similarity between documents in each cluster; H_2 , on the other hand, is hybrid as it combines an internal function with an external one. For agglomerative clustering, we experimented with all the criterion functions available

Criterion Function	Optimisation Function
I_1	maximise $\sum_{i=1}^k \frac{1}{n_i} \left(\sum_{v,u \in S_i} sim(v,u) \right)$
I_2	maximise $\sum_{i=1}^k \sqrt{\sum_{v,u \in S_i} sim(v,u)}$
E_1	minimise $\sum_{i=1}^k n_i \frac{\sum_{v \in S_i, u \in S} sim(v,u)}{\sum_{v,u \in S_i} sim(v,u)}$
G_1	minimise $\sum_{i=1}^k \frac{\sum_{v \in S_i, u \in S} sim(v,u)}{\sum_{v,u \in S_i} sim(v,u)}$
G'_1	minimise $\sum_{i=1}^k n_i^2 \frac{\sum_{v \in S_i, u \in S} sim(v,u)}{\sum_{v,u \in S_i} sim(v,u)}$
H_1	maximise $\frac{I_1}{E_1}$
H_2	maximise $\frac{I_2}{E_1}$

Table 3.2: The mathematical definition of CLUTO's clustering criterion functions

[Source: Karypis, 2003:10]

in CLUTO, namely, UPGMA, single linkage (slink), weighted single linkage (wslink), complete linkage (clink), and weighted complete linkage (wlink); we discussed these criterion functions in Chapter 2.

Zhao and Karypis (2002) explain that CLUTO optimises criterion functions in two stages. First, an initial clustering is calculated by randomly selecting k documents (where k is the user specified number of clusters) as the seeds for k clusters. The rest of the documents in the collection are then assigned to the various clusters based on their similarity with the cluster seeds. In the refinement stage, each document is considered individually; if moving the document to one of the other clusters results in an improvement in the overall value of the clustering criterion, the document is moved, otherwise it remains where it is. The refinement stage terminates when no improvement can be achieved by moving any of the documents. It can thus be seen that, as Zhao and Karypis (2002) also point out, the k -way clustering algorithm in CLUTO works slightly differently from the traditional k -means clustering algorithm we presented in Chapter 2; this is especially obvious when we compare the refinement process discussed in this paragraph, with stage 5 of Table 2.2.

3.6 Document Clustering

The final phase of the KDT process model around which we modelled our study is text mining (cluster analysis in our case). This involves running the clustering algorithms, and interpreting their output; we defer analysis and interpretation to Chapter 4, and in this section only discuss our cluster analysis approach. We performed cluster analysis at two levels. Firstly, we analysed the entire corpus to determine the pattern of news coverage over the whole study period. At this level, we experimented with all three clustering techniques, namely, direct k -way clustering, k -way clustering through repeated bisections, and agglomerative clustering. We then selected the clustering technique that, in our opinion, had produced the best results, and applied it at the second level of analysis, namely on a quarterly basis. Here, we split our corpus into four smaller corpora, containing data for the first, second, third and fourth quarters of the year 2004. Performing cluster analysis at this level allowed us to develop an appreciation of the variation in news coverage over the year.

In our experiments, we used gCLUTO, a graphical interface to CLUTO. A particular attraction of gCLUTO, apart from ease of use, is that it can export its output to HTML format, which output can then be easily read into a word processor for incorporation into a report, such as the current one. In Table 3.3 we show the options we selected when running direct k -way clustering; the last column of the table shows the equivalent options when running CLUTO from the command line. We used the same parameter setting for the other two clustering schemes, only changing the setting for the criterion function, as appropriate.

gCLUTO Option	Value Specified	Equivalent CLUTO Option
Similarity Function	Cosine	<i>-sim = string</i>
Row Model	None	<i>-rowmodel = string</i>
Column Model	Inverse Document Frequency	<i>-colmodel = string</i>
Number of Trials	10	<i>-ntrials = int</i>
Criterion Function	I_2	<i>crfun = string</i>
Number of Iterations	10	<i>-niter = int</i>

Table 3.3: gCLUTO clustering options for direct k-way clustering

3.7 Summary

In this chapter we discussed the work actually undertaken in this study. We based our methodology upon the three-phased KDT process suggested by Karanikas and Theodoulidis (2002), which in turn is really just a simplification of more elaborate KDD process models. Our approach can thus be divided into three phases, namely, data collection, document preprocessing, and text clustering. The data collection phase involved downloading the requisite data from the *Botswana Daily News* website, for which we developed a small C# program. We used the C++ programming language to develop a number of console based programs to facilitate the document preprocessing phase. Document preprocessing itself was undertaken in phases. First we separated valid data files from the rest of the data. We then proceeded to remove headers and footers from all the valid data files, and assigned each story an appropriate ID before merging the files into one. To filter HTML codes from the data, we simply opened the consolidated text corpus using Internet Explorer, and then used the ‘Save As’ option to save it as a ‘clean’ text file. We used a utility called *doc2mat*, which comes with CLUTO, to perform the functions of tokenising, stop word removal, and stemming. The utility also converted our text corpus into the matrix format that CLUTO expects. The actual cluster analysis itself was done at two levels, namely, for the whole year, and then for each quarter. In the next chapter, we discuss our findings.

Chapter 4

Data Analysis and Interpretation

4.1 Introduction

In this study we undertake a cluster-analytic exploration of news stories published in the *Botswana Daily News* during the period January through December 2004. We presented our methodology in Chapter 3, and we devote the current chapter to analysis and interpretation of the data. We begin the chapter with a brief description of the data used in this study. After discussing the type of output produced by the software used in the experiments, we present the results from the clustering experiments. We experimented with three different clustering algorithms, namely, a direct k-way clustering algorithm, a bisecting k-way clustering algorithm, and an agglomerative clustering algorithm; all three algorithms are implemented as part of the CLUTO package that we used in our study. We discuss the results from these three clustering approaches individually, before presenting a comparative analysis that we undertook to select the clustering solution that, in our opinion, is the best amongst the three. A particular challenge in exploratory cluster analysis is the determination of the target number of clusters. In confirmatory cluster analysis, the figure will typically come from the theory or model being tested. Our approach is to choose a ‘reasonable’ number of ‘reasonably’ sized clusters. Admittedly, this is inherently subjective; it is however, the only way. We also undertook a quarterly analysis of the data, which allowed us to see variations in news coverage over the course of the year.

4.2 Data Description

The data used in our study was downloaded from the website of the *Botswana Daily News*. The program that we wrote for this purpose simply requested web pages for each calendar day in 2004, which data were then saved to disk.

Month	Story Days		No. of Stories	
	Expected	Actual	For Month	Avg./Day
Jan	19	20	290	15
Feb	20	20	377	19
Mar	23	23	415	18
Apr	20	20	344	17
May	20	22	376	17
Jun	22	22	355	16
Jul	19	19	362	19
Aug	22	22	381	17
Sep	21	21	427	20
Oct	20	20	458	23
Nov	22	22	469	21
Dec	22	3	78	26
TOTAL	250	234	4332	19

Table 4.1: Summary corpus statistics

Some of the web pages contained actual stories for the specified day, while others merely indicated that the requested data was not available. We refer to the latter as ‘invalid data files’, and do not use them in our study. Thus, while we initially started off with 366 data files, corresponding to the number of calendar days in the year 2004 (a leap year), we eventually remained with 233 valid data files to work with. Not surprisingly, the majority of the ‘invalid’ data files correspond to days during which the *Botswana Daily News* does not publish; these ‘non-publishing’ days are weekends, and public holidays. We provide a complete list of the 2004 Botswana public holidays in Appendix C (2004 *Botswana Daily News* non-publishing days). There were also a number of instances of data being unavailable for official publishing days. For January 2004, data was not available for the 5th and the 16th, both of which were publishing days, but was strangely available for the 11th, a *non-publishing* day. In May, data was available for two *non-publishing* days, namely, the 2nd and the 16th, both of which were Sundays. While there were twenty-two publishing days in December 2004, data was only available for three days, namely Wednesday December 1 through Friday December 3. For all the other months, data was available as expected, as shown in Table 4.1. The absence of data in January and December might have been due to the fact that in December / January people generally tend to want to go on leave for Christmas, which might have impacted negatively on the data updating schedule. The availability of data for non-publishing days is more difficult to explain. Nevertheless, browsing through the appropriate data files revealed that the stories themselves were genuine, and we have therefore included them in our analysis.

As can be seen from Table 4.1, our corpus consisted of a total of 4332

stories that appeared in the *Botswana Daily News* over a period of 233 days. On average, about 19 stories were published per day i.e. per newspaper edition. We did not source the raw data directly from the published newspaper, but from a website that publishes extracts from the printed paper. It is, thus, possible that not all stories that appeared in the printed paper appear on the website; this is obviously true for December, 2004, but may also be true even for days that appear to have had all their stories uploaded. Additionally, the stories on the website may very well be, at least in some cases, extracts from - rather than full versions of - those that appeared in the printed paper.

Initially, the number of tokens in our corpus stood at 33 407. We used two dimensionality reduction techniques, namely, stop word removal, and stemming. On its own, stop word removal reduced the number of tokens to 32 816, while on its own stemming reduced the number of tokens to 23 303. Thus, removing stop words reduced the corpus by only about 2%, while, in contrast, stemming reduced it by roughly 30%. In our study, we used the SMART stop list (see Appendix B) to remove fluff words, and then applied the Porter stemmer; this reduced the number of dimensions from 33 407 to 23 036, which gave roughly the same number of dimensions as when stemming is used on its own!

4.3 Understanding the Output from gCLUTO

In our experiments, we used gCLUTO, the graphical user interface to CLUTO, and in this section we briefly explain its output. gCLUTO produces two types of output, the first of which is illustrated in Table 4.2. The meanings of the column headings ‘cluster’ and ‘size’ are as one would expect, namely, cluster ID and the total number of documents that have been assigned to the given cluster, respectively. The other four columns, however, deserve special mention. ISim stands for ‘internal **s**imilarity’, and refers to the average similarity between the objects in a given cluster. ISdev is the standard deviation of the internal similarities. Similarly, ESim stands for ‘external **s**imilarity’, and quantifies the external similarities (i.e. the average similarity of the objects in the specific cluster to the rest of the objects), while ESdev is the standard deviation of the external similarities. In essence, ISim and ESim are concerned with internal cohesion and external isolation, respectively. Thus, looking, for instance, at the five cluster solution shown in Table 4.2, we see that cluster 0 has a much higher ISim than cluster 4, suggesting that the former is more internally cohesive than the latter; we would expect, therefore, that the stories in cluster 0 be more related, than those in cluster 4. Nevertheless, we notice that cluster 4 does have a clearly smaller ISdev value than cluster 0; one suspects that that there may be a small number of documents in cluster 0 that are markedly different from the rest, causing a spike in the ISDev figure, while cluster 4 consists of a large number of objects that are fairly uniformly dispersed around a “wide” cluster.

Cluster	Size	ISim	ISdev	ESim	ESdev
0	608	0.097	0.029	0.021	0.005
1	445	0.066	0.018	0.015	0.005
2	902	0.038	0.012	0.019	0.006
3	1239	0.035	0.010	0.020	0.006
4	1134	0.033	0.010	0.019	0.007

Table 4.2: Sample gCLUTO output, showing ISim and ESIm, and their deviations

The second type of information that gCLUTO produces is shown in Table 4.3. Here, in addition to the ISim and ESIm parameters that we have already discussed; gCLUTO provides for each cluster two other parameters called ‘descriptive’ and ‘discriminating’. The difference between the two is rather subtle; the following explanation comes directly from the CLUTO manual:

The set of descriptive features is determined by selecting the columns that contribute the most to the average similarity between the objects of each cluster. On the other hand, the set of discriminating features is determined by selecting the columns that are more prevalent in the cluster compared to the rest of the objects. In general, there will be a large overlap between the descriptive and discriminating features. (Karypis, 2003:16)

The overlap that Karypis alludes to is obvious in Table 4.3. Nevertheless, in our analysis we tend to focus more on the descriptive features than on the discriminating features.

Cluster 0:- Size=608;ISim=0.097;Esim=0.021								
Descriptive:	parti	16.7%	bdp	11.5%	elect	7.0%	candid	4.9%
Discriminating:	parti	14.0%	bdp	10.1%	elect	5.3%	candid	4.2%
Cluster 1:- Size=445; ISim=0.066; ESIm=0.015								
Descriptive:	team	11.7%	game	8.4%	sport	5.0%	cup	2.8%
Discriminating:	team	8.4%	game	6.2%	sport	3.5%	cup	4.2%
Cluster 2:- Size=902; ISim=0.038; ESIm=0.019								
Descriptive:	hiv	4.6%	aid	4.1%	school	2.8%	music	2.5%
Discriminating:	hiv	4.4%	aid	3.9%	music	3.0%	church	2.8%
Cluster 3:- Size=1239; ISim=0.035; ESIm=0.020								
Descriptive:	develop	1.5%	sadc	1.5%	product	1.3%	bank	1.1%
Discriminating:	parti	3.6%	bdp	2.6%	sadc	1.9%	elect	1.6%
Cluster 4:- Size=1134; ISim=0.033; ESIm=0.019								
Descriptive:	polic	4.1%	land	2.8%	resid	2.3%	court	2.1%
Discriminating:	polic	4.4%	land	2.8%	parti	2.8%	court	2.1%

Table 4.3: Sample gCLUTO output, showing descriptive and discriminating features.

4.4 Side Effects of Stemming

Before we proceed with our analysis, it is important that we take note of an interesting, and potentially misleading, side effect of stemming. A stemmer, such as the one we used in this study, will frequently produce what may be called non-words, such as the entries ‘parti’ and ‘polic’ in Table 4.3. The stem ‘parti’ may have come from the word ‘parties’, as in ‘political parties’, or from the word ‘participation’, resulting in confusion as to the meaning of a cluster whose most descriptive dimension is ‘part’. One could then use other descriptive / discriminating features listed for the cluster to infer the meaning of the stem in question. Unfortunately, this will not always be straightforward. For instance, in one cluster, ‘parti’ might occur with entries that suggest that it refers to ‘political parties’, while in another cluster in the same clustering solution, it might occur with other entries that suggest it refers to ‘participation’, perhaps in developmental, or even regional, issues. We should also recall that in the vector space model used in this study, homonyms are collapsed into a single dimension, while synonyms become different dimensions within the document space. Thus, the word ‘bank’, for example, might refer to a particular commercial bank, commercial banks in general, a river bank (where the developments are taking place), or perhaps the act of banking on the survival of SADC! It is important that we keep these issues in mind as we analyse and interpret the different clustering solutions.

4.5 Direct K-Way Clustering

We ran the direct k-way clustering algorithm a number of times, creating 21 different clustering solutions whose cluster count ranged from 5 to 25. In the 5 cluster solution (see Table 4.3), clusters 0 and 1 are self describing. Cluster 0, whose descriptive and discriminating features are ‘parti’, ‘bdp’, ‘elect’, and ‘candid’ is about political issues, with emphasis on political party activities. Cluster 1, on the other hand, brings together stories about ‘Sports’. Clusters 2, 3 and 4 are a little more difficult to decipher. Cluster 2 seems to be about HIV/AIDS issues, but also makes reference to ‘school’, ‘music’ and ‘church’. Clusters 3 and 4 both make references to ‘parti’, which brings in an element of confusion as we also have ‘parti’ in Cluster 0.

Intuitively, one expects that as the number of clusters increases, the resulting clusters should become more internally cohesive and externally isolated, making their interpretation more straightforward. Indeed, looking at the 10-cluster solution, this appears to be the case. In the 5-cluster solution, ‘land’, ‘polic’ and ‘court’ were bundled together; now Cluster 0 is made up of stories dealing with land issues (plots are allocated by land boards), while ‘polic’ and ‘court’ which intuitively go together, are in cluster 5. The two new clusters both have a higher ISim value than the cluster they were (presumably) origi-

nally part of. Cluster 1 is about political issues, and is analogous to Cluster 0 in the 5-cluster solution; note that the cluster is now smaller than it was before, and has ISim and ESim scores that indicate that it now is ‘tighter’ than in the 5-cluster solution.

Earlier, when, comparing the ISdev of Cluster 0 and 1 in the 5-cluster solution (see Table 4.3), we made the suggestion that there may be a small number of documents in cluster 0 that are markedly different from the rest; the fact that the ‘political matters’ cluster has been retained in the 10-cluster solution, albeit now with a smaller number of members, and a higher ISim value would appear to confirm our suspicion. In the 5-cluster solution, music was bundled together with HIV/AIDS, school and church in cluster 2; in the 10-cluster solution, music is in a cluster of its own, namely cluster 2; ‘school’ also gets its own cluster, cluster 4. The ‘sports’ cluster, which was cluster 1 in the 5-cluster solutions, is retained, now as cluster 3 in the 10-cluster solution; the cluster membership has fallen slightly from 445 to 431, with a concomitant rise in the internal cohesion as indicated by the higher ISim value. Cluster 7 is about regional bodies (SADC, and ACP), while cluster 9 is geared towards business and financial matters; both clusters would appear to have been subsumed in cluster 3 in the 5-cluster solution. Our complete interpretation of the 10-cluster direct k-way clustering solution is presented in Table 4.5, while the 15 and 20 cluster solutions are presented in Tables 4.6 and 4.7 respectively.

A particular challenge in exploratory cluster analysis is the determination of the number of clusters. In confirmatory cluster analysis, the figure will typically come from the theory or model being tested. Our approach is to choose a ‘reasonable’ number of ‘reasonably’ sized clusters. Admittedly, this involves a fair amount of arbitrariness. Even so, less than ten clusters would appear to be too few, while more than twenty begins to feel like too many. For instance, as Table 4.7 shows, at 20 clusters it is rather difficult to meaningfully label quite a few of the identified clusters. Ultimately, we settled for 15 clusters. Table 4.4 shows the direct k-way 15 cluster solution produced by gCLUTO, while our interpretation of the various clusters is summarised in Table 4.6.

Direct k-way clustering 15 cluster solution

Cluster 0; Size: 96; ISim: 0.168; ESIm: 0.021									
Descriptive:	land	42.20%	plot	9.30%	alloc	9.00%	board	3.90%	
Discriminating:	land	28.80%	plot	6.40%	alloc	6.00%	board	2.20%	
Cluster 1; Size: 427; ISim: 0.122; ESIm: 0.023									
Descriptive:	parti	18.90%	bdp	16.30%	candid	5.00%	opposit	4.40%	
Discriminating:	parti	14.10%	bdp	13.70%	candid	3.70%	opposit	3.50%	
Cluster 2; Size: 278; ISim: 0.091; ESIm: 0.015									
Descriptive:	team	14.00%	game	9.20%	zebra	4.50%	cup	3.90%	
Discriminating:	team	8.90%	game	5.90%	zebra	3.20%	cup	2.70%	
Cluster 3; Size: 161; ISim: 0.099; ESIm: 0.025									
Descriptive:	elect	21.30%	Iec	5.40%	vote	4.90%	voter	4.80%	
Discriminating:	elect	17.00%	Iec	5.50%	voter	4.30%	vote	3.00%	
Cluster 4; Size: 240; ISim: 0.081; ESIm: 0.016									
Descriptive:	music	14.10%	album	11.60%	song	8.50%	artist	7.20%	
Discriminating:	music	10.50%	album	8.90%	song	6.40%	artist	5.40%	
Cluster 5; Size: 179; ISim: 0.079; ESIm: 0.018									
Descriptive:	sport	19.00%	athlet	9.00%	olymp	4.00%	championship	2.50%	
Discriminating:	sport	14.60%	athlet	7.40%	olymp	3.30%	championship	1.80%	
Cluster 6; Size: 188; ISim: 0.076; ESIm: 0.016									
Descriptive:	court	22.40%	magistr	5.10%	accus	3.40%	case	2.70%	
Discriminating:	court	16.30%	magistr	4.00%	accus	2.20%	applic	1.50%	
Cluster 7; Size: 233; ISim: 0.076; ESIm: 0.017									
Descriptive:	polic	29.80%	crime	4.50%	superintend	4.00%	suspect	2.80%	
Discriminating:	polic	22.00%	superintend	3.20%	crime	3.10%	suspect	2.00%	

Direct k-way clustering 15 cluster solution

Cluster 8; Size: 240; ISim: 0.069; ESIm: 0.022								
Descriptive:	school	20.80%	student	14.90%	teacher	6.80%	educ	5.40%
Discriminating:	school	17.80%	student	14.60%	teacher	6.20%	educ	3.70%
Cluster 9; Size: 467; ISim: 0.057; ESIm: 0.021								
Descriptive:	hiv	9.00%	Aid	8.10%	women	5.70%	church	4.80%
Discriminating:	hiv	8.60%	Aid 7.50%	church	5.20%	women 5.10%		
Cluster 10; Size: 343; ISim: 0.057; ESIm: 0.022								
Descriptive:	sadc	10.70%	region	4.00%	africa	2.50%	acp	2.50%
Discriminating:	sadc	12.60%	region	3.70%	acp	3.10%	parti	1.80%
Cluster 11; Size: 405; ISim: 0.053; ESIm: 0.022								
Descriptive:	water	7.20%	road	6.10%	resid	5.90%	villag	4.30%
Discriminating:	water	7.30%	road	6.10%	resid	5.40%	villag	3.70%
Cluster 12; Size: 280; ISim: 0.050; ESIm: 0.019								
Descriptive:	farmer	6.60%	tourism	5.90%	agricultur	5.20%	anim	2.10%
Discriminating:	farmer	6.60%	tourism	5.90%	agricultur	4.70%	anim	2.00%
Cluster 13; Size: 358; ISim: 0.051; ESIm: 0.022								
Descriptive:	bank	7.30%	cent	3.50%	busi	3.20%	privatis	2.80%
Discriminating:	bank	8.20%	privatis	3.30%	cent	3.10%	busi	2.50%
Cluster 14; Size: 433; ISim: 0.041; ESIm: 0.023								
Descriptive:	offic	2.40%	servic	2.30%	public	2.00%	employe	2.00%
Discriminating:	employe	2.60%	parti	1.90%	bdp	1.80%	servic	1.60%

Table 4.4: Direct k-way 15 cluster solution

CID	Cluster Name
0	Land Allocation and Related Matters
1	Political Parties and Related Matters
2	Music
3	National Football Team
4	Schooling and Education
5	Policing and Crime Prevention
6	HIV/AIDS
7	Regional Bodies
8	Inconclusive - Developments?
9	Inconclusive

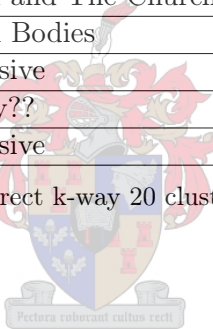
Table 4.5: Direct k-way 10 cluster taxonomy

CID	Cluster Name
0	Land Allocation and Related Issues
1	Political Parties and Related Matters
2	National Football Team
3	National Elections and Related Matters
4	Music
5	Sports
6	Legal Matters
7	Policing and Crime Prevention
8	Schooling and Education
9	HIV / AIDS
10	Regional Bodies
11	Developments
12	Agriculture and Tourism
13	Financial and Business News
14	The World of Work

Table 4.6: Direct k-way 15 cluster taxonomy

CID	Cluster Name
0	Land Allocation and Related Matters
1	Gender Issues
2	Political Parties and Related Matters
3	National Football Team
4	Music
5	Farming and Agriculture
6	Road and Traffic Safety
7	Policing and Crime Prevention
8	Sports
9	National Elections and Related Matters
10	Schooling and Education
11	Floods and Other Disasters ??
12	Legal Matters
13	Wildlife and Tourism
14	HIV/AIDS and Other Health Matters
15	Children and The Church??
16	Regional Bodies
17	Inconclusive
18	Economy??
19	Inconclusive

Table 4.7: Direct k-way 20 cluster taxonomy



4.6 K-Way Clustering Through Repeated Bisections

In this section we present our interpretation of the 15-cluster solution using the repeated bisection algorithm. Our choice of the 15-cluster solution is informed by our desire to compare this solution with the 15-cluster solution from the direct k-way algorithm we chose earlier. The 15 cluster bisecting k-way clustering solution is depicted in Table E.4. Assigned to cluster 0 are stories about ‘Political Parties and Related Matters’. Cluster 1 is concerned specifically with the ‘National Soccer Team’, affectionately known as the ‘Zebras’, while cluster 2 contains stories about ‘Sports’ in general. Stories relating to the ‘National Elections’ are in cluster 3. The subject of ‘Policing and crime prevention’ is addressed by stories in cluster 4, ‘Music’ by those in cluster 5, and ‘Education’ by those in cluster 6. Cluster 7 stories are concerned with ‘Legal Matters’, while those in cluster 8 are generally about ‘Land Allocation and Related Issues’. Stories in cluster 9 would appear to be about HIV/AIDS, with the references to the church and women probably related to the important role the two play in the fight against the scourge. Cluster 10 is about ‘Regional Bodies’. Cluster 11 and 12 are not so straightforward to label. Cluster 11 seems to be about ‘Residential (i.e. domestic) water supply in villages’, and cluster 12 about ‘Developments Projects undertaken by / in Local Governments’. Stories in cluster 13 are about ‘Financial and Business News’, while cluster 14 brings together stories about ‘Agriculture and Tourism’.

In the next section we look at the results from the agglomerative algorithm. We return to bisecting k-way clustering in a later section, where we compare it with the direct k-way clustering algorithm.

15 cluster bisecting k-way clustering solution

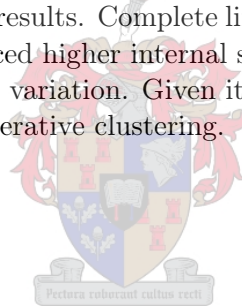
Cluster 0 - : Size: 464; ISim: 0.115; ESIm: 0.023								
Descriptive:	Parti	18.8%	bdp	15.0%	candid	5.2%	opposit	4.2%
Discriminating:	Parti	14.6%	bdp	12.8%	candid	4.0%	opposit	3.4%
Cluster 1 :- Size: 288; ISim: 0.084; ESIm: 0.015								
Descriptive:	Team	14.0%	game	9.3%	zebra	4.6%	cup	4.1%
Discriminating:	Team	9.0%	game	6.1%	zebra	3.4%	cup	3.0%
Cluster 2 :- Size: 194; ISim: 0.075; ESIm: 0.018								
Descriptive:	Sport	18.2%	athlet	8.0%	olymp	3.9%	championship	2.8%
Discriminating:	Sport	14.4%	athlet	6.8%	olymp	3.2%	championship	2.2%
Cluster 3 :- Size: 207; ISim: 0.078; ESIm: 0.026								
Descriptive:	Elect	17.5%	parliament	3.9%	iec	3.8%	voter	3.3%
Discriminating:	Elect	15.9%	iec	4.5%	voter	3.3%	parliament	3.2%
Cluster 4 :- Size: 258; ISim: 0.067; ESIm: 0.016								
Descriptive:	Polic	27.3%	crime	4.4%	superintend	3.9%	suspect	3.0%
Discriminating:	Polic	20.7%	superintend	3.2%	crime	3.1%	suspect	2.3%
Cluster 5 :- Size: 295; ISim: 0.064; ESIm: 0.016								
Descriptive:	Music	11.6%	album	9.9%	song	7.2%	artist	6.2%
Discriminating:	Music	9.2%	album	8.1%	song	5.8%	artist	5.0%
Cluster 6 :- Size: 255; ISim: 0.066; ESIm: 0.022								
Descriptive:	School	20.7%	student	12.9%	teacher	7.2%	educ	5.2%
Discriminating:	School	18.4%	student	12.7%	teacher	7.0%	educ	3.7%
Cluster 7 :- Size: 237; ISim: 0.061; ESIm: 0.017								
Descriptive:	Court	19.4%	magistr	3.8%	accus	2.7%	case	2.4%
Discriminating:	Court	15.8%	magistr	3.3%	basarwa	1.9%	accus	1.8%

15 cluster bisecting k-way clustering solution								
Cluster 8 :- Size: 246; ISim: 0.064; ESim: 0.022								
Descriptive:	Land	19.2%	alloc	4.1%	plot	3.5%	board	3.2%
Discriminating:	Land	19.4%	alloc	4.0%	plot	3.4%	board	2.8%
Cluster 9 :- Size: 408; ISim: 0.062; ESim: 0.022								
Descriptive:	Hiv	10.4%	aid	9.2%	church	5.5%	women	4.7%
Discriminating:	Hiv	9.6%	aid	8.3%	church	5.8%	women	3.7%
Cluster 10 :- Size: 303; ISim: 0.063; ESim: 0.023								
Descriptive:	Sadc	11.5%	region	3.9%	acp	2.9%	develop	2.0%
Discriminating:	Sadc	12.9%	acp	3.4%	region	3.4%	parti	1.9%
Cluster 11 :- Size: 297; ISim: 0.061; ESim: 0.023								
Descriptive:	water	11.1%	resid	10.2%	villag	5.4%	kgotla	2.6%
Discriminating:	water	11.2%	resid	9.9%	villag	4.6%	kgotla	2.7%
Cluster 12 :- Size: 246; ISim: 0.055; ESim: 0.022								
Descriptive:	Road	16.2%	council	5.8%	councillor	5.6%	project	3.0%
Discriminating:	Road	17.2%	councillor	5.2%	council	3.9%	construct	2.0%
Cluster 13 :- Size: 326; ISim: 0.053; ESim: 0.021								
Descriptive:	Bank	8.0%	cent	4.4%	compani	3.0%	privatis	2.8%
Discriminating:	Bank	8.5%	cent	3.9%	privatis	3.0%	compani 2.2%	
Cluster 14 :- Size: 304; ISim: 0.052; ESim: 0.021								
Descriptive:	agricultur	4.8%	product	4.8%	farmer	4.7%	tourism	4.5%
Discriminating:	farmer	5.1%	agricultur	5.0%	tourism	4.9%	product	4.3%

Table 4.8: 15 cluster bisecting k-way clustering solution

4.7 Agglomerative Clustering

In our study, we experimented with all the agglomerative clustering functions provided in CLUTO, and the results were rather disappointing. Table 4.9 shows a 15-cluster solution using the complete linkage criterion function. The table demonstrates a common problem we encountered with agglomerative clustering; clusters that are either too small to be useful, or too large to be meaningful. We chose to display the 15-cluster solution using weighted complete linkage to be consistent with the 15-cluster solution we settled upon when using direct k-way clustering, but also because compared to other agglomerative clustering solutions, it is one of the more appealing. At 15 clusters, single linkage and weighted complete linkage produced identical results, namely, one large cluster of 4311 members, one cluster with 3 members, one cluster with 2 members, and the rest of the clusters have one member each. UPMGA fared better with one cluster of 2964 members, one with 452 members, one with 410 members, one with 138 members, and the rest with membership ranging between 1 and 14. Thus, by comparison the complete linkage and weighted complete linkage criterion functions produced by far the most useful results of the agglomerative clustering results. Complete linkage, the results of which are depicted in Table 4.9, produced higher internal similarities and lower external similarities than its weighted variation. Given its poor showing, we elected to discard the results of agglomerative clustering.



15 cluster complete linkage agglomerative clustering solution

Cluster 0 :- Size: 360; ISim: 0.124; ESIm: 0.025						
Descriptive:	Parti	Bdp	15.30%	candid	bnf	3.70%
Discriminating:	Parti	Bdp	12.50%	candid	bnf	3.10%
Cluster 1 :- Size: 48; ISim: 0.120; ESIm: 0.028						
Descriptive:	parliament	Mp	9.60%	presid	molomo	2.50%
Discriminating:	Mp	parliament	8.50%	molomo	assembl	1.70%
Cluster 2 :- Size: 108; ISim: 0.068; ESIm: 0.026						
Descriptive:	univers	Resid	3.50%	moatlhodi	immigr	2.70%
Discriminating:	univers	moatlhodi	3.50%	immigr	resid	2.30%
Cluster 3 :- Size: 28; ISim: 0.144; ESIm: 0.026						
Descriptive:	sebetela	Minist	3.20%	telecommun	technolog	2.80%
Discriminating:	sebetela	telecommun	2.20%	technolog	boyc	1.60%
Cluster 4 :- Size: 1434; ISim: 0.024; ESIm: 0.022						
Descriptive:	Hiv	Aid	1.50%	school	road	1.10%
Discriminating:	Parti	Bdp	4.70%	elect	hiv	2.20%
Cluster 5 :- Size: 6; ISim: 0.268; ESIm: 0.011						
Descriptive:	Regret	Erron	11.40%	omiss	inconveni	6.30%
Discriminating:	Regret	Erron	6.60%	omiss	inconveni	3.60%
Cluster 6 :- Size: 132; ISim: 0.069; ESIm: 0.025						
Descriptive:	women	Gender	4.20%	moga	bhamje	1.80%
Discriminating:	women	Gender	5.00%	bhamje	gaolath	1.40%
Cluster 7 :- Size: 128; ISim: 0.048; ESIm: 0.025						
Descriptive:	Busi	Vision	5.40%	fair	societi	1.30%
Discriminating:	Vision	Busi	7.00%	fair	bitf	1.30%

15 cluster complete linkage agglomerative clustering solution

Cluster 8 :- Size: 108; ISim: 0.067; ESIm: 0.022								
Descriptive:	tourism	16.10%	Wast	8.10%	environment	3.10%	conserv	2.00%
Discriminating:	tourism	16.10%	Wast	8.20%	environment	3.00%	conserv	1.90%
Cluster 9 :- Size: 44; ISim: 0.121; ESIm: 0.025								
Descriptive:	councillor	30.60%	council	11.00%	mayor	6.30%	matimela	3.80%
Discriminating:	councillor	24.60%	council	6.10%	mayor	5.40%	matimela	3.40%
Cluster 10 :- Size: 72; ISim: 0.082; ESIm: 0.024								
Descriptive:	Health	12.60%	Patient	5.50%	educ	4.10%	blood	4.00%
Discriminating:	Health	10.80%	Patient	5.20%	blood	4.20%	nurs	3.00%
Cluster 11 :- Size: 154; ISim: 0.103; ESIm: 0.018								
Descriptive:	Team	15.50%	Zebra	10.50%	game	5.80%	cup	5.10%
Discriminating:	Team	9.60%	Zebra	8.00%	cup	3.60%	game	3.20%
Cluster 12 :- Size: 1426; ISim: 0.025; ESIm: 0.022								
Descriptive:	Land	1.40%	Polic	1.30%	music	1.20%	album	1.10%
Discriminating:	Parti	4.30%	Bdp	3.50%	album	3.30%	music	3.20%
Cluster 13 :- Size: 83; ISim: 0.091; ESIm: 0.022								
Descriptive:	Water	31.80%	Flood	7.80%	dam	2.50%	river	2.00%
Discriminating:	Water	25.30%	Flood	7.10%	dam	2.10%	river	1.50%
Cluster 14 :- Size: 197; ISim: 0.052; ESIm: 0.024								
Descriptive:	Elect	7.80%	Farmer	4.60%	agricultur	4.40%	iec	3.20%
Discriminating:	Elect	6.20%	Farmer	5.60%	agricultur	5.10%	iec	4.40%

Table 4.9: 15-cluster complete linkage agglomerative clustering solution

4.8 Comparing Direct and Bisecting K-way Clustering

In Table 4.10, we compare the 15 cluster solutions from the bisecting k-way technique and the direct k-way technique. Incidentally, the table should not be interpreted as suggesting some sort of correspondence between clusters from one solution and clusters from the other solution. The bisecting k-way technique produced clusters that are more uniformly sized than those produced by the direct k-way technique; the former produced clusters with ranging in size from 194 to 464 (standard deviation of 71.2), while the latter produced clusters with ranging in size from 96 to 467 (standard deviation of 112.3). This is in keeping with the assertion by Steinbach *et al.* (2000:17) that “bisecting k-means tends to produce clusters of a relatively uniform size, while regular k-means is known to produce clusters of widely different sizes”. While Steinbach *et al.* advanced this as the reason bisecting k-means works better than regular k-means, in our case, despite the variability in the cluster sizes it produced, direct k-way slightly outperformed bisecting k-way on internal cohesion and external isolation. For direct k-way clustering, $\sum ISim = 1.17$, while it is $\sum ISim = 1.02$ for bisecting k-way clustering; the higher $\sum ISim$ figure for the former suggests that its clusters are more internally cohesive. For direct k-way clustering, $\sum ESim = 0.302$, while it is $\sum ESim = 0.307$ for bisecting k-way clustering; the lower $\sum ESim$ figure for the former suggests its clusters are more externally isolated. Subjectively, we also found the direct k-way clustering solution slightly easier to interpret than the bisecting k-way clustering solution. Given these three findings, we preferred the direct k-way clustering solution over its bisecting k-way counterpart.

The poor results from agglomerative clustering notwithstanding, the close agreement between the results from direct k-way clustering and k-way clustering through repeated bisections is a good indicator that the clusters identified in this study are indeed valid. We also randomly selected 75 stories (5 per cluster) that we read through to determine the agreement between the cluster labels we assigned and the thematic content of the stories assigned to those clusters; the results are captured in Table 4.11; in the table, the ‘Positive’ column gives the number of stories, out of five, that agree with the cluster label to which they have been assigned.

BISECTING						DIRECT					
CID	Size	ISim	ISdev	ESim	ESdev	CID	Size	ISim	ISdev	ESim	ESdev
0	464	0.115	0.032	0.023	0.005	0	96	0.168	0.06	0.021	0.005
1	288	0.084	0.027	0.015	0.004	1	427	0.122	0.032	0.023	0.005
2	194	0.075	0.026	0.018	0.005	2	278	0.091	0.023	0.015	0.004
3	207	0.078	0.024	0.026	0.006	3	161	0.099	0.03	0.025	0.006
4	258	0.067	0.026	0.016	0.006	4	240	0.081	0.029	0.016	0.005
5	295	0.064	0.026	0.016	0.005	5	179	0.079	0.027	0.018	0.005
6	255	0.066	0.023	0.022	0.006	6	188	0.076	0.023	0.016	0.005
7	237	0.061	0.021	0.017	0.005	7	233	0.076	0.026	0.017	0.006
8	246	0.064	0.027	0.022	0.006	8	240	0.069	0.023	0.022	0.006
9	408	0.062	0.019	0.022	0.006	9	467	0.057	0.018	0.021	0.006
10	303	0.063	0.02	0.023	0.005	10	343	0.057	0.019	0.022	0.006
11	297	0.061	0.018	0.023	0.007	11	405	0.053	0.016	0.022	0.006
12	246	0.055	0.017	0.022	0.007	12	280	0.05	0.016	0.019	0.006
13	326	0.053	0.014	0.021	0.007	13	358	0.051	0.014	0.022	0.007
14	304	0.052	0.015	0.021	0.006	14	433	0.041	0.011	0.023	0.006
Σ	4328	1.02	0.335	0.307	0.086	Σ	4328	1.17	0.367	0.302	0.084

Table 4.10: A comparison of the bisecting and direct k-way 15 cluster solutions

CID	Cluster Name	Positive	%
0	Land Allocation and Related Issues	2	40
1	Political Parties and Related Matters	3	60
2	National Football Team	1	20
3	National Elections and Related Matters	2	40
4	Music	3	60
5	Sports	1	20
6	Legal Matters	1	20
7	Policing and Crime Prevention	3	60
8	Schooling and Education	3	60
9	HIV / AIDS	1	20
10	Regional Bodies	3	60
11	Development Issues	3	60
12	Agriculture and Tourism	2	40
13	Financial and Business News	3	60
14	The World of Work	0	0

Table 4.11: Agreement of randomly selected stories with their cluster labels

For 47% (7/15) of the clusters, 60% (3/5) of the selected stories agreed with the cluster labelling; for 7% (1/15) of the clusters, there was no agreement for the stories selected. For the remaining clusters, there was agreement 27% (4/15) had a 20% (2/5) agreement, while 20% (3/15) had a 40% (2/5) agreement. While not excellent, we feel that these figures are encouraging.

4.9 Quarterly Analysis

In Tables 4.12, 4.13, 4.14 and 4.15 we show the results of a cluster analysis of our corpus on a quarterly basis. We only present here our interpretation of the data; the actual output from gCLUTO is attached as Appendix E. The one theme that stands out in the first quarter that does not feature when looking at the overall corpus is the ‘location of the new university’; this is in reference to the work of the consultative task force setup by the government of Botswana to solicit views on where to locate the proposed second university. Noticeably, the ‘national elections’ theme which emerged from the analysis of the complete corpus is absent from the analysis of the quarterly corpus. Thus, although 2004 was an election year, coverage of election issues during the first quarter of the year was still low. This is hardly surprising because although it was publicly known that the year was indeed an election year, the actual polling day had not yet been announced. Nevertheless, political parties were, as cluster 2 indicates, already active.

The most prominent theme during the second quarter, which again did not emerge from the analysis of the complete corpus, is that of the ‘polio

CID	Cluster Name
0	Location of New University
1	Music
2	Political Parties and Related Matters
3	Sports
4	Legal Matters
5	Sports (National Football Team?)
6	Policing and Crime Prevention
7	Schooling and Education
8	Land Allocation and Related Matters
9	Village Development Issues
10	Parliamentary News
11	Business and Financial News
12	Tourism and Trade Issues
13	HIV / AIDS
14	Developments (mainly local government)

Table 4.12: Quarterly news analysis with the direct k-way clustering technique: first quarter

CID	Cluster Name
0	Land Allocation and Related Issues
1	Political Parties and Related Matters
2	Music
3	Regional Bodies (Including Regional Trade)
4	Polio Immunisation
5	Policing and Crime Prevention
6	National Football Team
7	Legal Matters
8	HIV / AIDS
9	Privatisation and Tourism
10	Culture and Arts (including beauty pageants)
11	Local Government and Service Delivery
12	Farming and Agriculture
13	Financial and Regional News
14	Employment Issues

Table 4.13: Quarterly news analysis with the direct k-way clustering technique: second quarter

CID	Cluster Name
0	Mines (Employer/Employee Relations)
1	Sports
2	Land Allocation and Related Issues
3	Music
4	Regional News
5	Political Parties and Related Matter
6	National Football Team
7	Legal Matters
8	Policing and Crime Prevention
9	Village Developments
10	HIV / AIDS
11	Farming and Agriculture
12	Culture and Related Matters
13	Business and Financial News
14	Tourism and Environmental Conservation

Table 4.14: Quarterly news analysis with the direct k-way clustering technique: third quarter

CID	Cluster Name
1	General Elections
1	Music
2	Political Parties
3	Sports
4	Legal Matters
5	HIV / AIDS
6	Sports and Disasters / Disabilities
7	Policing and Crime Prevention
8	Women in Parliament
9	Schooling and Education
10	Farming and Agriculture
11	Government
12	Beauty Pageants ???
13	Tourism
14	Local Government

Table 4.15: Quarterly news analysis with the direct k-way clustering technique: fourth quarter

immunisation campaign'. The campaign made the news largely because the government intended it that way, and naturally took full advantage of its own newspaper (the *Botswana Daily News*, which is the subject of our study), but also because of some communities who refused to have their children immunised, thereby generating debate around the 'polio immunisation campaign' theme. In the first quarter, there were two separate clusters that were really both about sports news; now in the second quarter there is just one, and it relates specifically to the national soccer team, 'the zebras'. The 'location of the new university theme' appears to have been exhausted, while a new theme about around employer / employee relations is just emerging.

Moving to the third quarter, the 'polio immunisation campaign' of the second quarter has abated. In contrast, the 'employer / employee relationships' theme that began to emerge in the second quarter is still visible in the third quarter. It would appear that this issue was brought to the fore largely by well publicised 'strikes' by employees of the Debswana Diamond Mining Company; counter intuitively, in a 10-cluster solution, the 'strikes' in question emerge in a cluster of their own, but not in the 15-cluster solution. Finally, we consider the fourth quarter of the study period. Interestingly, the 'National Elections' theme, which we identified in the cluster analysis of the complete corpus, is only clearly visible in the fourth quarter, the quarter in which the elections were actually held (polling day was 30 October 2004).

4.10 Summary

In this study we undertook a cluster-analytic exploration of news stories published in the *Botswana Daily News* during the period January through December 2004. The current chapter was a discussion and interpretation of our experimental results. We began the chapter with a brief description of the data used in this study, followed by a discussion of the output produced by the software used in the experiments. We experimented with three different clustering algorithms, namely, a direct k-way clustering algorithm, a bisecting k-way clustering algorithm, and a hierarchical, agglomerative clustering algorithm; all three algorithms are implemented as part of the CLUTO package that we use in our study. Though somewhat arbitrary, our target number of clusters was 15.

The results of agglomerative clustering were rather poor, and were thus discarded. Although the two k-way clustering algorithms produced results that were quite similar, upon further inspection we concluded that direct k-way clustering had fared better than k-way clustering through repeated bisections. The close similarity between the results produced by the two clustering techniques, however, served to reinforce the validity of the clustering results, as did the small sample of stories that were checked to determine whether their thematic content agreed with the labels of the clusters to which they had been

assigned. Interestingly, a quarterly cluster analysis of the data revealed some themes that were not identified by the cluster analysis of the complete corpus. In the next chapter, we summarise our achievements, present our conclusions, and also indicate avenues for further research.



Chapter 5

Summary, Conclusions and Further Work

5.1 Introduction

In this study, we set out to undertake a cluster-analytic analysis of news stories published in the *Botswana Daily News* during the period January through December, 2004. There are many reasons why the analysis of news stories is interesting. For instance, news analysis may be undertaken as part of a competitive intelligence function of an organisation, to monitor societal interests, or even to assess whether a newspaper is biased in its reporting. We introduced our study in Chapter 1, and subsequently, in Chapter 2, presented the foundational material upon which the study itself is based. Specifically, in Chapter 2 we discussed information retrieval models (with emphasis on the vector space model used in this study), index term selection, and cluster analysis. Our use of cluster analysis was motivated by two factors; firstly, given the volume of data to be analysed, a machine learning approach, as opposed to a manual approach, was imperative; secondly, the study is exploratory in nature, making cluster analysis a natural choice. In Chapter 3, we presented the work actually undertaken in this study, while Chapter 4 was devoted to data analysis and interpretation. As its heading suggests, in this our final chapter, we seek to accomplish three objectives. Firstly, we present a summary of the work undertaken in this study. Secondly, we present our conclusions; in a way, this may also be viewed as a summary of the data analysis and interpretation presented in Chapter 4. Finally, we consider avenues for further work.

5.2 Summary of Work Undertaken

The work undertaken in this study brought together ideas from such diverse areas as Information Retrieval, Natural Language Processing, and Machine Learning. Cluster analysis seeks to group together similar objects. In our

case, the objects being compared are documents, and as such it was necessary to model documents. Document models provide a framework within which the similarity of documents may be quantified, and therefore, compared. Modelling documents is an important part of information retrieval. In Chapter 2, we discussed the topic of information retrieval models, focusing specifically on the vector space model used in this study, in some detail. Corresponding to the four components of an information retrieval model suggested by Baeza-Yates and Ribeiro-Neto (1999), the vector space model has the following characteristics: (i) a document is reduced to a set of index terms (without regard to the relationship between the terms in the original document i.e. bag of words model), which are then used to create corresponding document vectors (note also that index terms may be individual words, word stems, or phrases); (ii) queries are considered documents; (iii) although other approaches are possible, the similarity between documents is almost invariably measured in terms of the cosine coefficient; and (iv) the larger the angle between the vectors of two documents, the more dissimilar the original documents; this makes it possible to rank documents according to their perceived relevance to a given query, with similarity used as a surrogate for relevance. A number of activities are undertaken in order to transform documents into vector form, and next we review the process followed in this study.

The data used in this study was downloaded from the Botswana government website, *Botswana Daily News* section. A number of document pre-processing activities, using software utilities developed in this study, were then undertaken. First, valid data files (i.e. files actually containing news stories) were separated away from invalid data files (i.e. those containing a message indicating that data is not available for the specified day). This step was necessitated by the fact that the data collection utility we developed simply requests the web page for a particular date from the server, and in response the server sends a web page that either contains the stories for the specified date, or is a message indicating that data is not available for the specified date. All valid data files were then stripped of recurring information that appeared on each story, either at the top (header), or at the bottom (footer), after which the stories were assigned IDs, and the complete corpus merged into a single file. A rather simple but effective procedure to remove HTML codes from our corpus entailed opening the corpus file with Microsoft Internet Explorer, and using the 'save as' option to save the document as a plain text file.

For document clustering proper, we used CLUTO which is really a collection of clustering algorithms. CLUTO also comes with an associated utility, *doc2mat*, which converts raw documents into the matrix format required by CLUTO; in the process, *doc2mat* undertakes tokenisation, stop word removal, and stemming. Tokenisation in *doc2mat* is done in two steps; (i) first, all non-alphanumeric characters are replaced with spaces, and then (ii) individual tokens, which are assumed to be bounded by white spaces, are extracted. For stop word removal, we used the list designed for use with SMART, while for

stemming we used Porter's stemmer. It is common in information retrieval to use Zipf's Law to set upper and lower cut-off points for what terms to include in the creation of an index. However, we felt that for the purposes of the current study, stop word removal and stemming were sufficient, and therefore did not use Zipf's Law at all. In terms of clustering, we experimented with three clustering algorithms, namely, direct k-way clustering, k-way clustering through repeated bisections, and agglomerative clustering. Agglomerative clustering produced low quality results which we thus discarded. Direct k-way clustering and k-way clustering through repeated bisections produced similar results, with the former slightly out performing the latter in terms of internal cohesion and external isolation of the clusters formed. Consequently, we retained the results yielded by direct k-way clustering, only using k-way through repeated bisections for validation. We subsequently used direct k-means clustering for a quarterly exploration of our corpus. In the next section, we summarise the findings of our study.

5.3 Summary of Findings

5.3.1 Analysis of Complete Corpus

In this section, we present a summary of our findings, and also comment on the wider contributions of our study. As we explained in Chapter 4, in an exploratory cluster analysis study, a certain level of arbitrariness is inevitable when deciding the number of clusters into which the data should be partitioned. We settled for 15 clusters, which we then named as shown in Table 4.6. A cursory look at Table 4.6, however, suggests that we could in fact merge some clusters because they are essentially variations of the same theme; a case in point is cluster 2, about the national football team, and cluster 5, about sports, which can be collapsed into just one cluster appropriately named 'Sports'. Proceeding in this way reduced the original 15 clusters to 12. Our final categorisation of news stories published in the *Botswana Daily News* in 2004 is thus depicted in Table 5.1. We endeavoured to assign cluster labels that, as much as possible, are self-explanatory. Nevertheless, we will make a few comments to buttress our taxonomy. The final cluster names are:

1. *Land Allocation and Related Issues*: this cluster carries over as is from Table 4.5. As its name suggests, the cluster contains stories dealing, on the main, with the allocation of land.
2. *Political Matters*: two clusters were merged to form this cluster; they are 'Political Parties and Related Matters', and 'National Elections and Related Matters', both of which are really concerned with political matters.

3. *Sports and Entertainment*: we have collapsed three clusters to form this cluster. The original clusters were labelled 'National Football Team', 'Sports', and 'Music'; the first two are clearly about 'Sports', and both 'Sports' and 'Music' are really about entertainment.
4. *Legal News*: Stories in this cluster will mainly be about issues that involve the courts e.g. litigation.
5. *Policing and Crime Prevention*: includes stories about incidences of crime, as well as the arrest of suspects. Obviously, there will be some overlap with the 'Legal Matters' cluster above as, for instance, suspects may be arraigned in court.
6. *Schooling and Education*: deals with matters pertaining to formal education in schools.
7. *Health News*: Stories in this cluster are largely about HIV / AIDS. However, quarterly analysis of the corpus suggests that other diseases too were reported on in 2004. For this reason, we have chosen to rename this cluster 'Health News'.
8. *Regional Bodies*: Regional bodies include the Southern African Development Community (SADC), as well as the group of countries known as the ACP (African, Caribbean and Pacific) countries. Africa, too, may be considered a region.
9. *Development Issues*: development here is used in the sense of improving the lives of general citizenry. Typical stories are about issues such as infrastructure (e.g. roads) development, and water supply.
10. *Agriculture and Tourism*: intuitively, one is uncomfortable lumping together agriculture and tourism. There is, however, a real, practical explanation for this finding: agriculture and tourism have competing needs that therefore result in them being reported on in tandem in the papers. For instance, wild life and domestic animals compete for grazing areas, and the former frequently cause damage to agricultural crops.
11. *Financial and Business News*: this includes news relating to government's privatisation drive, as well as the role of banks (both the various commercial banks and the central bank i.e. Bank of Botswana) in the economy.
12. *The World of Work*: it was not easy to coin an appropriate label for this cluster although intuitively one recognises what it is all about. Stories in this cluster include those dealing with employer - employee relations (e.g. industrial action by employees), as well as productivity in the work place.

CLUSTER	NAME	REMARKS
1.	Land Allocation and Related Issues	Self explanatory
2.	Political Matters	Subsumes 'Political Parties and Related Matters' and 'National Election and Related Matters'
3.	Sports and Entertainment	Subsumes 'National Football Team', 'Music' and 'Sports'
4.	Legal News	Self explanatory
5.	Policing and Crime Prevention	Self explanatory
6.	Schooling and Education	Self explanatory
7.	Health News	Renamed. Originally 'HIV / AIDS'
8.	Regional Bodies	e.g. SADC, ACP, Africa
9.	Development Issues	e.g. Water supply, construction of roads
10.	Agriculture and Tourism	Self explanatory
11.	Financial and Business News	e.g. (commercial) banking, privatisation
12.	The World of Work	e.g. productivity, employee relations

Table 5.1: Taxonomy of news stories published in the *Botswana Daily News*, January - December 2004.



5.3.2 Quarterly Analysis

We also performed a quarterly analysis of our data. To achieve this, we partitioned our corpus into four parts, each representing three months of the year, and then cluster analysed each quarter individually. When analysing the complete corpus we were interested in all the clusters present in the data. In contrast, our interest when analysing the corpus on a quarterly basis was to get a sense of variation of news coverage over the year. As such, we focused on clusters that emerged from quarterly analysis that had not been apparent when analysing the complete corpus. Our findings are summarised in Table 5.2

A theme that emerged in the cluster analysis of data from the first quarter which had not emerged in the analysis of the complete corpus is the national debate on the location of the second university. Similarly, analysis of data from the second quarter revealed a theme that analysis of the complete corpus had failed to pick, namely the polio immunisation campaign. An interesting difference between the first and second quarters is that while in the former

Quarter	Theme
1	Location Of The New University
2	Polio Immunisation Campaign
3	Employer / Employee Relationships
4	National Elections

Table 5.2: Summary of Results From Quarterly Analysis of Corpus

there were two separate clusters that both dealt with sports issues, in the latter sports news were predominantly about the national soccer team, also known as the ‘Zebras’. The ‘employer / employee’ relations theme began to emerge during the second quarter and became quite topical in the third quarter. Specifically, the news stories in this cluster seemed to be mostly about strikes by employees of the Debswana Diamond Mining Company. Finally, in the fourth quarter, a theme that had emerged in the cluster analysis of the complete corpus also emerged in the analysis of data from the single quarter, namely the National Elections. While one would have expected this theme to be quite topical throughout 2004 - an election year - it would appear that the theme only became topical around the actual polling day.

5.3.3 Wider Contributions Of The Study

We see the contributions of our study as being wider than the specific objectives we set out to achieve. The study adds to the growing literature on text mining, specifically cluster analysis, which demonstrates that the application of machine learning techniques to the analysis of textual corpora is both feasible, and beneficial. This is important because the amount of data organisations have to sift through to make important business decisions keeps growing exponentially. Useful as it is, data mining restricts itself to the analysis of structured data, such as may be found in typical online transaction processing systems. However, it has been suggested that the most natural form of storing information is as text, which would in turn suggest that mining text might have a higher commercial value than mining structured data (Yu, Wang, and Lai, 2005); indeed, quoting earlier work by the Delphi Group, Yu et al suggest that up to 80% of a company’s information is typically contained in textual documents. While there is indeed a growing body of literature on text mining, much of it tends to be more on theory than on practice; this is particularly true of research emanating from computer science (machine learning) where the focus is on efficiency of algorithms as tested against standard corpora, than on the applications of algorithms to real world data. This state of affairs was also acknowledged by Holsheimer and Siebes (1994:8), who lamented that “[t]here is a growing interest in data mining, both in research and application. Unfortunately, it is difficult to provide an overview of success in the latter

area since experimental results are often confidential because of their strategic importance”. Our research helps fill this yawning void.

5.4 Further Work

In this section, we consider avenues for further research. In so doing, we will address three main themes. Firstly, we will look at improvements to the current study that nevertheless continue to use the same clustering techniques utilised in the current study. We will then consider ways in which the research reported here can be extended through the use of other clustering techniques. Finally, we will suggest ways in which the research can be extended by using other text mining techniques - over and above clustering techniques - and additional data to provide more insight than could be provided within the confines of the current study.

In Chapter 2, we indicated that when selecting terms to use for indexing documents, nouns are sometimes preferred because they are considered to carry the semantics of documents (Baeza-Yates and Ribeiro-Neto, 1999). In our study, however, we did not explicitly seek to extract nouns for indexing purposes, opting instead to go with the word count approach popular in information retrieval work. Natural language processing techniques, such as part of speech tagging, can be used to identify nouns in documents. Additionally, while our approach to tokenisation focused on single words, it might be useful to consider using phrases for indexing purposes. While on text processing issues, other improvements too can be made to the current study. Often, hyphens are used to indicate that a word extends from the end of line to the start of the next line; it may be useful to ‘reconstitute’ such words before the actual process of index term selection begins. In this study, we decided against the use of Zipf’s Law as a feature reduction technique; while our intuition is that the use of the law would not significantly alter the findings of this study, experimenting with the law might still prove be fruitful.

In Chapter 2, when discussing information retrieval models, we pointed out that though the vector space model, which we used in this study, is arguably the most popular information retrieval model, it is not the only one; others include the Boolean model, the probabilistic model, and fuzzy retrieval. One way in which the current study may be extended would be through the use of an alternative information retrieval model. In this regard, we would be inclined to believe that probabilistic and fuzzy approaches hold more promise than the standard or extended Boolean models. Implicitly, the assumption in this study was that a story belonged to one, and only one, cluster. However, this assumption is, in practice, too strict. Consider, for example, a story about college students performing a musical piece at a crime prevention seminar. The story might be considered to be in the ‘Schooling and Education’, ‘Sports and Entertainment’, or ‘Policing and Crime Prevention’ categories. Probabilistic

approaches resolve this dilemma by allowing a document to simultaneously belong to two or more clusters, but with varying probabilities.

An improvement on the vector space model is the so-called latent semantic indexing (LSI) approach. Witter and Berry (1998) motivate the use of LSI with the following words:

As an information retrieval technique, simple lexical matching is insufficient, often producing inaccurate and irrelevant results. This is due to synonymy (multiple ways to express a concept) and polysemy (multiple meanings for a word or words) within the set of information to be searched . . . LSI assumes that an underlying semantic structure of word usage exists in a document collection and uses this to estimate the semantic content of documents in the collection . . . Using LSI, relevant terms and/or documents can be matched and retrieved even when the query contains no words in common with the relevant documents.

With regards to the use of additional data, two avenues may be considered. Firstly, the study may be extended to cover a longer period than the twelve months we limited ourselves to. Our choice of the year 2004 was motivated by the fact of it being an election year. While that meant that it would potentially yield interesting results, it also meant that it was not 'typical' year, and such might have masked trends common to typical years; using data from a longer period (say five years) would help mitigate these effects. Secondly, the study could be extended by using data from other local newspapers to allow for comparative analysis. This, naturally, would be subject to the availability of data from the other newspapers, as well as willingness on their part to allow its use for the intended research.

Due to time and other resource constraints, in this study we only experimented with three clustering techniques, namely, agglomerative clustering and partition-based clustering (direct k-way clustering and k-way clustering through repeated bisections). Further, the two partitioning algorithms were really variations of the same basic algorithm. It would thus be interesting to apply other clustering algorithms to the data to see whether analysis results could be improved. Often a distinction is made between hard clustering, in which each document belongs to exactly one cluster, and soft clustering which simply gives the probability that a given document belongs to a particular class.

Finally, we consider extending the current research by using text mining techniques otherwise not used in this study. The work undertaken in this study falls within the discipline of text mining. Like its cousin data mining, text mining seeks to discover knowledge in existing databases. In the current study, we only restricted ourselves to unsupervised learning in the form of document clustering. Other text mining approaches too can be used to reveal

interesting patterns inherent in the data. Indeed, cluster analysis is often an entry point into the knowledge discovery process that also includes other data / text mining approaches. Approaches that may be worth pursuing include text classification (which may be used to test the taxonomy created through cluster analysis), association rule mining (which may reveal some form of affinity between topics) and trend analysis (to explore the evolution of topics over time).



References

- Bacher, J. 2002. Cluster Analysis. Available at: <http://www.soziologie.wiso.uni-erlangen.de/koeln/>, [2005, August 28].
- Baeza-Yates, R. and Ribeiro-Neto, B. 1999. *Modern Information Retrieval*. New York: Addison-Wesley.
- Belew, R.K. 2000. *Finding Out About: A Cognitive Perspective on Search Engine Technology and the WWW*. Cambridge: Cambridge University Press.
- Berry, M. (ed.). 2004. *Surveying Text Mining: Clustering, Classification, and Retrieval*. New York: Springer.
- Berry, M.J.A. and Linoff, G. 1997. *Data Mining Techniques for Marketing, Sales, and Customer Support*. New York: Wiley.
- Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. and Wirth, R. 2000. CRISP-DM 1.0: Step-by-Step Data Mining Guide. Available at: <http://www.crisp-dm.org/CRISPWP-0800.pdf>, [2005, September 01]. The CRISP-DM consortium.
- Cook, M. and Cook, C. 2000. *Competitive Intelligence: Create an Intelligent Organization and Compete to Win*. London: Kogan Page.
- Dilly, R. 1995. Data Mining: An Introduction (Version 2.0). Student Notes (Parallel Computer Centre, Queens University Belfast).
- Dunham, M.H. 2003. *Data Mining: Introductory and Advanced Topics*. Upper Saddle River, New Jersey: Prentice Hall / Pearson Education.
- Everitt, B. 1993. *Cluster Analysis*. London / New York: E. Arnold / Halstead Press.
- Gordon, A.D. 1981. *Classification: Methods for the Exploratory Analysis of Multivariate Data*. London: Chapman and Hall.
- Hair, J.F., Anderson, R.E., Tatham, R.L. and Black, W.C. 1995. *Multivariate Data Analysis with Readings*. 4th edition. Englewood Cliffs, New Jersey: Prentice Hall.

- Hand, D., Mannila, H. and Smyth, P. 2001. *Principles of Data Mining*. Cambridge, Massachusetts / London, England: MIT Press.
- Jain, A.K., Murty, M.N. and Flynn, P.J. 1999. Data Clustering: A Review. *ACM Computing Surveys*, 31(3).
- Jurafsky, D. and Martin, J.H. 2000. *Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Upper Saddle River, New Jersey: Prentice Hall.
- Karanikas, H. and Theodoulidis, B. 2002. Knowledge Discovery in Text and Text Mining Software. Technical Report, CRIM, UMIST.
- Karypis, G. 2003. CLUTO: A Clustering Toolkit (Release 2.1.1). Technical Report 02-017, Department of Computer Science, University of Minnesota.
- Korfhage, R.R. 1997. *Information Storage and Retrieval*. New York: John Wiley.
- Lan, M., Tan, C., Low, H. and Sung, S. 2005. A Comprehensive Comparative Study on Term Weighting Schemes for Text Categorization with Support Vector Machines. Available at: [www2005.org/cdrom/docs/p1032.pdf](http://www.www2005.org/cdrom/docs/p1032.pdf), [2005, May 17]. The 14th International World Wide Web Conference (WWW2005).
- Large, A., Tedd, L.A. and Hartley, R.J. 1999. *Information Seeking in the Online Age: Principles and Practice*. London: Bowker-Saur.
- Manning, C.D. and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: MIT Press.
- McGonagle, J. and Vella, C.M. 1998. *Protecting Your Company Against Competitive Intelligence*. Westport, Connecticut: Quorum Books.
- Perry, C. 1995. A Structured Approach To Presenting PhD Theses: Notes For Candidates And Their Supervisors. Paper presented to the ANZ Doctoral Consortium, University of Sydney, February 1994 (with later additions to 18 September 1995).
- Roiger, R.J. and Geatz, M.W. 2003. *Data Mining: A Tutorial-Based Primer*. Boston: Addison Wesley / Pearson Education Inc.
- Romesburg, C.H. 1984. *Cluster Analysis for Researchers*. Belmont, California: Lifetime Learning Publications.
- Russel, S. and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*. 2nd edition. Upper Saddle River, N.J.: Prentice Hall.

- Sahami, M. 1998. Using Machine Learning to Improve Information Access. Ph.D. thesis, Computer Science, Stanford University. Available at <http://robotics.stanford.edu/users/sahami/papers-dir/thesis.pdf>, [2005, August 28].
- Salton, G. and McGill, M.J. 1983. *Introduction to Modern Information Retrieval*. New York: McGraw-Hill.
- Steinbach, M., Karypis, G. and Kumar, V. 2000. A Comparison of Document Clustering Techniques. Technical Report 00-034, Department of Computer Science and Engineering, University of Minnesota.
- Sturn, A. 2000. Cluster analysis for large scale gene expression studies. Master's thesis, Institute for Biomedical Engineering, Graz University of Technology, Austria.
- Subrahmanian, V.S. 1998. *Principles of Multimedia Systems*. San Francisco, California: Morgan Kaufmann.
- Tombros, A. 2002. The effectiveness of query-based hierarchic clustering of documents for information retrieval. Ph.D. thesis, Department of Computing Science, University of Glasgow.
- Two Crows Corporation. 1999. Introduction to Data Mining and Knowledge Discovery. Available at: <http://www.twocrows.com/>, [2005, August 28].
- van Rijsbergen, C. 1979. *Information Retrieval*. Boston: Butterworths.
- West, C. 2001. *Competitive Intelligence*. Basingstoke, Hampshire / New York: Houndmills / Palgrave.
- Witten, I.H. and Frank, E. 2000. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. San Diego: Academic Press.
- Witter, D.I. and Berry, M.W. 1998. DOWDATING THE LATENT SEMANTIC INDEXING MODEL FOR CONCEPTUAL INFORMATION RETRIEVAL. *The Computer Journal*, 41(8).
- Zhao, Y. and Karypis, G. 2002. Criterion Functions for Document Clustering: Experiments and Analysis. Technical Report 01-40, Department of Computer Science / Army HPC Research Center, University of Minnesota. Accessed 05 May 2005.

Appendix A

Source Code Listings

grabdata.cs
Queries the Botswana Daily News website for news [extracts] for a specific day and saves the response it receives to a data file.

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
using System.Net;

namespace grabData
{
    class Program
    {
        static void Main(string[] args)
        {
            DateTime startDate = new DateTime(2004,1,1);
            DateTime endDate = new DateTime(2004, 12, 31);

            WebRequest request; // Request data from server
            HttpWebResponse response; // Get the response.
            Stream dataStream; // Get the stream containing
                // content returned by
                // the server.
            StreamReader reader; // Open the stream using a
                // StreamReader for easy access.
            string responseFromServer; // Read the content
            string outFileName, processDate;
```



```
Console.WriteLine("Working ... please wait ...");
while (startDate < endDate)
{
    Console.WriteLine("\tNow processing: "
        + startDate.Date);

    processDate = startDate.Year.ToString();
    processDate = processDate
        + startDate.Month.ToString("d2");
    processDate = processDate
        + startDate.Day.ToString("d2");

    // Create a request for the URL.
    request =
        WebRequest.Create("http://www.gov.bw/cgi-bin/news.cgi?d="
            + processDate);

    // Get the response.
    response = (HttpWebResponse)request.GetResponse();

    // Get the stream containing content returned by the server.
    dataStream = response.GetResponseStream();

    // Open the stream using a StreamReader for easy access.
    reader = new StreamReader(dataStream);

    // Read the content.
    responseFromServer = reader.ReadToEnd();

    // Write the string to a file.
    outFileName = processDate + ".txt";
    System.IO.StreamWriter file = new
        System.IO.StreamWriter("c:\\dailynews\\" + outFileName);
    file.WriteLine(responseFromServer);

    file.Close();

    // next day
    startDate = startDate.AddDays(1.0);
}

// Done
Console.WriteLine("Done ... thank you for your patience.");
}
```

```
    }  
}
```

isvf.cpp

Checks if specified file is a valid data file. If specified file contains actual news stories, the program responds 'yes', otherwise it responds 'no'.

```
#include <iostream>  
#include <stdlib.h>  
#include "FileChecker.H"  
#include "Exception.H"  
  
using namespace std;  
  
int main(int argc, char *argv[])  
{  
    // Ensure proper number of arguments  
    if(argc != 2) {  
        cout << "\nUsage: isvf <filename>\n";  
        return EXIT_FAILURE;  
    }  
  
    // check if given file is a valid data file  
    string nameOfFile = argv[1];  
    FileChecker myFileChecker;  
    try {  
        if(myFileChecker.isValidDataFile(nameOfFile))  
            cout << "yes.\n\n";  
        else  
            cout << "no.\n\n";  
    }  
  
    catch(Exception exp) {  
        cout << "\n" << exp.getDescription();  
        cout << " ... Bailing out ... \n\n";  
    }  
  
    catch(...) {  
        cout << "Unknown error ... bailing out!\n\n";  
    }  
}
```

```
    return 0;
}
```

listvf.cpp

Lists all the valid files in the specified directory, and also gives a count of such files.

```
#include <dirent.h>
#include <iostream>
#include <cstdlib>
#include <list>
#include "FileChecker.H"
#include "FileManager.H"

using namespace std;

int main(int argc, char *argv[])
{
    // correct number of command line parameters?
    if(argc != 2) {
        cout << "\nUsage:";
        cout << "\n\t\t\tlistvf <directoryName>" << endl;
        cout << "\nPlease Note: directory must not contain";
        cout << " subdirectories." << endl;
        return EXIT_FAILURE;
    }

    // read file names into list
    string searchDirectory = argv[1];
    FileManager myFileManager;
    list<string> fileList;    // list of valid files
    try {
        myFileManager.getDirFiles(searchDirectory, fileList);
    }
    catch(Exception exp) {
        cout << "\nError: " << exp.getDescription() << endl;
    }

    // for each file, check if it is valid
    int validFileCount = 0;
    FileChecker myFileChecker;
    list<string>::iterator lstItr; // to traverse list
    // of valid files
    cout << "Valid data files:" << endl;
```

```

try {
    lstItr = fileList.begin();
    while(lstItr != fileList.end()) {
        if(myFileChecker.isValidDataFile(*lstItr)) {
            cout << '\t' << *lstItr << endl;
            ++validFileCount;
        }
        ++lstItr;
    }
}

catch(Exception exception) {
    cout << "Error: " << exception.getDescription();
}

// how many valid files found?
cout << "\n" << validFileCount << " file(s)." << endl;

// done
return EXIT_SUCCESS;
}

```

copyvf.cpp

Used to copy data files from one directory to another; it can also be used to copy a single data from one directory to another, or even to the same directory but under a different file name.

```

#include <iostream>
#include <string>
#include <list>
#include "FileChecker.H"

using namespace std;

int main(int argc, char *argv[])
{
    // ensure proper number of command line arguments
    if((argc < 3) || (argc > 4)) { // error: terminate program
        cout << "\nUsage" << endl;
        cout << "copyvf <sourceDirectory> <destinationDirectory>" << endl;
        cout << "Please note: ";
        cout << "sourceDirectory> must not contain subdirectories." << endl;
    }
}

```



```
    return EXIT_FAILURE;
}

// get directory names
string sourceDirectory = argv[1];
string destinationDirectory = argv[2];

// read source file names into list
int validFileCount = 0;
list<string> fileList;    // list of valid files
list<string>::iterator lstItr; // to traverse list of valid files
FileChecker myFileChecker;
FileManager myFileManager;

try {
    // get data file names
    myFileManager.getDirFiles(sourceDirectory, fileList);

    cout << "Copied files:" << endl;
    lstItr = fileList.begin();
    while(lstItr != fileList.end()) {
        if(myFileChecker.copyValidFile(*lstItr, destinationDirectory)) {
            cout << '\t' << *lstItr << endl;
            ++validFileCount;
        }
        ++lstItr;
    }
}

catch(Exception exception) {
    cout << "Error: " << exception.getDescription();
}

// how many valid files found?
cout << "\nCopied " << validFileCount << " file(s)." << endl;

// done
return EXIT_SUCCESS;
}
```

gethf.cpp

Extracts header and footer common to specified pair of files, saving them to header and footer files respectively.

```
#include <iostream>
#include "AffixManager.H"
#include "FileManager.H"
#include "Exception.H"

using namespace std;

int main(int argc, char *argv[])
{

    AffixManager myAffixManager;
    ofstream headerStream;
    ofstream footerStream;
    string headerString;
    string footerString;
    FileManager myFileManager;

    // proper number of command line parameters?
    if(argc != 5) {
        cout << "Usage:" << endl;
        cout << "\tgethf <file1> <file2> <headerFile> <footerFile>" << endl;
        return EXIT_FAILURE;
    }

    try {
        // get header and footer strings
        myAffixManager.getHeader(argv[1], argv[2], headerString);
        myAffixManager.getFooter(argv[1], argv[2], footerString);

        // send to output files
        myFileManager.outputOpenBinary(argv[3], headerStream);
        headerStream << headerString;

        myFileManager.outputOpenBinary(argv[4], footerStream);
        footerStream << footerString;

        // close streams
        headerStream.close();
        footerStream.close();
    }

    catch(Exception exp) {
        cout << exp.getDescription();
    }
}
```

```
// send to output files
return EXIT_SUCCESS;
}

comparehf.cpp
Compares headers and footers from two different pairs of files.

#include <iostream>
#include <stdlib.h>
#include "AffixManager.H"
#include "Exception.H"

using namespace std;

int main(int argc, char *argv[])
{
    // correct number of command line parameters?
    if(argc != 5) {
        cout << "\nUsage:" << endl;
        cout << "\tcomparehf <file1> <file2> <file3> <file4>" << endl;
        return EXIT_FAILURE;
    }

    string firstFile = argv[1];
    string secondFile = argv[2];
    string thirdFile = argv[3];
    string fourthFile = argv[4];
    string firstFooter, secondFooter;
    string firstHeader, secondHeader;
    AffixManager myAffixManager;

    try {
        // get headers
        myAffixManager.getHeader(firstFile, secondFile, firstHeader);
        myAffixManager.getHeader(thirdFile, fourthFile, secondHeader);

        // get footers
        myAffixManager.getFooter(firstFile, secondFile, firstFooter);
        myAffixManager.getFooter(thirdFile, fourthFile, secondFooter);

        // and now compare ...
        if(!firstHeader.compare(secondHeader)) // ... headers
            cout << "\nHeaders are different!" << endl;
    }
}
```

```

        else
            cout << "\nHeaders are identical." << endl;

            if(!firstFooter.compare(secondFooter)) // ...and footers
                cout << "\nFooters are different!" << endl;
            else
                cout << "\nFooters are identical." << endl;
        }

        catch(Exception exp) {
            cout << exp.getDescription();
            cout << ": " << exp.getMethodName();
        }

        return EXIT_SUCCESS;
    }

```

stripd.cpp

Removes header and footer from all files in specified directory,
saving stripped files to another directory.

```

#include <list>
#include <iostream>
#include "FileManager.H"
#include "Exception.H"
#include "AffixManager.H"

using namespace std;

int main(int argc, char *argv[])
{
    // correct number of command line arguments?
    if(argc != 5) {
        cout << "Usage:" << endl;
        cout << "\tstripd <inDirectory> <headerFile> ";
        cout << "<footerFile> <outDirectory>" << endl;
        return EXIT_FAILURE;
    }

    string inDir = argv[1];
    string headerFile = argv[2];
    string footerFile = argv[3];
    string outDir = argv[4];
    string inFile, outFile;

```



```

int last, fileCount = 0;

try {
// get list of files
list<string> inFileList;
FileManager myFileManager;
myFileManager.getDirFiles(argv[1], inFileList);

// for each file, call stripfile
AffixManager myAffixManager;
list<string>::iterator itrInFileList;
itrInFileList = inFileList.begin();
while(itrInFileList != inFileList.end()) {
    inFile = *itrInFileList;
    last = inFile.length() - 1;
    // extract filename
    outFile.clear();
    for(int index = 0; index < inFile.length(); ++index) {
        if(inFile[last-index] == '\\') break;
        outFile = inFile[last - index] + outFile;
    }
    outFile = outDir + '\\\' + outFile;
    cout << '\\"' << inDir << '\\\' << inFile << '\\"';
    cout << " stripped to " << '\\\' << outFile << '\\"' << endl;
myAffixManager.stripFile(*itrInFileList,headerFile,footerFile,outFile);
++itrInFileList;
++fileCount;
}

}

catch(Exception exp) {
    cout << exp.getDescription();
    cout << " in ";
    cout << exp.getMethodName();
}

cout << "\n" << fileCount << " file(s) processed." << endl;

return EXIT_SUCCESS;
}

```

stripf.cpp

Same as stripd.cpp, but only targets individual files.

```
#include <iostream>
#include "AffixManager.H"
#include "Exception.H"

using namespace std;

int main(int argc, char *argv[])
{
    // correct number of arguments?
    if(argc != 5) {
        cout << "Usage:" << endl;
        cout << "\tstripf <infile> <headerFile> ";
        cout << "<footerFile> <outfile>" << endl;
        return EXIT_FAILURE;
    }

    AffixManager myAffixManager;
    try {
        myAffixManager.stripFile(argv[1], argv[2], argv[3], argv[4]);
    }

    catch(Exception exp) {
        cout << exp.getDescription();
    }

    return EXIT_SUCCESS;
}
```

annod.cpp

Assigns IDs to all stories in all files in a given directory.

```
#include <iostream>
#include <string>
#include "Exception.H"
#include "StoryManager.H"
#include "FileManager.H"

using namespace std;
```

```
int main(int argc, char *argv[])
{
    // correct number of command line parameters/?
    if(argc != 3) {
        cout << "Usage:" << endl;
        cout << "\tannod <inDirectory> <outDirectory>" << endl;
        return EXIT_FAILURE;
    }

    string inDir = argv[1];
    string outDir = argv[2];
    string inFile, outFile;
    StoryManager myStoryManager;
    int last, fileCount = 0;

    try {
        // get list of files
        list<string> inFileList;
        FileManager myFileManager;
        myFileManager.getDirFiles(inDir, inFileList);

        // for each file, call stripfile
        StoryManager myStoryManager;
        list<string>::iterator itrInFileList;
        itrInFileList = inFileList.begin();
        while(itrInFileList != inFileList.end()) {

            inFile = *itrInFileList; // input file name

            // output file name
            myFileManager.extractFileName(inFile, outFile);
            outFile = outDir + '\\\' + outFile;
            myStoryManager.annotate(inFile, outFile); // annotate

            // progress report!
            cout << '\\\' << inDir << '\\\' << inFile << '\\\'';
            cout << " annotated to " << '\\\' << outFile << '\\\'' << endl;

            ++itrInFileList; // next file please!
            ++fileCount;
        }

        catch(Exception exp) {
```

```
        cout << exp.getDescription();
        cout << " in ";
        cout << exp.getMethodName();
    }

    cout << "\n" << fileCount << " file(s) processed." << endl;

    return EXIT_SUCCESS;
}
```

annof.cpp

Assigns IDs to all stories in a single file.

```
#include <iostream>
#include <stdlib.h>
#include "Exception.H"
#include "StoryManager.H"

using namespace std;

int main(int argc, char *argv[])
{
    // correct number of command line parameters?
    if(argc != 3) {
        cout << "\nUsage:" << endl;
        cout << "\tannof <infile> <outfile>" << endl;
        return EXIT_SUCCESS;
    }

    string inFile = argv[1];
    string outFile = argv[2];
    StoryManager myStoryManager;
    try {
        myStoryManager.annotate(inFile, outFile);
    }
    catch(Exception exp) {
        cout << exp.getDescription();
        cout << ": " << exp.getMethodName();
    }

    return EXIT_SUCCESS;
}

remhl.cpp
```

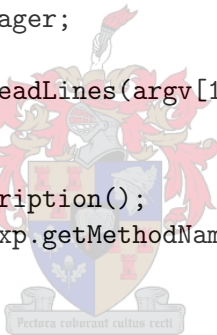

Removes story headlines, saving them in the process.

```
#include <iostream>
#include "StoryManager.H"

using namespace std;

int main(int argc, char *argv[])
{
    // correct number of command line parameters?
    if(argc != 4) {
        cout << "\nUsage:" << endl;
        cout << "remhl <infile> <headLineFile> <storyFile>" << endl;
        return EXIT_FAILURE;
    }

    // remove headlines
    StoryManager myStoryManager;
    try {
        myStoryManager.getHeadLines(argv[1], argv[2], argv[3]);
    }
    catch(Exception exp) {
        cout << exp.getDescription();
        cout << " in " << exp.getMethodName();
    }
}
```



fmerge.cpp

Merges all files in specified directory to create one corpus file.

```
#include <iostream>
#include <list>
#include "FileManager.H"
#include "Exception.H"
#include "StoryManager.H"

using namespace std;

int main(int argc, char *argv[])
{
    // correct number of command line parameters?
    if(argc != 3) {
        cout << "\nUsage:" << endl;
    }
}
```

```

        cout << "\tfmerge <inDirectory> <outFile>" << endl;
        cout << "\nPlease note that if <outFile> already exists, "
            << "other files will\nsimply be appended to it; otherwise "
            << "it will be created." << endl;
        return EXIT_FAILURE;
    }

    list<string> mergeFileList;
    list<string>::iterator lstItr;
    FileManager myFileManager;
    StoryManager myStoryManager;

    try {
        // get files to merge
        myFileManager.getDirFiles(argv[1], mergeFileList);
        lstItr = mergeFileList.begin();

        // proceed to merge files
        while(lstItr != mergeFileList.end()) {
            cout << "Appending: \"" << *lstItr
                << "\" to \"" << argv[2] << "\" " << endl;
            myStoryManager.mergeDayFiles(argv[2], *lstItr);
            ++lstItr;
        }
    }

    catch(Exception exp) {
        cout << exp.getDescription() << ": ";
        cout << exp.getMethodName();
    }

    return EXIT_SUCCESS;
}

```

ForCluto.cpp

Prepares text corpus for processing with CLUTO's doc2mat utility.

```

#include <iostream>
#include "Exception.H"
#include "StoryManager.H"

using namespace std;

```

```
int main(int argc, char *argv[])
{
    // correct number of command line arguments?
    if(argc != 3) {
        cout << "Usage:" << endl;
        cout << "\tForCluto <infile> <outfile>" << endl;
        return EXIT_FAILURE;
    }

    StoryManager myStoryManager;
    try {
        myStoryManager.prepareForCluto(argv[1], argv[2]);
    }

    catch(Exception exp) {
        cout << exp.getDescription();
        cout << ": " << exp.getMethodName();
    }

    return EXIT_SUCCESS;
}
```

lineB.cpp

Inserts line breaks as appropriate to ensure that each story ID is on a line by itself.

```
#include <iostream>
#include "Exception.H"
#include "StoryManager.H"

using namespace std;

int main(int argc, char *argv[])
{
    // correct number of command line parameters?
    if(argc != 3) {
        cout << "\nUsage:" << endl;
        cout << "\tlineB <infile> <outFile>" << endl;
        return EXIT_SUCCESS;
    }

    StoryManager myStoryManager;
    try {
```

```
        myStoryManager.insertLineBreaks(argv[1], argv[2]);
    }

    catch (Exception exp){
        cout << exp.getDescription();
        cout << "in: " << exp.getMethodName();
    }

    return EXIT_SUCCESS;
}
```

HEADER FILES

exception.h

```
// Exception Class
// METHODS:
// string getDescription()
// returns exception description
// return not defined if description not set
//
// string getMethodName()
// returns method reporting the exception
// return not defined if method name not set
// please note that reporting method may be different from
// originating method!!!
//
// void setDescription(const string& expDes)
// sets the exception description to expDes
//
// void setMethodName(const string& name)
// sets reporting method name
//

#ifndef __Exception__
#define __Exception__

#include <string>
using namespace std;

class Exception {
public:
```

```
        // accessors
        string getDescription();
        string getMethodName();

        // mutators
        void setDescription(const string& expDes);
        void setMethodName(const string& name);

    private:
        string expDescription;
        string methodName;
};

////////////////////////////////////
// Method Implementations

string Exception::getDescription()
{
    return expDescription;
}

string Exception::getMethodName()
{
    return methodName;
}

void Exception::setDescription(const string& expDes)
{
    expDescription = expDes;
}

void Exception::setMethodName(const string& name)
{
    methodName = name;
}

#endif

filechecker.h

// FileChecker Class
// METHODS:
// bool isValidDataFile(const string& fileName)
```

```

// returns true if file is a valid data file
// file valid => file does not contain "No such file or directory"
//
// bool copyValidFile(const string& inFile, const string& outDir)
// if file is valid, it is copied to specified directory
// returns true if file has been copied
//
#ifndef __FileChecker__
#define __FileChecker__

#include <string>
#include <fstream>
#include <vector>
#include <algorithm>
#include "Exception.H"
#include "FileManager.H"

using namespace std;

class FileChecker {
public:
FileChecker();

    bool isValidDataFile(const string& fileName);
    bool copyValidFile(const string& inFile,
                       const string& outDir);

private:
    string fileValidity;
};

////////////////////////////////////
// Method Impmentations

FileChecker::FileChecker()
{
    fileValidity = "No such file or directory";
}

bool FileChecker::isValidDataFile(const string& fileName)
{
    // File is considered valid data file if it does not
    // contain the string "No such file or directory"

```

```
    ifstream inStream; // input data file
    FileManager myFileManager;

    try {
        // open input file stream
myFileManager.inputOpenBinary(fileName, inStream);
    }

catch(Exception exception) { // update exception and pass on
exception.setMethodName("FileChecker.isValidDataFile()");
    throw(exception);
}

// read file contents into string lineData
char ch; // to hold each character as it is
        // read from input stream
vector<char> fileVector;
inStream.get(ch);
while(inStream) {
    fileVector.push_back(ch);
    inStream.get(ch);
}
// done with file
inStream.close();

// now search for "No such file or directory"
vector<char>::iterator p;
    p = search(fileVector.begin(), fileVector.end(),
        fileValidity.begin(), fileValidity.end());

if(p == fileVector.end()) // not found
return true;
else // found
return false;
}

bool FileChecker::copyValidFile(const string& inFile,
                                const string& outDir)
{
    FileManager myFileManager;
    string inputFile, outputFile;
    string::iterator strItr;
    bool hasCopied(false);
```

```

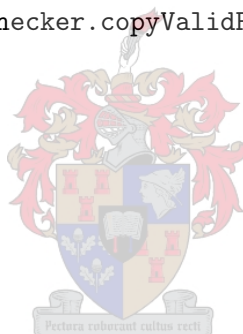
        try {
            inputFile = inFile;
            if(isValidDataFile(inputFile)) {
                // filter out file name from input path
                outputFile.clear();
                strItr = inputFile.end();
                while(strItr != inputFile.begin()) {
                    --strItr;
                    outputFile = *strItr + outputFile;
                    if(*strItr == '\\') break;
                }
                outputFile = outDir + outputFile;
                myFileManager.copyFile(inputFile, outputFile);
                hasCopied = true;
            }
        }
        catch(Exception exp) { // update exception and pass on
            exp.setMethodName("FileChecker.copyValidFile()");
            throw(exp);
        }
        return hasCopied;
    }

#endif

filemanager.h

// FileManager Class
// METHODS:
// void inputOpenBinary(const string& fileName, ifstream& streamName)
// opens file for reading in binary mode; caller must close stream.
//
// void outputOpenBinary(const string& fileName, ofstream& streamName)
// opens file for output in binary mode; caller must close stream.
//
// void appendOpenBinary(const string& fileName, ofstream& streamName)
// opens file for append in binary mode; caller must close stream.
//
// void getDirFiles(const string& inDir, list<string>& fileList)
// fills fileList with names of all files in inDir
//
// void copyFile(const string& inFile, const string& outFile)
// copies inFile to outFile
//

```




```

#ifndef __FileManager__
#define __FileManager__

#include <list>
#include <string>
#include <fstream>
#include <dirent.h>
#include <dir.h>
#include <errno.h>
#include "Exception.H"

using namespace std;

class FileManager {
public:
    // all methods throw exception on failure

    void inputOpenBinary(const string& fileName,
                        ifstream& streamName);

    void outputOpenBinary(const string& fileName,
                        ofstream& streamName);

    void appendOpenBinary(const string& fileName,
                        ofstream& streamName);

    void extractFileName(const string& fullPath,
                        string& fileName);

    void getDirFiles(const string& inDir, list<string>& fileList);
    void copyFile(const string& inFile, const string& outFile);
};

////////////////////////////////////
// Method Implementations

void FileManager::inputOpenBinary(const string& fileName,
                                ifstream& streamName)
{
    Exception exp; // throw exception upon failure

    streamName.open(fileName.c_str(), ios::in | ios::binary);
    if(!streamName) { // failed to open input file

```

```
exp.setDescription("Error opening file <" + fileName + ">.");
exp.setMethodName("FileManager.inputOpenBinary()");
throw(exp);
}

return;
}

void FileManager::outputOpenBinary(const string& fileName,
                                   ostream& streamName)
{
Exception exp; // throw exception upon failure

streamName.open(fileName.c_str(), ios::out | ios::binary);
if(!streamName) { // failed to open output file
exp.setDescription("Error opening file <" + fileName + ">.");
exp.setMethodName("FileManager.outputOpenBinary()");
throw(exp);
}
}

void FileManager::appendOpenBinary(const string& fileName,
                                   ostream& streamName)
{
Exception exp; // throw exception upon failure

streamName.open(fileName.c_str(), ios::app | ios::binary | ios::ate);
if(!streamName) { // failed to open output file
exp.setDescription("Error opening file <" + fileName + ">.");
exp.setMethodName("FileManager.appendOpenBinary()");
throw(exp);
}
}

void FileManager::extractFileName(const string& fullPath,
                                  string& fileName)
{
string localFullPath = fullPath;
string::iterator strItr = localFullPath.end();
string localFileName;

--strItr;
while(strItr != localFullPath.begin()) {
    if(*strItr == '\\') break;
```

```

        localFileName = *strItr + localFileName;
        --strItr;
    }

    fileName = localFileName;
}

void FileManager::getDirFiles(const string& inDir,
                             list<string>& fileList)
{
    DIR *pInDir;
    struct dirent *pent;
    int dotCount = 0;
    string localInDir;
    Exception exp;
    string inFile;

    // can we open source directory?
    localInDir = inDir;
    pInDir = opendir(localInDir.c_str());
    if(!pInDir) { // failed to open source directory
        exp.setDescription("Error opening input directory <"
                           + localInDir + ">.");
        exp.setMethodName("FileManager.getDirFiles()");
        throw(exp);
    }

    errno = 0; // change in errno indicates an
              // error reading files/directories
    while((pent = readdir(pInDir))) {
        if(dotCount++ < 2) continue; // if "." and ".." skip
        if(errno) { // errno has changed => error has occurred
            string des = "Undocument error reading directory <"
                + localInDir + ">.";
            exp.setDescription(des);
            exp.setMethodName("FileManager.getDirFiles()");
            throw(exp);
        }

        string::iterator strItr;
        strItr = localInDir.end();
        --strItr;

        if(*strItr != '\\')

```

```
        inFile = localInDir + "\\\" + pent -> d_name;
    else
        inFile = localInDir + pent -> d_name;

    fileList.push_back(inFile);
}

}

void FileManager::copyFile(const string& inFile,
                           const string& outFile)
{
    ifstream inStream;
    ofstream outStream;

    try {
        // open inStream for binary reading
        inputOpenBinary(inFile, inStream);

        // open outStream for binary writing
        outputOpenBinary(outFile, outStream);
    }
    catch(Exception exp) {
        exp.setMethodName("FileManager.copyFile()");
        throw(exp);
    }

    char ch; // to hold each character as
              // it is read from input stream
    inStream.get(ch);
    while(inStream) {
        outStream << ch;
        inStream.get(ch);
    }

    outStream.close();
    inStream.close();
}

#endif
```

```
affixmanager.h

#ifndef __AffixManager__
#define __AffixManager__

#include <string>
#include <fstream>
#include <deque>

#include "FileManager.H"
#include "Exception.H"

using namespace std;

class AffixManager {
public:
void getHeader(const string& firstFile,
               const string& secondFile,
               string& header);

void getFooter(const string& firstFile,
               const string& secondFile,
               string& footer);

bool stripHeader(deque<char>& fileDeque,
                 deque<char>& headerDeque);

bool stripFooter(deque<char>& fileDeque,
                 deque<char>& footerDeque);

void stripFile(const string& inFile,
               const string& headerFile,
               const string& footerFile,
               const string& outFile);

private:
void readFileIntoString(const string& fileName,
                       string& fileString);

void readFileIntoDeque(const string& fileName,
                       deque<char> & fileDeque);

void extractHeader(const string& first,
                  const string& second,
```

```
        string& header);

void extractFooter(const string& first,
                  const string& second,
                  string& footer);

};

////////////////////////////////////
// Method Implementations

void AffixManager::getHeader(const string& firstFile,
                             const string& secondFile,
                             string& header)
{
    string firstString, secondString;

    try {
        // first read files into strings
        readFileIntoString(firstFile, firstString);
        readFileIntoString(secondFile, secondString);

        // now get the header
        extractHeader(firstString, secondString, header);
    }

    catch(Exception exp){
        exp.setMethodName("AffixManager.getHeader()");
        throw(exp);
    }
}

void AffixManager::getFooter(const string& firstFile,
                             const string& secondFile,
                             string& footer)
{
    string firstString, secondString;

    try {
        // first read files into strings
        readFileIntoString(firstFile, firstString);
        readFileIntoString(secondFile, secondString);

        // now get the header
```

```
    extractFooter(firstString, secondString, footer);
}

    catch(Exception exp){
        exp.setMethodName("AffixManager.getFooter()");
        throw(exp);
    }
}

void AffixManager::readFileIntoString(const string& fileName,
                                     string& fileString)
{
    ifstream inStream;
    FileManager myFileManager;

    try {
        myFileManager.inputOpenBinary(fileName, inStream);
    }

    catch(Exception exp) {
        exp.setMethodName("AffixManager.readFileIntoString()");
        throw exp;
    }

    char ch;
    fileString.clear();
    inStream.get(ch);
    while(inStream) {
        fileString.append(1, ch);
        inStream.get(ch);
    }

    // done!
}

void AffixManager::readFileIntoDeque(const string& fileName,
                                     deque<char> & fileDeque)
{
    ifstream inStream;
    FileManager myFileManager;

    try {
        myFileManager.inputOpenBinary(fileName, inStream);
    }
```



```
catch(Exception exp) {
exp.setMethodName("AffixManager.readFileIntoDeque()");
throw exp;
}

char ch;
fileDeque.clear();
inStream.get(ch);
while(inStream) {
fileDeque.push_back(ch);
inStream.get(ch);
}

// done!

}

void AffixManager::extractHeader(const string& first,
                                const string& second,
                                string& header)
{
string::const_iterator s1, s2;
string tempString;

s1 = first.begin();
s2 = second.begin();
while(s1 != first.end() && s2 != second.end()) {
if(*s1 == *s2) {
tempString.push_back(*s1); // build footer string
}
else
break; // at first mismatch go home!

++s1; ++s2;
}

// return information to calling program
header = tempString;
}

void AffixManager::extractFooter(const string& first,
                                 const string& second,
                                 string& footer)
```



```
{
    string tempString;

    // compare strings backwards to determine footer
    string::const_iterator s1;
    string::const_iterator s2;
    int charCount = 0;

    s1 = first.end();
    s2 = second.end();
    while(s1 != first.begin() && s2 != second.begin()) {
        s1--; s2--;
        if(*s1 == *s2) {
            tempString = *s1 + tempString;
            charCount++;
        }
        else break;
    }

    // return information to calling module
    footer = tempString;
}

void AffixManager::stripFile(const string& inFile,
                             const string& headerFile,
                             const string& footerFile,
                             const string& outFile)
{
    try {
        // read streams into appropriate dequeues
        deque<char> fileDeque, headerDeque, footerDeque;

        readFileIntoDeque(inFile, fileDeque);
        readFileIntoDeque(headerFile, headerDeque);
        readFileIntoDeque(footerFile, footerDeque);

        // now strip
        stripHeader(fileDeque, headerDeque);
        stripFooter(fileDeque, footerDeque);

        // send stripped inString to output file
        ofstream outputStream;
        FileManager myFileManager;
```

```

        myFileManager.outputOpenBinary(outFile, outputStream);
int lenDeque = fileDeque.size();
for(int index = 0; index < lenDeque; ++index)
    outputStream << fileDeque[index];
//        outputStream << inString; // send string to stream

        outputStream.close(); // close stream
    }

    catch(Exception exp) { // update exception and rethrow
exp.setMethodName("AffixManager.stripFile()");
throw exp;
}
}

bool AffixManager::stripHeader(deque<char>& fileDeque,
                               deque<char>& headerDeque)
{
    bool wasStripped(false);
    Exception exp;

    int counter = 0;
    while(fileDeque[0] == headerDeque[counter]){
fileDeque.pop_front();
        ++counter;
        wasStripped = true;
    }

    return wasStripped;
}

bool AffixManager::stripFooter(deque<char>& fileDeque,
                                deque<char>& footerDeque)
{
    bool wasStripped(false);
    Exception exp;

    int last;
    int counter = footerDeque.size() - 1;
    while(counter > 0) {
        last = fileDeque.size() - 1;
        if(fileDeque[last] == footerDeque[counter]) {
            wasStripped = true;
            fileDeque.pop_back();
        }
    }
}

```

```
    }

/*     else {
        exp.setDescription("Footer mismatch.");
        exp.setMethodName("AffixManager::stripFooter()");
        throw(exp);
    } */
    --counter;
}

return true;
}

/*
bool AffixManager::stripFooter(string& fileString,
                               string& footerString)
{
bool wasStripped(false);

while(fileString.rbegin() == footerString.rbegin()) {
    wasStripped = true;

    fileString.erase(fileString.length() - 1, 1);
    footerString.erase(footerString.length() - 1, 1);

    if(footerString.length() == 0) break;
}

return wasStripped;
}
*/
#endif
```

storymanager.h

```
#ifndef __StoryManager__
#define __StoryManager__

#include "Exception.H"
#include "FileManager.H"

class StoryManager {
public:
```

```

        void insertLineBreaks(const string& inFile,
                               const string& outFile);

        void mergeDayFiles(const string& firstFile,
                            const string& secondFile);

        void annotate(const string& inFile,
                      const string& outFile);

        void getHeadLines(const string& inFile,
                           const string& headLineFile,
                           const string& storyFile);

        void prepareForCluto(const string& inFile,
                              const string& outFile);
private:
};

void StoryManager::insertLineBreaks(const string& inFile,
                                     const string& outFile)
{
    ifstream inStream;
    ofstream outStream;
    FileManager myFileManager;

    try {
        myFileManager.inputOpenBinary(inFile, inStream);
        myFileManager.outputOpenBinary(outFile, outStream);

        char ch;
        inStream.get(ch);
        while(inStream) {
            if(ch == '{') {
                outStream << "\n\n";
                outStream << ch;
            }
            else if(ch == '}') {
                outStream << ch;
                outStream << "\n\n";
            }
            else {
                outStream << ch;
            }
        }
    }
}

```

```
        inStream.get(ch);
    }
}

catch(Exception exp){
    exp.setMethodName("StoryManager.insertLineBreaks()");
    throw(exp);
}

// close file streams
inStream.close();
outStream.close();
}

void StoryManager::getHeadLines(const string& inFile,
                                const string& headLineFile,
                                const string& storyFile)
{
    FileManager myFileManager;
    ofstream hlStream; // headline file
    ofstream stStream; // story file - minus the head lines
    ifstream inStream;
    string lineData;

    try {
        myFileManager.inputOpenBinary(inFile, inStream);
        myFileManager.outputOpenBinary(headLineFile, hlStream);
        myFileManager.outputOpenBinary(storyFile, stStream);

        getline(inStream, lineData);
        while(!inStream.eof()) {
            if(lineData.substr(0,3) == "{HL}")
                // send to headline file
                hlStream << lineData << '\n';
            else
                // send to story file
                stStream << lineData << '\n';

            getline(inStream, lineData);
        }
        stStream << lineData << '\n';
    }
}
```

```
        catch(Exception exp){
            exp.setMethodName("StoryManager.getHeadLines()");
            throw(exp);
        }

        // close streams
        hlStream.close();
        stStream.close();
        inStream.close();
    }

void StoryManager::mergeDayFiles(const string& firstFile,
                                const string& secondFile)
{
    // secondFile is appended to firstFile, firstFile is now larger!
    FileManager myFileManager;
    ofstream outStream;
    ifstream inStream;

    try {
        myFileManager.appendOpenBinary(firstFile, outStream);
        myFileManager.inputOpenBinary(secondFile, inStream);

        char ch = inStream.get();
        outStream << '\n';
        while(!inStream.eof()) {
            outStream << ch;
            ch = inStream.get();
        }
    }
    catch(Exception exp) {
        exp.setMethodName("StoryManager.mergeDayFiles()");
        throw(exp);
    }

    outStream.close();
    inStream.close();
}

void StoryManager::annotate(const string& inFile,
                            const string& outFile)
{
    ifstream inStream;
    ofstream outStream;
```

```
FileManager myFileManager;

// open files
try {
    myFileManager.inputOpenBinary(inFile, inStream);
    myFileManager.outputOpenBinary(outFile, outStream);
}
catch(Exception exp) {
    exp.setMethodName("StoryManager.annotate()");
    throw(exp); // pass exception to calling module
}

// headline is assumed to be marked by:
// <TR><TD ALIGN="left" WIDTH="426"
string hString = "<TR><TD ALIGN=\"left\" WIDTH=\"426\"";
string fileName;
myFileManager.extractFileName(inFile, fileName);

string lineData;
//string strBaseID = inFile.substr(0,8);
unsigned long baseID =
    (atol((fileName.substr(0,8)).c_str()))
    * 100;//atol(strBaseID.c_str()) * 100;

// first line in inFile is always a head line!
getline(inStream, lineData);
outStream << "{HL:" << ++baseID << "}";
outStream << lineData;

// now for the rest of the file
getline(inStream, lineData);
while(!inStream.eof()) {
    if(!lineData.compare(0,
        hString.length(),
        hString)) { // header line found

        // insert story ID
        outStream << "\n\n{ID:" << baseID << "}\n\n";

        // insert headline marker
        outStream << "{HL:" << ++baseID << "}";
    }
    outStream << lineData << '\n'; // send to output file
    getline(inStream, lineData);
}
```

```

    }

    // append ID of last story
    outputStream << "\n\n{ID:" << baseID << "}";

    // close files
    inputStream.close();
    outputStream.close();
}

void StoryManager::prepareForCluto(const string& inFile,
    const string& outFile)
{
    ifstream inputStream;
    ofstream outputStream;
    FileManager myFileManager;

    // open files
    try {
        myFileManager.inputOpenBinary(inFile, inputStream);
        myFileManager.outputOpenBinary(outFile, outputStream);
    }
    catch(Exception exp) {
        exp.setMethodName("StoryManager.prepareForCluto()");
        throw(exp); // pass exception to calling module
    }

    int outputID;
    string storyText = "";
    string lineData;
    getline(inputStream, lineData);
    while(!inputStream.eof()) {
        lineData = lineData.substr(0, lineData.length() - 1);
        if(lineData.substr(0,3) != "{ID") {
            storyText = storyText + " " + lineData;
        }

        else {
            outputID = lineData.length();
            outputStream << lineData.substr(4, 10); // << ' ';
            outputStream << storyText;
            outputStream << '\n';
            storyText = "";
        }
    }
}

```



```
        getline(inStream, lineData);
    }
}

#endif

changeCase.h

#ifndef __ChangeCase__
#define __ChangeCase__

#include <string>
using namespace std;

class ChangeCase {
public:
    void lowerCase(string& target);
    void upperCase(string& target);
};


void ChangeCase::upperCase(string& target)
{
    string temp;
    string::iterator sItr;

    sItr = target.begin();
    while(sItr != target.end()){
        temp += toupper(*sItr);
        sItr++;
    }

    target = temp;
}

void ChangeCase::lowerCase(string& target)
{
    string temp;
    string::iterator sItr;

    sItr = target.begin();
    while(sItr != target.end()) {
        temp += tolower(*sItr);
        sItr++;
    }
}
```

A watermark of a university crest is visible in the center of the page. The crest features a shield with various symbols, including a cross and a book, and is topped with a crown. Below the shield is a motto scroll with the Latin text "Pectora roburant cultus recti".

```
    target = temp;  
}  
  
#endif
```



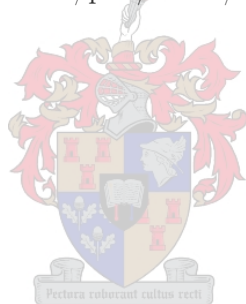
Appendix B

SMART Stop List

a a's able about above according accordingly across actually after afterwards
again against ain't all allow allows almost alone along already also although
always am among amongst an and another any anybody anyhow anyone any-
thing anyway anyways anywhere apart appear appreciate appropriate are aren't
around as aside ask asking associated at available away awfully b be became be-
cause become becomes becoming been before beforehand behind being believe
below beside besides best better between beyond both brief but by c c'mon
c's came can can't cannot cant cause causes certain certainly changes clearly
co com come comes concerning consequently consider considering contain con-
taining contains corresponding could couldn't course currently d definitely de-
scribed despite did didn't different do does doesn't doing don't done down
downwards during e each edu eg eight either else elsewhere enough entirely
especially et etc even ever every everybody everyone everything everywhere ex-
actly example except f far few fifth first five followed following follows for
former formerly forth four from further furthermore g get gets getting given
gives go goes going gone got gotten greetings h had hadn't happens hardly
has hasn't have haven't having he he's hello help hence her here here's here-
after hereby herein hereupon hers herself hi him himself his hither hopefully
how howbeit however I i'd i'll i'm i've ie if ignored immediate in inasmuch inc
indeed indicate indicated indicates inner insofar instead into inward is isn't it
it'd it'll it's its itself j just k keep keeps kept know knows known l last lately
later latter latterly least less lest let let's like liked likely little look looking
looks ltd m mainly many may maybe me mean meanwhile merely might more
moreover most mostly much must my myself n name namely nd near nearly
necessary need needs neither never nevertheless new next nine no nobody non
none noone nor normally not nothing novel now nowhere o obviously of off
often oh ok okay old on once one ones only onto or other others otherwise
ought our ours ourselves out outside over overall own p particular particularly
per perhaps placed please plus possible presumably probably provides q que
quite qv r rather rd re really reasonably regarding regardless regards relatively

respectively right s said same saw say saying says second secondly see seeing
seem seemed seeming seems seen self selves sensible sent serious seriously seven
several shall she should shouldn't since six so some somebody somehow some-
one something sometime sometimes somewhat somewhere soon sorry specified
specify specifying still sub such sup sure t t's take taken tell tends th than
thank thanks thanx that that's that's the their theirs them themselves then
thence there there's thereafter thereby therefore therein theres thereupon these
they they'd they'll they're they've think third this thorough thoroughly those
though three through throughout thru thus to together too took toward to-
wards tried tries truly try trying twice two u un under unfortunately unless
unlikely until unto up upon us use used useful uses using usually uucp v value
various very via viz vs w want wants was wasn't way we we'd we'll we're we've
welcome well went were weren't what what's whatever when whence when-
ever where where's whereafter whereas whereby wherein whereupon wherever
whether which while whither who who's whoever whole whom whose why will
willing wish with within without won't wonder would would wouldn't x y yes
yet you you'd you'll you're you've your yours yourself yourselves z zero

Source: <ftp://ftp.cs.cornell.edu/pub/smart/english.stop>; Accessed 23 August 2005.



Appendix C

Botswana Daily News Non-publishing Days

For the year 2004, Botswana Public Holidays were as follows:

1. New Years Day, 01 January
2. Public Holiday, 02 January
3. Good Friday, 09 April
4. Public Holiday, 10 April
5. Easter Monday, 12 April
6. Labour Day, 01 May
7. Ascension Day, 20 May
8. Sir Seretse Khama Day, 01 July
9. Presidents Day, 19 July
10. Public Holiday, 20 July
11. Botswana Day, 30 September
12. Public Holiday, 01 October
13. Christmas Day, 25 December
14. Boxing Day, 26 December
15. Public Holiday, 27 December



Officially, the *Botswana Daily News* does not publish during public holidays and weekends.

Appendix D

Direct K-Way Clustering 10-Cluster Solution

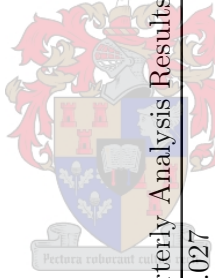


Cluster 0; Size: 109; ISim: 0.143; ESIm: 0.021								
Descriptive:	land	40.90%	plot	8.80%	alloc	8.70%	board	3.90%
Discriminating:	land	29.00%	plot	6.30%	alloc	6.00%	board	2.30%
Cluster 1; Size: 567; ISim: 0.102; ESIm: 0.021								
Descriptive:	parti	17.30%	bdp	12.10%	elect	6.50%	candid	5.20%
Discriminating:	parti	14.00%	bdp	10.40%	elect	4.50%	candid	4.30%
Cluster 2; Size: 250; ISim: 0.078; ESIm: 0.016								
Descriptive:	music	13.70%	album	11.20%	song	8.20%	artist	7.00%
Discriminating:	music	10.40%	album	8.70%	song	6.30%	artist	5.30%
Cluster 3; Size: 431; ISim: 0.068; ESIm: 0.015								
Descriptive:	team	12.00%	game	8.70%	sport	4.80%	zebra	2.80%
Discriminating:	team	8.50%	game	6.40%	sport	3.30%	zebra	2.10%
Cluster 4; Size: 277; ISim: 0.062; ESIm: 0.022								
Descriptive:	school	18.30%	student	13.10%	teacher	6.70%	educ	5.40%
Discriminating:	school	16.50%	student	13.50%	teacher	6.70%	educ	4.00%
Cluster 5; Size: 391; ISim: 0.055; ESIm: 0.016								
Descriptive:	polic	16.60%	court	7.50%	crime	2.70%	case	2.50%
Discriminating:	polic	12.90%	court	5.50%	suspect	2.00%	crime	1.90%
Cluster 6; Size: 464; ISim: 0.056; ESIm: 0.021								
Descriptive:	hiv	9.50%	aid	8.50%	7 church	5.20%	women	2.60%
Discriminating:	hiv	9.10%	aid	7.90%	church	5.70%	parti	2.10%
Cluster 7; Size: 474; ISim: 0.047; ESIm: 0.022								
Descriptive:	sadc	7.60%	region	3.10%	africa	2.20%	african	2.00%
Discriminating:	sadc	10.40%	region	3.30%	acp	2.20%	parti	1.90%
Cluster 8; Size: 621; ISim: 0.043; ESIm: 0.021								
Descriptive:	water	4.50%	resid	4.10%	road	4.00%	villag	2.90%
Discriminating:	water	4.90%	road	4.50%	resid	4.10%	villag	2.70%
Cluster 9; Size: 744; ISim: 0.038; ESIm: 0.021								
Descriptive:	product	2.30%	bank	2.20%	compani	2.00%	cent	1.70%
Discriminating:	parti	2.70%	product	2.50%	bank	2.40%	bdp	1.90%

Table D.1: Direct K-Way Clustering 10-Cluster Solution

Appendix E

Quarterly Analysis CLUTO output



Quarterly Analysis Results: Quarter 1

Cluster 0; Size: 24; ISim: 0.254; ESIm: 0.027								
Descriptive:	univers	52.20%	locat	3.40%	task	2.70%	forc	1.60%
Discriminating:	univers	36.50%	locat	2.20%	task	1.60%	parti	0.80%
Cluster 1; Size: 52; ISim: 0.135; ESIm: 0.018								
Descriptive:	music	15.00%	album	13.90%	song	6.60%	artist	5.50%
Discriminating:	music	10.30%	album	9.80%	song	4.60%	artist	3.80%
Cluster 2; Size: 88; ISim: 0.132; ESIm: 0.023								
Descriptive:	parti	23.20%	bdp	8.00%	elect	7.10%	candid	3.00%
Discriminating:	parti	17.80%	bdp	6.30%	elect	4.20%	opposit	2.10%
Cluster 3; Size: 45; ISim: 0.121; ESIm: 0.019								
Descriptive:	sport	15.80%	athlet	6.60%	olymp	3.70%	boxer	3.00%

Quarterly Analysis Results: Quarter 1

Descriptive:	million	8.60%	cent	5.00%	bank	4.20%	budget	3.10%
Discriminating:	million	7.40%	cent	4.30%	bank	3.90%	loan	2.50%
Cluster 12; Size: 107; ISim: 0.064; ESIm: 0.025								
Descriptive:	trade	4.10%	product	2.40%	compani	2.20%	tourism	2.20%
Discriminating:	trade	4.40%	tourism	2.20%	nkate	2.10%	product	2.00%
Cluster 13; Size: 131; ISim: 0.058; ESIm: 0.024								
Descriptive:	Hiv	5.30%	church	4.70%	women	4.60%	aid	4.40%
Discriminating:	church	5.10%	hiv	4.50%	women	4.40%	aid	3.70%
Cluster 14; Size: 97; ISim: 0.055; ESIm: 0.022								
Descriptive:	road	7.30%	water	6.40%	council	2.30%	question	1.90%
Discriminating:	road	6.00%	water	4.80%	question	1.60%	answer	1.50%

Table E.1: Quarterly Analysis Results: Quarter 1

Quarterly Analysis Results: Quarter 2

Cluster 0; Size: 38; ISim: 0.254; ESIm: 0.023								
Descriptive:	land	27.80%	plot	12.40%	alloc	11.10%	commiss	7.60%
Discriminating:	land	16.90%	plot	7.90%	alloc	7.00%	commiss	4.30%
Cluster 1; Size: 106; ISim: 0.143; ESIm: 0.023								
Descriptive:	parti	17.10%	bdp	16.10%	elect	6.90%	candid	5.80%
Discriminating:	bdp	12.70%	parti	12.40%	candid	4.40%	elect	4.30%
Cluster 2; Size: 44; ISim: 0.122; ESIm: 0.017								
Descriptive:	album	14.40%	music	11.10%	song	10.20%	track	3.80%
Discriminating:	album	10.20%	music	7.50%	song	7.00%	track	2.50%

Quarterly Analysis Results: Quarter 2

Cluster 3; Size: 57; ISim: 0.117; ESIm: 0.024									
Descriptive:	acp	25.80%	trade	2.70%	agreement	2.30%	moga	2.10%	
Discriminating:	acp	22.30%	trade	1.70%	merafh	1.50%	agreement	1.50%	
Cluster 4; Size: 59; ISim: 0.108; ESIm: 0.024									
Descriptive:	children	14.90%	polio	11.00%	school	9.90%	immunis	9.50%	
Discriminating:	children	11.60%	polio	9.60%	immunis	8.30%	school	6.30%	
Cluster 5; Size: 53; ISim: 0.098; ESIm: 0.018									
Descriptive:	polic	29.00%	suspect	4.50%	superintend	3.20%	crime	2.50%	
Discriminating:	polic	20.60%	suspect	3.20%	superintend	2.40%	crime	1.50%	
Cluster 6; Size: 110; ISim: 0.092; ESIm: 0.017									
Descriptive:	team	11.10%	game	8.10%	zebra	6.30%	cup	4.90%	
Discriminating:	team	7.50%	game	5.60%	zebra	4.60%	cup	3.60%	
Cluster 7; Size: 49; ISim: 0.094; ESIm: 0.019									
Descriptive:	court	17.60%	gupta	3.10%	accus	2.30%	elect	2.20%	
Discriminating:	court	12.50%	gupta	2.50%	magistr	1.50%	accus	1.30%	
Cluster 8; Size: 87; ISim: 0.086; ESIm: 0.025									
Descriptive:	hiv	12.00%	aid	10.40%	drug	2.30%	test	2.20%	
Discriminating:	hiv	9.90%	aid	8.30%	drug	2.00%	test	1.80%	
Cluster 9; Size: 54; ISim: 0.083; ESIm: 0.024									
Descriptive:	privatis	18.50%	tourism	4.90%	tsogwan	4.30%	compani	2.70%	
Discriminating:	privatis	17.80%	tsogwan	4.10%	tourism	4.10%	ceda	1.60%	
Cluster 10; Size: 63; ISim: 0.069; ESIm: 0.020									
Descriptive:	cultur	12.40%	pageant	6.20%	tradit	4.20%	art	2.60%	

Quarterly Analysis Results: Quarter 2

Discriminating:	cultur	10.30%	pageant	5.70%	tradit	3.30%	art	2.20%
Cluster 11; Size: 90; ISim: 0.072; ESIm: 0.025								
Descriptive:	council	6.40%	water	4.90%	resid	4.00%	district	3.80%
Discriminating:	council	4.60%	water	4.20%	resid	3.20%	settlement	3.20%
Cluster 12; Size: 68; ISim: 0.065; ESIm: 0.021								
Descriptive:	farmer	6.70%	agricultur	5.00%	road	4.70%	product	3.20%
Discriminating:	farmer	6.20%	agricultur	4.10%	road	3.30%	cattl	2.60%
Cluster 13; Size: 91; ISim: 0.068; ESIm: 0.025								
Descriptive:	bank	6.90%	sadc	6.80%	women	6.10%	region	3.00%
Discriminating:	sadc	7.10%	bank	7.10%	women	4.90%	region	2.50%
Cluster 14; Size: 104; ISim: 0.061; ESIm: 0.026								
Descriptive:	servic	3.20%	employe	2.80%	media	2.50%	public	2.40%
Discriminating:	employe	3.10%	media	2.30%	servic	2.20%	worker	1.80%

Table E.2: Quarterly Analysis Results: Quarter 2

Quarterly Analysis Results: Quarter 3

Cluster 0; Size: 36; ISim: 0.151; ESIm: 0.023								
Descriptive:	mine	12.40%	strike	10.30%	union	9.10%	worker	5.90%
Discriminating:	mine	8.80%	strike	7.60%	union	6.40%	worker	4.00%
Cluster 1; Size: 42; ISim: 0.136; ESIm: 0.020								
Descriptive:	athlet	15.60%	sport	12.70%	olymp	10.10%	game	2.90%
Discriminating:	athlet	11.60%	sport	8.50%	olymp	7.40%	boxer	1.80%
Cluster 2; Size: 45; ISim: 0.138; ESIm: 0.026								

Quarterly Analysis Results: Quarter 3

Descriptive:	land	31.10%	alloc	5.90%	board	4.40%	commiss	2.80%
Discriminating:	land	24.40%	alloc	4.60%	board	2.90%	plot	1.80%
Cluster 3; Size: 58; ISim: 0.127; ESIm: 0.017								
Descriptive:	music	17.50%	album	14.50%	artist	7.20%	song	7.20%
Discriminating:	music	11.70%	album	10.00%	song	4.90%	artist	4.80%
Cluster 4; Size: 61; ISim: 0.125; ESIm: 0.026								
Descriptive:	sadc	30.00%	region	7.80%	women	2.00%	mkapa	1.80%
Discriminating:	sadc	26.80%	region	5.70%	mkapa	1.60%	parti	1.60%
Cluster 5; Size: 156; ISim: 0.119; ESIm: 0.024								
Descriptive:	parti	17.90%	bdp	12.50%	candid	6.80%	elect	4.20%
Discriminating:	parti	14.20%	bdp	10.50%	candid	5.60%	elect	2.80%
Cluster 6; Size: 91; ISim: 0.104; ESIm: 0.018								
Descriptive:	team	12.80%	game	8.70%	footbal	4.70%	cup	3.80%
Discriminating:	team	8.20%	game	5.50%	footbal	3.30%	cup	2.70%
Cluster 7; Size: 64; ISim: 0.095; ESIm: 0.019								
Descriptive:	court	19.20%	magistr	6.20%	ckgr	4.10%	basarwa	3.30%
Discriminating:	court	13.50%	magistr	4.80%	ckgr	3.20%	basarwa	2.50%
Cluster 8; Size: 77; ISim: 0.071; ESIm: 0.019								
Descriptive:	Polic	19.10%	road	11.40%	crime	4.30%	superintend	2.50%
Discriminating:	polic	14.90%	road	8.30%	crime	3.00%	superintend	2.00%
Cluster 9; Size: 92; ISim: 0.073; ESIm: 0.024								
Descriptive:	villag	6.20%	water	5.90%	project	4.70%	council	3.40%
Discriminating:	villag	4.80%	water	4.70%	project	3.30%	disast	2.90%

Quarterly Analysis Results: Quarter 3

Cluster 10; Size: 117; ISim: 0.072; ESIm: 0.024										
Descriptive:	hiv	8.70%	aid	7.20%	church	6.40%	youth	5.10%		
Discriminating:	hiv	7.70%	church	6.60%	aid	6.20%	youth	4.20%		
Cluster 11; Size: 81; ISim: 0.066; ESIm: 0.023										
Descriptive:	product	6.70%	student	5.90%	farmer	5.10%	agricultur	4.60%		
Discriminating:	student	5.20%	product	5.20%	farmer	5.00%	agricultur	4.10%		
Cluster 12; Size: 54; ISim: 0.062; ESIm: 0.020										
Descriptive:	cultur	5.70%	torch	4.20%	kgosi	3.00%	british	2.30%		
Discriminating:	cultur	4.30%	torch	3.90%	kgosi	2.60%	ruth	2.10%		
Cluster 13; Size: 115; ISim: 0.062; ESIm: 0.025										
Descriptive:	bank	4.30%	compani	3.40%	busi	3.10%	diamond	3.10%		
Discriminating:	bank	4.20%	diamond	3.40%	compani	2.50%	busi	2.30%		
Cluster 14; Size: 79; ISim: 0.058; ESIm: 0.022										
Descriptive:	media	3.70%	environment	3.70%	conserv	2.80%	environ	2.50%		
Discriminating:	environment	3.90%	media	3.20%	conserv	3.10%	tourism	2.40%		

Table E.3: Quarterly Analysis Results: Quarter 3

Quarterly Analysis Results: Quarter 4

Cluster 0; Size: 43; ISim: 0.137; ESIm: 0.031										
Descriptive:	elect	15.60%	iec	7.60%	poll	7.40%	vote	6.50%		
Discriminating:	elect	9.60%	iec	7.00%	poll	6.60%	vote	3.10%		
Cluster 1; Size: 50; ISim: 0.123; ESIm: 0.018										
Descriptive:	song	12.30%	music	12.30%	artist	8.60%	album	8.40%		

Quarterly Analysis Results: Quarter 4

Discriminating:	song	8.80%	music	8.50%	album	6.00%	artist	5.90%
Cluster 2; Size: 136; ISim: 0.121; ESIm: 0.027								
Descriptive:	bdp	13.80%	parti	13.60%	candid	5.30%	vote	4.30%
Discriminating:	bdp	11.20%	parti	9.50%	candid	4.10%	bnf	3.40%
Cluster 3; Size: 79; ISim: 0.108; ESIm: 0.016								
Descriptive:	team	16.00%	game	8.40%	player	4.70%	leagu	2.80%
Discriminating:	team	10.00%	game	5.40%	player	3.20%	leagu	1.90%
Cluster 4; Size: 51; ISim: 0.102; ESIm: 0.018								
Descriptive:	court	11.50%	applic	3.90%	pilan	3.30%	accus	3.20%
Discriminating:	court	7.70%	applic	2.50%	pilan	2.30%	bail	2.00%
Cluster 5; Size: 71; ISim: 0.102; ESIm: 0.027								
Descriptive:	aid	15.70%	hiv	14.90%	test	2.70%	women	2.10%
Discriminating:	aid	12.80%	hiv	12.30%	parti	2.10%	test	2.10%
Cluster 6; Size: 45; ISim: 0.097; ESIm: 0.023								
Descriptive:	sport	26.70%	disast	6.70%	disabl	5.70%	health	2.70%
Discriminating:	sport	21.50%	disast	5.50%	disabl	4.40%	parti	1.80%
Cluster 7; Size: 58; ISim: 0.091; ESIm: 0.019								
Descriptive:	polic	22.40%	superintend	2.90%	crime	2.60%	suspect	2.40%
Discriminating:	polic	15.20%	superintend	2.10%	crime	1.60%	parti	1.60%
Cluster 8; Size: 48; ISim: 0.099; ESIm: 0.029								
Descriptive:	women	12.30%	mp	4.70%	parliament	4.20%	elect	3.10%
Discriminating:	women	9.90%	mp	4.30%	parliament	3.00%	nomin	1.80%
Cluster 9; Size: 90; ISim: 0.089; ESIm: 0.025								

Quarterly Analysis Results: Quarter 4

Descriptive:	school	18.00%	student	9.70%	educ	4.90%	teacher	4.30%
Discriminating:	school	14.60%	student	8.30%	teacher	3.80%	educ	3.10%
Cluster 10; Size: 58; ISim: 0.079; ESIm: 0.022								
Descriptive:	water	9.00%	agricultur	5.80%	ict	4.40%	farmer	4.20%
Discriminating:	water	6.60%	agricultur	4.30%	ict	3.90%	farmer	3.30%
Cluster 11; Size: 66; ISim: 0.085; ESIm: 0.033								
Descriptive:	govern	3.10%	minist	2.10%	parliament	2.00%	youth	2.00%
Discriminating:	bill	2.30%	bta	2.10%	amend	1.80%	parliament	1.50%
Cluster 12; Size: 50; ISim: 0.069; ESIm: 0.024								
Descriptive:	model	6.80%	africa	4.50%	african	4.00%	wast	2.70%
Discriminating:	model	6.90%	africa	2.90%	african	2.80%	wast	2.40%
Cluster 13; Size: 82; ISim: 0.063; ESIm: 0.023								
Descriptive:	cent	4.40%	tourism	3.80%	sadc	3.50%	bank	2.30%
Discriminating:	tourism	3.40%	cent	3.30%	sadc	2.90%	parti	2.00%
Cluster 14; Size: 78; ISim: 0.057; ESIm: 0.025								
Descriptive:	project	3.50%	council	2.80%	offic	2.70%	councillor	2.60%
Discriminating:	project	2.30%	councillor	2.20%	parti	2.00%	bdp	1.60%

Table E.4: 15 cluster bisecting k-way clustering solution