Unsupervised Classification of Text Documents

By

Roxana K. Aparicio Carrasco

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science
in
Scientific Computing

UNIVERSITY OF PUERTO RICO

MAYAGÜEZ CAMPUS

July, 2007

Approved by:

Edgar Acuña, Ph.D. President, Graduate Committee	Date
Alexander Urintsev, Ph.D. Member, Graduate Committee	Date
Ana C. González, M.Sc. Member, Graduate Committee	Date
William Hernández-Rivera, Ph.D. Representative of Graduate Studies	Date
Julio Quintana, Ph.D. Department Chairman	Date

Unsupervised Classification of Text Documents

Roxana K. Aparicio Carrasco

Abstract

The automatic extraction of knowledge from very large document collections is becoming an important issue in order to exploit the increasing available information stored in text form. A significant aspect of this extraction of knowledge consists in organize the collection into clusters of related documents; this task is known as unsupervised classification or clustering. As a result of preprocessing the collection using the vector space model, a vector representation of each document is obtained. The main characteristics of these vectors are their high dimensionality and sparsity. In this thesis we had studied and implemented algorithms for clustering large document collections, that fully exploit these characteristics. We propose a sparse representation of the document vectors stored in a relational database and developed SQL implementations of two different clustering algorithms: PAM and EM using Multinomial Naive Bayes Mixtures.

Resumen

La extracción automática de conocimiento de grandes colecciones de documentos se esta convirtiendo en un asunto cada vez mas importante con el fin de explotar la creciente información disponible en forma de texto. Un aspecto importante de esta extracción de conocimiento consiste en organizar la colección en grupos de documentos relacionados; esta tarea es conocida como clasificación no supervisada o análisis de conglomerados. Como resultado de preprocesar la información usando el modelo de espacio vectorial se obtiene un vector como representación de cada documento. Las características principales de estos vectores son su gran dimensión y esparcidad. En esta tesis estudiamos e implementamos algoritmos para clasificación no supervisada de grandes colecciones de documentos, que explotan estas características. Proponemos una representación esparcida de los vectores de documentos almacenada en una base de datos relacional y desarrollamos implementaciones en lenguaje SQL de dos distintos algoritmos: PAM y EM usando mezclas de distribución Naive Bayes multinomial.

To Karen Sofía, the person for whom I wish become a better person every day.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to specially thank my advisor, Dr. Edgar Acuña, without his expertise, guidance and support none of this would have been possible. Thank you for believing in me and motivating me every step of the way.

I would also like to thank my committee members: Alexander Urintsev and Ana Carmen Gonzalez, thank you for your time and support. Furthermore, I must thank the Mathematical Science Department at the University of Puerto Rico Mayagüez for providing the academic environment that has allowed me to grow as both a professional and person.

Thanks to Ollantay for his support and advice in this study; thanks to my parents Maria and Edgar for their example of effort and sacrifice and for always give me the best of them.

This work was funded by The Office of Naval Research (ONR) under grant number N0014-03-0359 and The Department of Defense under grant number N0014-06-1-0555.

Chapter 1

Introduction

An invaluable portion of significant data occurs naturally in text form. Text mining techniques have been widely used for knowledge discovery from text collections in many domains, such as scientific literature, business documents and web pages. The large amount of document collections makes manual attempts to organize and efficiently partitioning the collection into previously unseen categories at best extremely labor-intensive and at worst nearly impossible to achieve. As a widely recognized technique, clustering or unsupervised classification has proven to be very useful in detecting unknown object categories and revealing hidden correlations among objects. It has received a lot of attention in recent years.

By using a vector space model, text data can be treated as high-dimensional but sparse numerical data vectors. It is a contemporary challenge to efficiently preprocess and cluster very large document collections.

Effective and efficient document clustering algorithms play an important role in providing intuitive navigation and browsing mechanisms by categorizing large amounts of

information into a small number of meaningful clusters. Cluster analysis has been used to group related documents for browsing, to find genes and proteins that have similar functionality, or as a means of data compression (Steinbach, Ertoz and Kumar) [29]. While clustering has a long history and a large number of clustering techniques have been developed in statistics, pattern recognition, data mining, and other fields, significant challenges still remain.

A starting point for applying clustering algorithms to unstructured text data is to create a vector space model, alternatively known as a bag-of-words model. Typically, a large number of words exist in even a moderately sized set of documents where a few thousand words or more are common. Thus for large document collections, both the row and column dimensions of the matrix are quite large. However, this matrix is typically very sparse with a high percent of the matrix entries being zero (Dhillon, Fan and Guan) [7].

Using the vector space model various classical clustering algorithms such as the k-means algorithm and its variants, hierarchical agglomerative clustering, and graph-theoretic methods have been explored in the text mining literature [7].

On the other hand, among data miners, to perform clustering in high dimensional databases, has been receiving a lot of attention recently. Ordonez and Cereghini [22] present a SQL implementation of the EM algorithm to perform clustering in very large databases. This is presented in the context of data mining systems, where data is highly structured.

In this thesis we had studied and implemented preprocessing and clustering algorithms for large document collections, that fully exploit the high dimensionality and sparcity of text representation. We propose a sparse representation of the document vectors stored

in a relational database and developed SQL implementations of two different algorithms in this domain: PAM and EM using Multinomial Naive Bayes Mixtures.

This thesis is organized as follows: Chapter 2 introduces the text mining framework; Chapter 3 covers the theory related to the preprocessing task in text mining systems; Chapter 4 provides the theory for unsupervised classification for text mining and describes the selected algorithms implemented for this thesis; Chapter 5 presents the experimental results obtained with the described clustering techniques; Chapter 6 concludes with a discussion of the main results achieved in this research and an outline for proposed future work.

Chapter 2

Text Mining

Text mining can be defined as a knowledge-intensive process in which a user interacts with a document collection over time by using a suite of analysis tools. In a manner analogous to data mining, text mining seeks to extract useful information from data sources through the identification and exploration of interesting patterns. In the case of text mining, however, the data sources are document collections, and interesting patterns are found not among formalized database records but in the unstructured textual data in the documents in these collections (Feldman and Sanger) [10].

Text mining derives much of its inspiration and direction from seminal research on data mining. Therefore, it is not surprising to find that text mining and data mining systems evince many high-level architectural similarities. For instance, both types of systems rely on preprocessing routines, pattern-discovery algorithms, and presentation-layer elements such as visualization tools to enhance the browsing of answer sets. Further, text mining adopts many of the specific types of patterns in its core knowledge discovery operations that were

first introduced and vetted in data mining research [10].

2.1 General Architecture of Text Mining Systems

Text mining systems, do not run their knowledge discovery algorithms on unprepared document collections. Considerable emphasis in text mining is devoted to what are commonly referred to as preprocessing operations.

Text mining preprocessing operations include a variety of different types of techniques culled and adapted from information retrieval, information extraction, and computational linguistics research that transform raw, unstructured, original-format content into a carefully structured, intermediate data format. Knowledge discovery operations, in turn, are operated against this specially structured intermediate representation of the original document collection (Konchady) [17].

Figure 2.1 shows a model where data flows upwards with an application layer at the top . At the bottom layer, input documents are received. These are converted to unstructured text in the standardization level. The tokenization layer, breaks the stream of text into units called tokens. A token can be thought as a single unit of information. The tokens could be assembled into composite tokens based on a set of patterns and phrases from an external dictionary. We will explain the tokenization process later in Chapter 3 Section 3.3.

A set of functions in the next layer uses the tokens as input. We do not need the order of tokens to represent a document. The set of unique words occurring in a document or the collection weighted by their importance in the document is a reasonable representation

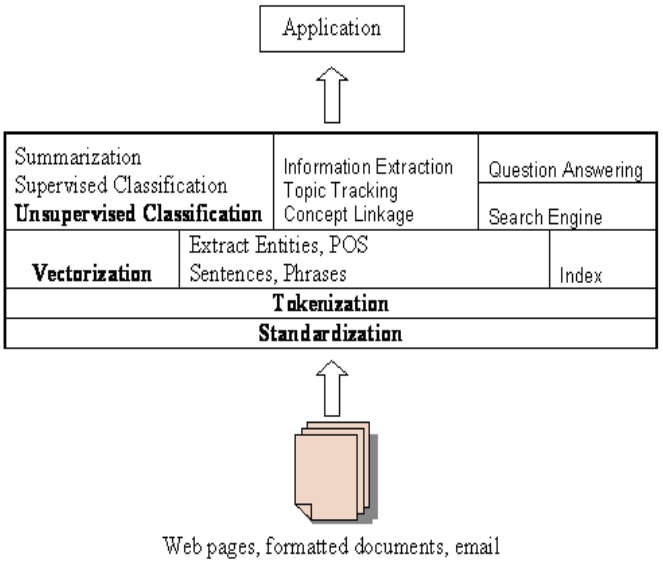

Figure 2.1: A layered model of Text Mining Systems

[17]. We will talk about document representation in Chapter 3 Section 3.1.

In the next layer, we show the techniques of Text Mining. Unsupervised classification or clustering use as a representation of the document a vector of tokens. It is easier to compare documents using such vectors. At its simplest, a document collection can be any grouping of text-based documents. Practically speaking, however, most text mining solutions are aimed at discovering patterns across very large document collections. The number of documents in such collections can range from the many thousands to the tens of millions.

Document collections can be either static, in which case the initial complement of documents remains unchanged, or dynamic, which is a term applied to document collections characterized by their inclusion of new or updated documents over time. Extremely large

document collections, as well as document collections with very high rates of document change, can pose performance optimization challenges for various components of a text mining system [10].

2.2 The Text Document

A text document can be very informally defined as a unit of discrete textual data within a collection that usually, but not necessarily, correlates with some real-world document such as a business report, legal memorandum, e-mail, research paper, manuscript, article, press release, or news story [10].

A document generally exists in any number or type of collections – from the very formally organized to the very ad hoc. A document can also be a member of different document collections, or different subsets of the same document collection, and can exist in these different collections at the same time. For example, a document related to Microsoft's antitrust litigation could exist in completely different document collections oriented toward current affairs, legal affairs, antitrust-related legal affairs, and software company news [10].

Despite the somewhat misleading label that it bears as unstructured data, a text document may be seen, from many perspectives, as a structured object. From a linguistic perspective, even a rather innocuous document demonstrates a rich amount of semantic and syntactical structure, although this structure is implicit and to some degree hidden in its textual content. In addition, typographical elements such as punctuation marks, capitalization, numerics, and special characters – particularly when coupled with layout artifacts such as white spacing, carriage returns, underlining, asterisks, tables, columns,

and so on – can often serve as a kind of "soft markup" language, providing clues to help identify important document subcomponents such as paragraphs, titles, publication dates, author names, table records, headers, and footnotes.

Problems about to high feature dimensionality (i.e., the size and scale of possible combinations of feature values for data) are typically of much greater magnitude in text mining systems than in classic data mining systems. Structured representations of natural language documents have much larger numbers of potentially representative features – and thus higher numbers of possible combinations of feature values – than one generally finds with records in relational or hierarchical databases.

The high dimensionality of potentially representative features in document collections is a driving factor in the development of text mining preprocessing operations aimed at creating more streamlined representational models. This high dimensionality also indirectly contributes to other conditions that separate text mining systems from data mining systems such as greater levels of pattern overabundance and more acute requirements for postquery refinement techniques [10].

Another characteristic of natural language documents is what might be described as feature sparsity. Only a small percentage of all possible features for a document collection as a whole appears in any single document, and thus when a document is represented as a binary vector of features, nearly all values of the vector are zero [10].

2.3 Document Features

Because text mining algorithms operate on the feature-based representations of documents and not the underlying documents themselves, there is often a trade-off between two important goals. The first goal is to achieve the correct calibration of the volume and semantic level of features to portray the meaning of a document accurately, which tends to incline text mining preprocessing operations toward selecting or extracting relatively more features to represent documents. The second goal is to identify features in a way that is most computationally efficient and practical for pattern discovery, which is a process that emphasizes the streamlining of representative feature sets; such streamlining is sometimes supported by the validation, normalization, or cross-referencing of features against controlled vocabularies or external knowledge sources such as dictionaries, thesauri, ontologies, or knowledge bases to assist in generating smaller representative sets of more semantically rich features [10].

Although many potential features can be employed to represent documents, the following four types are most commonly used [10]:

2.3.1 Characters

The individual component-level letters, numerals, special characters and spaces are the building blocks of higher-level semantic features such as words, terms, and concepts.

2.3.2 Words

A linguistic definition of a word is the smallest syntactic unit that cannot be broken into smaller segments. The construction of word is often a complex mix of rules. Words are used in their root forms (morphemes) or modified forms. Morphological rules govern how a root form can be modified with a prefix or suffix. Words in a sequence governed by the grammar of the language form sentences (Weiss, Indurkhya, Zhang and Damerau) [31].

Specific words selected directly from a "native" document are at what might be described as the basic level of semantic richness. This can lead to some word-level representations of document collections having tens or even hundreds of thousands of unique words in its feature space. However, most word-level document representations exhibit at least some minimal optimization and therefore consist of subsets of representative features filtered for items such as stop words, symbolic characters, and meaningless numerics [10].

2.3.3 Part of speech

Words can be classified into word classes or part of speech (POS). The eight standard parts of speech are adjectives, adverbs, conjunctions, determiners, nouns, prepositions, pronouns, and verbs. Two other word classes, interjections and punctuation marks, are sometimes included among the POS. We refer to four of the eight parts of speech: nouns, verbs, adjectives and adverbs as content words. The remaining four parts of speech: conjunctions, determiners, pronouns, and prepositions are called function words. Most of the words in the lexicon (dictionary) are content words [17].

2.3.4 Terms

Terms are single words and multiword phrases selected directly from the corpus of a native document by means of term-extraction methodologies. Term-level features, in the sense of this definition, can only be made up of specific words and expressions found within the native document for which they are meant to be generally representative. Hence, a term-based representation of a document is necessarily composed of a subset of the terms in that document.

Several of term-extraction methodologies can convert the raw text of a native document into a series of normalized terms – that is, sequences of one or more tokenized and lemmatized word forms associated with part-of-speech tags. Sometimes an external lexicon is also used to provide a controlled vocabulary for term normalization. Term-extraction methodologies employ various approaches for generating and filtering an abbreviated list of most meaningful candidate terms from among a set of normalized terms for the representation of a document. This culling process results in a smaller but relatively more semantically rich document representation than that found in word-level document representations [10].

2.3.5 Concepts

Concepts are features generated for a document by means of manual, statistical, rule-based, or hybrid categorization methodologies. Concept-level features can be manually generated for documents but are now more commonly extracted from documents using complex preprocessing routines that identify single words, multiword expressions, whole clauses, or even larger syntactical units that are then related to specific concept identifiers

[10].

2.4 Domains and Background Knowledge

In text mining systems, concepts belong not only to the descriptive attributes of a particular document but generally also to domains. With respect to text mining, a domain has come to be loosely defined as a specialized area of interest for which dedicated elements of information may be developed.

Domains can include very broad areas of subject matter (e.g., biology) or more narrowly defined specialisms (e.g., genomics or proteomics). Text mining systems with some element of domain-specificity in their orientation can leverage information from formal external knowledge sources for these domains to greatly enhance elements of their preprocessing, knowledge discovery, and presentation-layer operations [10].

Domain knowledge, more frequently referred to in the literature as background knowledge, can be used in text mining preprocessing operations to enhance concept extraction and validation activities. In preprocessing operations, background knowledge is an important adjunct to classification and concept-extraction methodologies. Background knowledge can also be leveraged to enhance core mining algorithms and browsing operations. In addition, domain-oriented information serves as one of the main bases for search refinement techniques.

In addition, background knowledge may be utilized by other components of a text mining system. For instance, background knowledge may be used to construct meaningful constraints in knowledge discovery operations. Likewise, background knowledge may also

be used to formulate constraints that allow users greater flexibility when browsing large result sets [10].

Chapter 3

Preprocessing

For data mining systems it is assumed that data have already been stored in a structured format, then much of its preprocessing focus falls on tasks such normalization, error detection and correction and dimension reduction. For text mining systems, preprocessing operations center on the identification and extraction of representative features for natural language documents. These preprocessing operations are responsible for transforming unstructured data stored in document collections into a more explicitly structured intermediate format, which is a concern that is not relevant for most data mining systems [10].

The preprocessing operations that support text mining attempt to leverage many different elements contained in a natural language document in order to transform it from an irregular and implicitly structured representation into an explicitly structured representation. However, given the potentially large number of words, phrases, sentences, typographical elements, and layout artifacts that even a short document may have, an essential

task for most text mining systems is the identification of a simplified subset of document features that can be used to represent a particular document as a whole [10].

3.1 Text Representation

In order to be clustered, the documents must be represented by a data structure, which is more appropriate for further processing than a plain text file. Even though, several methods exist that try to exploit the syntactic structure and semantics of text, most text mining approaches are based on the idea that a text document can be represented by a set of words, i.e. a text document is described based on the set of words contained in it (bag-of-words representation). However, in order to be able to define at least the importance of a word within a given document, usually a vector representation is used, where for each word a numerical "importance" value is stored. The currently predominant approaches based on this idea are the probabilistic model, the logical model and the vector space model (Hotho, Nürnberger and Paaß) [15].

3.1.1 The probabilistic model

The probabilistic model uses the probability theory for modeling documents, based on the "Probability Ranking Principle" (Robertson, 77) that says that optimum retrieval is achieved when documents are ranked according to decreasing values of the probability of relevance (with respect to certain query). Given a user query, there is an ideal set of documents which contains exactly the relevant documents and no other. The query process is considered as a process of specifying the properties of that ideal answer set. The initial

guess allows us to generate a preliminary probabilistic description of the ideal answer set which is used to retrieve a first set of documents. Interaction with the user is then initiated with the purpose of improving the probabilistic description of the ideal answer set. More details can be found in (Fuhr) [12].

3.1.2 The logical model

The basic point of a logical model is the assumption that documents can be represented effectively by logical formulas and the notion of logical consequence is used to decide relevance (Rijsbergen) [30]. For example, in the propositional model, a type of logical model, the indexing vocabulary is represented by a propositional alphabet (Losada) [20]. Each propositional letter represents an index term. Documents are represented by propositional formulas. For instance, given a propositional alphabet with the form $P = \{algebra, calculus, \ldots\}$, a document represented by the formula $algebra$ is a document dealing with algebra and a document represented by $\neg calculus$ is a document which does not deal with calculus. A query represented by $algebra \wedge \neg calculus$ is asking for documents dealing with algebra and not dealing with calculus. The implementation of logical models is usually complex.

3.1.3 The vector space model

Text documents can be conveniently represented in a high-dimensional vector space where terms are associated with vector components. Despite of its simple data structure without using any explicit semantic information, the vector space model enables very efficient analysis of huge document collections. It was originally introduced for indexing and

information retrieval but is now used in several text mining approaches as well as in most of the currently available document retrieval systems [10].

The most common way of doing this, the bag-of-words document representation, assumes that each word is a dimension in the feature space. Each vector representing a document in this space will have a component for each word. If a word is not present in the document, the word's component of the document vector will be zero. Otherwise, it will be some positive value, which may depend on the frequency of the word in the document and in the whole document collection. The details and the different possibilities of the bag-of-words document representation are discussed in Section 3.5 Vectorization.

For text classification the most widely used model is the vector space model and is the one used in this thesis.

3.2 Document Standardization

Documents exist in a variety of different formats, depending on how the documents were generated. For example, some documents may have been generated by a word processor with its own proprietary format; others may have been generated using a simple text editor and saved as ASCII text; and some may have been scanned and stored as images. Clearly, if we are to process all the documents, it's helpful to convert them to a standard format.

3.2.1 XML

The computer industry as a whole, including most of the text-processing community, has adopted XML (Extensible Markup Language) as its standard exchange format

[31], and this is the standard we adopt for our document collections as well. Briefly, XML is a standard way to insert tags onto a text to identify its parts. Tags can be nested within other tags to arbitrary depth. We assume that each document is marked off from the other documents in the corpus by having a distinguishing tag at the beginning, such as <DOC>. By XML convention, tags come in beginning and ending pairs. They are enclosed in angle brackets, and the ending tag has a back slash immediately following the opening angle bracket. Within a document, there can be many other tags to mark off sections of the document. Common sections are <DATE>, <SUBJECT>, <TOPIC>, and <TEXT>. The names are arbitrary. They could just as well be <HEADLINE> and <BODY>.

An example of an XML document is shown in Figure 3.1, where the document has a distinguishing tag of <DOC>.

Many currently available corpora are already in this format (e.g., the new corpus available from Reuters). The main reason for identifying the pieces of a document consistently is to allow selection of those parts that will be used to generate features. We will almost always want to use the part delimited as <TEXT> but may also want to include parts marked <SUBJECT>, <HEADLINE>, or the like. Additionally, for text classification or clustering, one wants to generate features from a TOPIC section if there is one.

Many word processors allow documents to be saved in XML format, and standalone filters can be obtained to convert existing documents without having to process each one manually [31].

The main advantage of standardizing the data is that the mining tools can be applied without having to consider the format of the document. For extract information

```
<DOC>
<TEXT>
<TITLE>
Parallel computation of kernel density estimates classifiers and their
ensembles
</TITLE>
<AUTHORS>
<AUTHOR>
Elio Lozano
</AUTHOR>
<AUTHOR>
Edgar Acuña
</AUTHOR>
</AUTHORS>
<ABSTRACT>
Nonparametric supervised classifiers are interesting because they do not require
distributional assumptions for the class conditional density,such as nomality
or equal covariance. However their use is not widespread because it takes a lot
of time to compute them due to the intensive use of the available data.
On the other hand bundling classifiers to produce a single one, known
as an ensemble, leads in general to an improvement in the accuracy of the classification.
Two popular methods for creating ensembles are Bagging introduced by Breiman, (1996)
and, AdaBoosting by Freund and Schapire (1996). In this paper we have used parallel
programming to compute the cross validated misclassification error for classifiers
based on kernel density estimation as well as their ensembles.
</ABSTRACT>
</TEXT>
</DOC>
```

Figure 3.1: A XML document.

from a document, it is irrelevant what editor was used to create it or what the original format was [31].

In this thesis, we assume that documents are already standardized and we will focus in the next steps.

3.3 Tokenization

The first step in handling text is to break the stream of characters into words or, more precisely, tokens [31]. A token is a more formal definition of a single unit of text. A single word may not be the smallest unit of text and a token may consist of one or more words [10].

A token is a word, number, punctuation mark, or any other sequence of characters that should be treated as a single unit [17].

The importance of tokenization is sometimes overlooked since it appears to be a simple task. But the accurate extraction of tokens is important for precise results in higher-level applications. Vector representations of documents used in clustering and text categorization are made up of a sequence of tokens and weights. Documents can be correctly categorized only when the vector representatives accurately the contents of documents [17].

To get the best possible features, one should always customize the tokenizer for the available text—otherwise extra work may be required after the tokens are obtained. Note that the tokenization process is language-dependent. In this thesis we focus on documents in English. For other languages, although the general principles will be the same, the details will differ [31].

There are many considerations that we have to take into account when developing a tokenizer. We considered the following:

1. We assume that the characters space, tab, and newline are always delimiters and are not counted as tokens. They are often collectively called white space.

2. The characters ( ) <> ¿‘ are always delimiters and may also be tokens.

3. Non-alphanumeric characters could be mixed with alphanumeric characters to form tokens (Yahoo!, AT&T)

4. A period is usually a sentence separator, but it can be found in abbreviations, the initials for a person, or in IP addresses (Mr. Smith, lb., or 136.145.155.1). For the purposes of tokenization, it is probably best to treat any ambiguous period as a word delimiter and also as a token [3].

5. Hyphens join words that usually belong to two different tokens to form a single token (small-scale, x-ray). A hyphen can also be found in a range of numbers (32-122).

6. Web URL and email addresses can have a number of embedded non-alphanumeric characters.

7. A number can be found in different forms: integer, real, with an exponent, or with a sign (-3.23E-03). A period, comma, or colon between numbers would not normally be considered a delimiter but rather part of the number.

8. The apostrophe has a number of uses. When preceded and followed by non-delimiters, it should be treated as part of the current token (e.g., isn’t or D’angelo). When fol-

lowed by an unambiguous terminator, it might be a closing internal quote or might indicate a possessive (e.g., Ross'). An apostrophe preceded by a terminator is unambiguously the beginning of an internal quote, so it is possible to distinguish the two cases by keeping track of opening and closing internal quotes.

9. A string of words such as "with respect to" or "kick the bucket" is interpreted as a single token, even tough, spaces are found within such tokens.

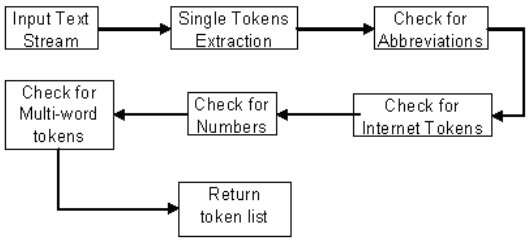

Figure 3.2: Steps in the tokenization process.

As shown in figure 3.2 the tokenization process consists in the the following: A text stream is received as input. First we extract the single tokens. Then, we assemble composite tokens using a set of rules and dictionary tables. Several passes are made over the list of tokens to find composite tokens such as abbreviations, numbers, internet tokens, and multi word tokens.

Single Tokens Extraction

The algorithm for the single token extraction as proposed in [17] is as follows:

Algorithm 1 *Single_ Tokens_ Extraction*

Input:

$S =$ *The text stream*

Output:

$T =$ *list of tokens*

1. *Define the set of legal token characters (alphanumeric characters and optional characters) and initialize a token list.*

2. *Scan the text stream one character at a time; if the current character is not in the ASCII range of 32 to 122, assign a space to the character.*

   (a) *If the current character is a token character:*

      i. *If the previous character was not a token character, add the previous token to the list and create a new token.*

      ii. *Concatenate the current character to the current token. Continue at step 2.*

   (b) *Else, (if the current character is a space character):*

      i. *If the previous character was not a space, add the previous token to the list.*

      ii. *Create a new token with a space (consecutive space characters form one token). Continue at step 2.*

   (c) *Default: All others characters form individual tokens.*

3. *Handle the last token.*

Composite Tokens Extraction

The abbreviations are stored in a database table as proposed in [17]. This table contains a list of common abbreviations for currencies, dimensions, time, places, and some organizations. This list can be extended to include custom abbreviations. When a period is encountered in the token stream, it may represent the last character of a potential abbreviation. The token immediately preceding the period is checked against the list of abbreviations. When the abbreviation consist of two words the combination of the two tokens preceding the period is checked against the list of dictionary abbreviations [17].

Internet tokens come in many forms, including email addresses, URLs, hostnames, and IP addresses. If we find four numbers in the range 0 to 255 that are joined together with periods, the token is considered an IP address. Hostnames of the form www.cnn.com and www.whitehouse.gov are scanned from right to left. If the rightmost token is com, edu, gov, net, or any other legitimate code (including all two character country codes) and the preceding token is a period, we have a potential hostname. Anything preceding the last two tokens that is not a space is part of the internet token. This takes care of email addresses as well.Internet URLs have a well-defined format. The first token is a protocol such as http, ftp, or telnet, followed by the string :// and the rest of the URL. Anything that is not a space character to the right of the string :// is considered part of the URL [17].

Like internet tokens, number tokens also come in many forms. The basic integer is just a string of digits. Optionally, commas can be embedded in integers to separate digits. A real number is two strings of digits joined by a period. An optional exponent is sometimes included. The number and exponent can both be preceded by an optional sign (+ or -).

Errors in detecting numbers are usually caused by a space inserted between the sign and the number or between the number and the exponent.

In the final pass, we identify common collocations such as "with respect to", "as well as", "second hand", and "drop in". The lead tokens for all collocations and the collocations themselves could be stored in a table. Any token from the text stream that is not found in the set of lead tokens is skipped [17].

3.4 Lemmatization or Stemming

Stemming consists of converting each word to its stem, i.e. a neutral form with respect to tag-of-speech and verbal/plural inflections. In essence, to get the stem of a word it is necessary to eliminate its suffixes representing tag-of-speech and/or verbal/plural inflections. For instance, the words “compression” and “compressed” would both be converted to their stem “compress”.

Whether or not this step is necessary is application-dependent. Notice that one effect of stemming is to reduce the number of distinct types in a text corpus and to increase the frequency of occurrence of some individual types. Stemming algorithms usually incorporate a great deal of linguistic knowledge, so that they are language-dependent. The most popular algorithm used for stemming is the Porter’s algorithm, originally developed for the English language (Larocca, Alexandre, Kaestner and Freitas) [18].

3.4.1 Inflectional Stemming

In many languages, words occur in text in more than one form. For example singular and plural. Often, but not always, it is advantageous to eliminate this kind of variation before further processing. When the normalization is confined to regularizing grammatical variants such as singular/plural and present/past, the process is called "inflectional stemming." In linguistic terminology, this is called "morphological analysis". In English, with many irregular word forms and nonintuitive spelling, the process is not easy. There is no simple rule, for example, to bring together "seek" and "sought." Similarly, the stem for "rebelled" is "rebel," but the stem for "belled" is "bell" [31].

An algorithm for inflectional stemming must be part rule-based and part dictionary-based. Any stemming algorithm for English that operates only on tokens, without more grammatical information such as part-of-speech, will make some mistakes because of ambiguity. For example, is "bored" the adjective as in "he is bored" or is it the past tense of the verb "bore"? Furthermore, is the verb "bore" an instance of the verb "bore a hole," or is it the past tense of the verb "bear"? In the absence of some often complicated disambiguation process, a stemming algorithm should probably pick the most frequent choice.Although the inflectional stemmer is not expected to be perfect, it will correctly identify quite a siguificant number of stems [31].

3.4.2 Stemming to a Root

Stemming to a root refers to the process of reach a root form with no inflectional or derivational prefixes and suffixes. For example, "denormalization" is reduced to the stem

"norm." The end result of such aggressive stemming is to reduce the number of tokens in a text collection very drastically, thereby making distributional statistics more reliable. Additionally, words with the same core meaning are coalesced, so that a concept such as "apply" has only one stem, although the text may have "reapplied", "applications", etc [31].

Extending the concept, we can also map synonyms to the same token. This adds a layer of complexity to the processing of text. Stemming can occasionally be harmful for some words. If we apply a universal procedure that effectively trims words to their root form, we will encounter occasions where a subtle difference in meaning is missed. The words "exit" and "exiting" may appear to have identical roots, but in the context of programming and error messages, they may have different meanings. Overall, stemming will achieve a large reduction in dictionary size and is modestly beneficial for predictive performance when using a smaller dictionary.

3.5 Vectorization

The collective set of features is typically called a dictionary or vocabulary (V). The tokens or words in the dictionary form the basis for creating the numeric vectors corresponding to the document collection.

More precisely, a text document d can be represented as a sequence of terms, $d = (w_1, w_2, ..., w_{|d|})$, where $|d|$ is the length of the document and $w_t \in V$. A vector-space representation of d is then defined as a real vector $x \in R^{|V|}$, where each component x_j is a statistic related to the occurrence of the j^{th} vocabulary entry in the document.

Vector-based representations are sometimes referred to as a 'bag of words', emphasizing that document vectors are invariant with respect to term permutations, since the original word order $w_1, w_2, ..., w_{|V|}$ is clearly lost. Representations of this kind are appealing for their simplicity. Moreover, although they are necessarily lossy from an information theoretic point of view, many text retrieval and categorization tasks can be performed quite well in practice using the vector-space model. Note that typically the total number of terms in a set of documents is much larger than the number of distinct terms in any single document, $|V| >> |d|$, so that vector-space representations tend to be very sparse. This property can be advantageously exploited for both memory storage and algorithm design.

The simplest vector-based representation is Boolean, i.e. $x_j \in \{0, 1\}$ indicates the presence or the absence of term w_j in the document being represented. Several refinements can be obtained by extending the boolean vector model and introducing real-valued weights associated with terms in a document. Other vectorization mechanisms are described in the next subsections.

3.5.1 Term Frequency

A more informative weighting scheme consists of counting the actual number of occurrences of each term in the document. This value may be multiplied by the constant $1/|d|$ to obtain a vector of term frequencies (TF) within the document.

Let $D = \{d_1, ..., d_n\}$ be a collection of documents. For each term $w_j \in V$, let n_{ij} denote the number of occurrences of w_j in d_i.Then we define:

$$TF_{ij} = \frac{n_{ij}}{|d_i|}$$

3.5.2 Inverse document frequency

A difference of Term frequencies that are relative to each document, inverse document frequency (IDF) is an 'absolute' measure of term importance. IDF decreases as the number of documents in which the term occurs increases in a given collection. So terms that are globally rare receive a higher weight.

Let $D = \{d_1, ..., d_n\}$ be a collection of documents. Let $w_j \in V$. Let n_j be the number of documents that contain w_j at least once. Then we define:

$$IDF_j = log\frac{n}{n_j}$$

Here the logarithmic function is employed as a damping factor.

3.5.3 The TF-IDF weight

An important family of weighting schemes combines term frequencies with inverse document frequency. Let $x = (x_{ij})$ be the vector representation of The TF–IDF weight of w_j in di can be computed as:

$$x_{ij} = TF_{ij}.IDF_j$$

Or alternatively as:

$$x_{ij} = \frac{TF_{ij}}{\max_{w_k \in d_i} TF_{ik}} \frac{IDF_j}{\max_{w_k \in d_i} IDF_k}$$

The IDF weighing is commonly used as an effective heuristic. A theoretical justification has been proposed by Papineni (2001) [23], who proved that IDF is the optimal weight of a term with respect to the minimization of a distance function that generalizes Kullback–Leibler divergence or relative entropy.

3.5.4 Stop Words Removal

An obvious reduction in dictionary size is to compile a list of stopwords and remove them from the dictionary. These are words that almost never have any predictive capability, such as articles a and the and pronouns such as it and they. These common words can be discarded before the feature generation process, but it's more effective to generate the features first, apply all the other transformations, and at the very last stage reject the ones that correspond to stopwords [31].

Chapter 4

Unsupervised Classification

Unsupervised classification, also known as clustering, is a process through which objects are classified into meaningful groups called clusters, without any prior information. Any labels associated with objects are obtained solely from the data.

Clustering is useful in a wide range of data analysis fields, including data mining, document retrieval, image segmentation, and pattern classification. In many such problems, little prior information is available about the data, and the decision-maker must make as few assumptions about the data as possible. It is for those cases the clustering methodology is especially appropriate.

4.1 Clustering Tasks in Text Analysis

One application of clustering is the analysis and navigation of big text collections such as Web pages. The basic assumption, called the cluster hypothesis, states that relevant documents tend to be more similar to each other than to non-relevant ones. If this

assumption holds for a particular document collection, the clustering of documents based on the similarity of their content may help to improve the search effectiveness [10].

4.1.1 Improving Search Recall

Standard search engines and IR systems return lists of documents that match a user query. It is often the case that the same concepts are expressed by different terms in different texts. For instance, a "car" may be called "automobile," and a query for "car" would miss the documents containing the synonym. However, the overall word contents of related texts would still be similar despite the existence of many synonyms. Clustering, which is based on this overall similarity, may help improve the recall of a query-based search in such a way that when a query matches a document its whole cluster can be returned.

This method alone, however, might significantly degrade precision because often there are many ways in which documents are similar, and the particular way to cluster them should depend on the particular query [10].

4.1.2 Improving Search Precision

As the number of documents in a collection grows, it becomes a difficult task to browse through the lists of matched documents given the size of the lists. Because the lists are unstructured, except for a rather weak relevance ordering, he or she must know the exact search terms in order to find a document of interest. Otherwise, the he or she may be left with tens of thousands of matched documents to scan.

Clustering may help with this by grouping the documents into a much smaller number of groups of related documents, ordering them by relevance, and returning only the

documents from the most relevant group or several most relevant groups.

Experience, however, has shown that the user needs to guide the clustering process so that the clustering will be more relevant to the user's specific interest. An interactive browsing strategy called scatter/gather is the development of this idea [10].

4.1.3 Scatter/Gather

The scatter/gather browsing method (Cutting et al. [5]; Hearst and Pedersen [14]) uses clustering as a basic organizing operation. The purpose of the method is to enhance the efficiency of human browsing of a document collection when a specific search query cannot be formulated. The method is similar to the techniques used for browsing a printed book. An index, which is similar to a very specific query, is used for locating specific information. However, when a general overview in needed or a general question is posed, a table of contents, which presents the logical structure of the text, is consulted. It gives a sense of what sorts of questions may be answered by more intensive exploration of the text, and it may lead to the particular sections of interest [10].

During each iteration of a scatter/gather browsing session, a document collection is scattered into a set of clusters, and the short descriptions of the clusters are presented to the user. Based on the descriptions, the user selects one or more of the clusters that appear relevant. The selected clusters are then gathered into a new subcollection with which the process may be repeated. In a sense, the method dynamically generates a table of contents for the collection and adapts and modifies it in response to the user's selection [10].

4.1.4 Query-Specific Clustering

Direct approaches to making the clustering query-specific are also possible. The hierarchical clustering is especially appealing because it appears to capture the essence of the cluster hypothesis best. The most related documents will appear in the small tight clusters, which will be nested inside bigger clusters containing less similar documents. The cluster hypothesis was tested in (Tombros, Villa and Rijsbergen) [27] on several document collections and showed that it holds for query-specific clustering.

Experiments with cluster-based retrieval using language models show that this method can perform consistently over document collections of realistic size, and a significant improvement in document retrieval can be obtained using clustering without the need for relevance information from by the user [10].

4.2 The General clustering problem

A clustering task may include the following components [10], :

- Problem representation.
- Definition of proximity measure suitable to the domain.
- Actual clustering of objects.
- Data abstraction, and
- Evaluation.

4.2.1 Problem Representation

All clustering problems are, in essence, optimization problems. The goal is to select the best among all possible groupings of objects according to the given clustering quality function. The quality function maps a set of possible groupings of objects into the set of real numbers in such a way that a better clustering would be given a higher value.

A good clustering should group together similar objects and separate dissimilar ones. Therefore, the clustering quality function is usually specified in terms of a similarity function between objects. In fact, the exact definition of a clustering quality function is rarely needed for clustering algorithms because the computational hardness of the task makes it infeasible to attempt to solve it exactly. Therefore, it is sufficient for the algorithms to know the similarity function and the basic requirement — that similar objects belong to the same clusters and dissimilar to separate ones.

A similarity function takes a pair of objects and produces a real value that is a measure of the objects' proximity. To do so, the function must be able to compare the internal structure of the objects.

The most common vector space model assumes that the objects are vectors in the high-dimensional feature space. A common example is the bag-of-words model of text documents. In a vector space model, the similarity function is usually based on the distance between the vectors in some metric.

4.2.2 Similarity Measures

The most popular metric is the usual Euclidean distance:

$D(x_i, x_j) = \sqrt{\sum_k (x_{ik} - x_{jk})^2}$

which is a particular case with $p = 2$ of Minkowski metric:

$D_p(x_i, x_j) = (\sum_k (x_{ik} - x_{jk})^p)^{1/p}$

For the text clustering, however, the cosine similarity measure is the most common:

$Sim(x_i, x_j) =< \hat{x}_i, \hat{x}_j >= \sum_k \hat{x}_{ik}.\hat{x}_{jk}$

where $\hat{x}$ is the normalized vector $\hat{x} = x/\left\|x\right\|$.

There are many other possible similarity measures.

4.3 Clustering algorithms

Traditionally clustering techniques are broadly divided in hierarchical and partitioning (Berkhin) [1]. Each of these can either be a hard clustering or a soft one. In a hard clustering, every object may belong to exactly one cluster. In soft clustering, the membership is fuzzy, objects may belong to several clusters with a fractional degree of membership in each.

Hierarchical algorithms build clusters gradually. The basics of hierarchical clustering include the idea of conceptual clustering. Classic algorithms SLINK (Single LINKage), COBWEB, as well as newer algorithms CURE (Clustering Using Representatives) and CHAMELEON.

Partitioning algorithms learn clusters directly. In doing so, they either try to discover clusters by iteratively relocating points between subsets (Partitioning Relocation Methods), or try to identify clusters as areas highly populated (Density-Based Partitioning). Partitioning Relocation Methods are further categorized into probabilistic clustering (EM

(Expectation Maximization) framework, algorithms SNOB, AUTOCLASS, MCLUST), k-medoids methods like algorithms (Kaufman and Rousseeuw) PAM (Partitioning Around Medoids), CLARA (Clustering LARge Applications), CLARANS (clustering Large Application based upon Randomized Search) and its extensions [16], and k-means methods (different schemes initialization, optimization, harmonic means, extensions). Such methods concentrate on how well points fit into their clusters and tend to build clusters of proper convex shapes.

Many other clustering techniques are developed, that work well in particular scenarios. The most commonly used algorithms are the K-means, the EM-based mixture resolving (soft, flat, probabilistic), and the HAC (hierarchical, agglomerative).

The clustering of textual data has several unique features that distinguish it other clustering problems. The most prominent feature of text documents as objects to be clustered is their very complex and rich internal structure. With big document collections, the dimension of the feature space may easily range into the tens and hundreds of thousands.

In this thesis we have implemented and compared two diferent clustering algorithms:

1. A variant of spherical k-means using the k-medoids approach: the PAM (Partitioning Around Medoids) algorithm [16], using the cosine similarity as measure of similarity.

2. EM using Multinomial Naive Bayes mixture model (Nigam, McCallum and Mitchell) [21].

In the following we will explain in detail the two algorithms.

4.4 PAM (Partitioning Around Medoids)

The PAM algorithm first computes k representative objects, called medoids. A medoid is an actual document that can be defined as the object of a cluster, whose average dissimilarity to all the objects in the cluster is minimal. In the classification literature, such representative objects are called centrotypes. After finding the set of medoids, each object of the data set is assigned to the nearest medoid. The best partition will be the one minimizing the average dissimilarity of objects to their closest representative object [16].

The algorithm is divided into two phases. In the first phase, called BUILT, the k representative objects are chosen. The second phase, called SWAP, is attempted to improve the set of representative objects that was chosen in the first phase. The algorithm works as follows:

BUILT PHASE

In this phase the first object chosen is the one for which the sum of the dissimilarities to the other objects is the smallest. This object is the most centrally located in the set of objects. At each step the object that decreases the objective function is selected.

1. Consider an object i which has not yet been selected.

2. Consider a non selected object j and calculate its dissimilarity D_j with the previous objects chosen and calculate its dissimilarity $d(j,i)$ with object i. Calculate the difference between D_j and $d(j,i)$. If the difference is positive, then object j will contribute in the selection of object i. Then calculate,

$$C_{ji} = max(D_j - d(j,i), 0).$$

3. Calculate the total gain obtained if object i is selected,

$$\sum_j C_{ji}$$

4. Choose the object i such that:

$$\arg\max_i \sum_j C_{ji}$$

5. The process ends when the k representative objects have been found.

Now, consider all the pair on objects (i, h) for which object i has been selected, but object h has not. The main objective is to determine if there is a positive effect when a swap is carried out, that is, when object i is no longer selected as a representative object, but object h is.

SWAP PHASE

1. To calculate the effect of a swap between objects i and h the following calculations need to be completed.

2. First, consider an object j that has not been selected. Then calculate its contribution C_{jih} to the swap:

   (a) If j is near from one of the other representative objects than from both i and h then the contribution of object j to the swap is $C_{jih} = 0$.

   (b) If j is not further from i than from any other selected representative object ($d(j, i) = D_j$), then consider this two situations:

i. If j is closer to h that from any other representative object, that is, $d(j,h) < E_j$ where Ej is the dissimilarity between j and the second most similar representative object, then the contribution of object j to the swap is

$$C_{jih} = d(j,h) - d(j,i)$$

ii. If j is at least as distant from h than from the second closest representative object, that is, $d(j,h) \geq Ej$, then the contribution of object j to the swap is

A. $C_{jih} = E_j - d(j,i)$

(c) If j is more distant from object i that from at least one of the other representative objects but closer to h than to any representative object the contribution to the swap is:

$$C_{jih} = d(j,h) - d(j,i)$$

3. Calculate de total result of a swap by adding the contributions C_{jih}:

$$T_{ih} = \sum_j C_{jih}$$

4. Select the pair (i,h) which:

$$\arg\min_{i,h} T_{ih}$$

5. The swap is carried out if minimum T_{ih} is negative and the algorithm return to step 1. If minimum T_{ih} is positive or zero, then the swap is not carry out and the algorithm stops.

4.5 EM Clustering Algorithm using Multinomial Naive Bayes Mixture Models

4.5.1 Multinomial Naive Bayes Mixture Model for Text

The underlying assumption of mixture-resolving algorithms is that the objects to be clustered are drawn from k distributions, and the goal is to identify the parameters of each that would allow the calculation of the probability $P(C_i|d)$ of the given object's belonging to the cluster C_i.

In this case, we assume that documents are generated by a mixture of multinomials model, where each mixture component corresponds to a class. Let there be K clusters and a vocabulary of size $|V|$; each document d_i has $|d_i|$ words in it.

Every document is generated according to a probability distribution defined by the parameters for the mixture model, denoted θ.

The probability distribution consists of a mixture of components $c_j \in \{1, ..., K\}$. A document, d_i, is created by first selecting a mixture component according to the mixture weights (or class probabilities), $P(c_j|\theta)$, then using this selected mixture component to generate a document according to its own parameters, with distribution $P(d_i|c_j;\theta)$.

Thus, the likelihood of seeing document d_i is a sum of total probability over all mixture components:

$$P(d_i|\theta) = \sum_{j=1}^{K} P(c_j|\theta)P(d_i|c_j;\theta) \tag{4.1}$$

The class label for a particular document d_i is written y_i. If document d_i was

generated by mixture component c_j we say $y_i = c_j$.

We consider the document, d_i, as a vector of word counts. We write N_{it} to be the number of times word w_t occurs in document d_i.

Another assumption we make is that the document lenght is independent of class. When a document is to be generated by a particular mixture component a document length, $|d_i| = \sum_{t=1}^{|V|} N_{it}$, is first chosen independently of the component. Then, the selected mixture component is used to generate a document of the specified length, by drawing from its multinomial distribution.

Using the above along with standard Naive Bayes assumption: that the words of a document are conditionally independent of the other words in the same document, given the class label, we can expand the second term from Equation 4.1, and express the probability of a document given a mixture component in terms of its constituent features: the document length and the words in the document.

$$P(d_i|cj;\theta) \propto P(|d_i|) \prod_{w_t \in V} P(w_t|c_j;\theta)^{N_{it}} \tag{4.2}$$

Thus the parameters of an individual mixture component define a multinomial distribution over words, i.e. the collection of word probabilities, each written $\theta_{wt|cj}$, such that $\theta_{wt|cj} \equiv P(w_t|c_j;\theta)$,where $t \in \{1, ..., |V|\}$ and $\sum_t P(w_t|c_j;\theta) = 1$.

Since we assume that for all classes, document length is identically distributed, it does not need to be parameterized for classification.

Then, the other parameter of the model is the mixture weights (class probabilities), $\theta_{c_j} \equiv P(c_j|\theta)$, which indicate the probabilities of selecting the different mixture components. Thus the complete collection of model parameters, θ, defines a set of multinomials and class

probabilities:

$$\theta = \left\{\theta_{wt|cj} : w_t \in V, c_j \in \{1, ..., K\}; \theta_{c_j} : c_j \in \{1, ..., K\}\right\}$$

To summarize, the full generative model, given by combining equations 4.1 and 4.2, assigns probability $P(D_i|\theta)$ to generating document d_i as follows:

$$P(d_i|\theta) \propto P(|d_i|) \sum_{j=1}^{K} P(c_j|\theta) \prod_{w_t \in V} P(w_t|c_j;\theta)^{N_{it}} \tag{4.3}$$

Maximum a posteriori (MAP) estimation of multinomial parameters

Let X be a random discrete variable assuming values on $1, 2, \ldots, K$, such that $p(x = k) = p_k$. This can be written as:

$$P(x|p) = \prod_{k=1}^{K} p_k^{\delta(x=k)}$$

where $\delta(x = k) = 1$ if $x = k$, and 0 otherwise. The joint distribution of a random sample of X of size n will be:

$$P(X|p) = \prod_{i=1}^{n} \prod_{k=1}^{K} p_k^{\delta(x=k)} = \prod_{k=1}^{K} p_k^{N_k}$$

where, $N_k = \sum_i \delta(x_i = k)$.This represents the number of times that the value k appears in the sample. The joint probability of $N_1, \ldots, N_k$ is given by

$$P(N_1, \ldots, N_k|p) = N! \prod_{k=1}^{K} \frac{p_k^{N_k}}{N_k!}$$

where $N = \sum_k N_k$.This a multinomial distribution with parameter p.

A conjugate prior for p is the Dirichlet distribution, given by:

$$P(p|\alpha_1, ..., \alpha_k) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_{k=1}^{K} p_k^{\alpha_k - 1}$$

Large α correspond to strong prior knowledge about the distribution and small α correspond to ignorance. The maximum of the density is obtained at

$$p_k = \frac{\alpha_k - 1}{\sum_{k=1}^{K} \alpha_k - K}$$

The joint distribution of X and p is given by

$$P(X, p|\alpha_1, ..., \alpha_k) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_{k=1}^{K} p_k^{N_k + \alpha_k - 1}$$

The posterior $P(p|X; \alpha)$ is a Dirichlet with parameter $N_k + \alpha_k$.

Hence the MAP estimator of p_k will be:

$$\hat{p}_k = \frac{N_k + \alpha_k - 1}{\sum_{k=1}^{K} N_k + \sum_{k=1}^{K} \alpha_k - K} \tag{4.4}$$

Maximum a posteriori (MAP) estimation of the multinomial naive bayes parameters

Learning a naive Bayes text classifier from a set of labeled documents consists of estimating the parameters of the generative model. The estimate of the parameters is writen $\hat{\theta}$. We use the maximum a posteriori (MAP) estimate of the parameters, that is finding $\arg\max_\theta P(\theta|D, Y)$. This is the value of θ that is most probable given the evidence of the training data and a prior.By Bayes rule:

$$\arg\max_\theta P(\theta|D) \propto \arg\max_\theta P(\theta) P(D|\theta)$$

$$P(\theta)P(D|\theta) = P(\theta) \prod_{i=1}^{|D|} P(d_i|\theta)$$

Assuming a conjugate prior with a Dirichlet distribution with $\alpha_k = 2$, for the

vector of parameters $\theta = (\theta_{c_1}, ..., \theta_{c_K}, \theta_{w_1|c_1|}, ..., \theta_{w_{|V|}|c_{|K|}})$

where $\theta_{c_j} \equiv P(c_j|\theta)$ and $\theta_{wt|cj} \equiv P(w_t|c_j;\theta)$, the following is obtained:

$$P(\theta|D) \propto \prod_{i=1}^{|D|} P(|d_i|) \sum_{j=1}^{K} \theta_{c_j} \prod_{w_t \in V} \theta_{wt|cj}^{N_{it}} \prod_{j=1}^{K} \theta_{c_j} \prod_{w_t \in V} \theta_{wt|cj} \tag{4.5}$$

Therefore:

$$P(\theta|D) \propto \prod_{i=1}^{|D|} P(|d_i|) \sum_{j=1}^{K} \theta_{c_j} \prod_{j=1}^{K} \theta_{c_j} \prod_{w_t \in V} \theta_{wt|cj}^{N_{it}+1} \tag{4.6}$$

Equation 4.6 can be written as:

$P(\theta|D) \propto \sum_{j=1}^{K} \prod_{l=1}^{K} \theta_{c_j}^{M^*+1} \prod_{t=1}^{|V|} \theta_{wt|cj}^{N_t^*+1}$

where $M_l^* = \sum_{i=1}^{|D|} \delta_{ij}$ and $N_t^* = \sum_{i=1}^{|D|} \delta_{ij} N_{it}$ with $\delta_{ij} = 1$ if $y_i = c_i$ and 0 otherwise.

Using maximum a posteriori (MAP) to estimate the parameters $\theta_{wt|cj}$ and θ_{c_j} of a multinomial distribution with Dirichlet prior, as shown in equation 4.5, yields a familiar soothed ratios of empirical counts:

$$\hat{\theta}_{wt|cj} \equiv P(w_t|c_j;\hat{\theta}) = \frac{1 + \sum_{i=1}^{|D|} \delta_{ij} N_{it}}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} \delta_{ij} N_{is}} \tag{4.7}$$

where δ_{ij} is given by the class label: 1 when $y_i = c_j$ and 0 otherwise.

The class probabilities, $\hat{\theta}_{cj}$, are estimated in the same manner, and also involve a ratio of counts with smoothing:

$$\hat{\theta}_{c_j} \equiv P(c_j|\hat{\theta}) = \frac{1 + \sum_{i=1}^{|D|} \delta_{ij}}{K + |D|} \tag{4.8}$$

Given estimates of these parameters, it is possible to turn the generative model backwards and calculate the probability that a particular mixture component generated a

given document to perform classification. This follows from an application of Bayes' rule:

$$\begin{aligned} P(y_i &= c_j|d_i;\hat{\theta}) = \frac{P(c_j|\hat{\theta})P(d_i|c_j;\hat{\theta})}{P(d_i|\hat{\theta})} \\ P(y_i &= c_j|d_i;\hat{\theta}) = \frac{P(c_j|\hat{\theta}) \prod_{w_t \in V} P(w_t|c_j;\hat{\theta})^{N_{it}}}{\sum_{k=1}^{K} P(c_k|\hat{\theta}) \prod_{w_t \in V} P(w_t|c_k;\hat{\theta})^{N_{it}}} \end{aligned} \tag{4.9}$$

Then, to classify a test document into a single class, the class with the highest posterior probability, $\arg\max_j P(y_i = c_j|d_i;\hat{\theta})$ is selected.

4.5.2 EM clustering algorithm with Multinomial Naive Bayes

For unsupervised classification, we would still like to find MAP parameter estimates, as in the supervised setting above. Because there are no labels for the data, the closed-form equations from the previous section are not applicable. However, using the Expectation-Maximization (EM) technique, we can find locally MAP parameter estimates for the generative model.

More formally, learning a classifier is approached as calculating a maximum a posteriori estimate of θ, i.e. $\arg\max_\theta P(\theta)P(D,Y|\theta)$, which is equivalent to maximizing the log of the same. Consider the second term of the maximization, the probability of all the observable data. The probability of an individual unlabeled document is a sum of total probability over all the classes, as in Equation 4.1. The expected log probability of the data is:

$$l(\theta|D,Y) = \log(P(\theta)) + \sum_{i=1}^{|D|} \log \sum_{j=1}^{K} P(c_j|\theta)P(d_i|c_j;\theta) \tag{4.10}$$

The formalism of Expectation-Maximization (EM) (Dempster et al. [1977]) provides an iterative hill-climbing approach to finding a local maxima of model probability in parameter space. The E-step of the algorithm estimates the expectations of the missing values (i.e. class information) given the latest iteration of the model parameters. The M-step maximizes the likelihood of the model parameters using the previously-computed expectations of the missing values as if they were the true ones.

In practice, the E-step corresponds to performing classification of each unlabeled document using Equation 4.9. The M-step corresponds to calculating a new maximum a posteriori estimate for the parameters, $\hat{\theta}$, using Equations 4.7 and 4.8 with the current estimates for $P(c_j|D_i;\hat{\theta})$.

Essentially all initializations of the parameters lead to some local maxima with EM. Many instantiations of EM begin by choosing a starting model parameterization randomly. The algorithm iterates until it converges to a point where $\hat{\theta}$ does not change from one iteration to the next. Algorithmically, we determine that convergence has occurred by observing a below-threshold change in the log-probability of the parameters (Equation 4.10), which is the height of the surface on which EM is hillclimbing.

Algorithm 2 *EM_Clustering_NaiveBayes*

Inputs:

$D =$ *Collection of unlabeled documents*

$K =$ *Number of cluster (components of the mixture model)*

1. *Select eather randomly or by any other method a initial estimate of the parameters* $\hat{\theta}$.

   (Or initial labels for the documents and use maximum a posteriori parameter estima-

tion to find $\hat{\theta}$)

2. *Loop while classifier parameters improve, as measured by the change in $l(\theta|D,Y)$ (the log probability of the unlabeled data, and the prior) (see Equation 4.10):*

   (E-step) *Use the current classifier, $\hat{\theta}$, to estimate component membership of each document, i.e., the probability that each mixture component generated each document, $P(c_j|D_i;\hat{\theta})$ (see Equation 4.9).*

   (M-step) *Re-estimate the classifier, $\hat{\theta}$, given the estimated component membership of each document. Use maximum a posteriori parameter estimation to find $\hat{\theta} = \arg\max_\theta P(D,Y|\theta)P(\theta)$ (see Equations 4.7 and 4.8).*

4.6 Evaluation of text clustering

Measuring the quality of a clustering algorithm is a common problem in text as well as data mining. In unsupervised classification algorithms, the real partitions are not known, you will never know what the absolute true solution is, but nevertheless some validation methods could help to understand how well the algorithm worked with the data.

In order to investigate cluster validity, there are three approaches: external criteria, internal criteria, and relative criteria (Gonzales) [13].

• External validation measures. This approach is based on a previous knowledge of classes of the data set.

• Internal validation measures. This approach is based on the information intrinsic to the data set alone.

• Relative Criteria. Given a set of parameters associated with a specific clustering algorithm, among the clusters obtained by an algorithm using different values of the parameter, choose the one that best fits the data set.

In this thesis, we have implemented and used the following validity measures:

4.6.1 Purity

Given a set of categorized (manually classified) documents, it is possible to use this benchmark labeling for evaluation of clustering, this is known as external criteria measure. The most common measure is purity [10]. Assume $\{C_1, C_2, \ldots, C_n\}$ are the manually labeled classes of documents, and $\{K_1, K_2, \ldots, K_m\}$are the clusters returned by the clustering process [10]. Then cluster *purity* of cluster K_i is defined as:

$$purity(K_i) = \frac{1}{|K_i|} \max_j |C_j \cap K_i|$$

Note that each cluster may contain samples from different classes. Purity gives the ratio of the dominant class size in the cluster to the cluster size itself. A high purity value implies that the cluster is a "pure" subset of the dominant class [7].

4.6.2 Entropy

Let $\{C_1, C_2, \ldots, C_c\}$ be the manually labeled classes of documents, and $\{K_1, K_2, \ldots, K_k\}$ the clusters returned by the clustering process. *Entropy* (H) is defined as follows [7]:

$$H(K_i) = \frac{1}{\log c} \sum_{j=1}^{c} \frac{|C_j \cap K_i|}{|K_i|} \log \frac{|C_j \cap K_i|}{|K_i|}$$

Entropy is a more comprehensive measure than purity. It considers the distribution of classes in a cluster. Note that, in this case, entropy is normalized to take values between 0 and 1. An entropy value of 0 means the cluster is comprised entirely of one class, while an entropy value near 1 is bad since it implies that the cluster contains a uniform mixture of classes.

4.6.3 Silhouette Index

In order to construct silhouettes we need a partition obtained by the application of some clustering algorithm, and a dissimilarity matrix containing all the dissimilarities between objects.

This method assign to each object a quantitative measure $s(i)$, known as the silhouette width. The silhouette width indicates the membership of object i in the cluster it has been assigned [13].

Let i be any object in the data set and denote by C_j the cluster to which object i has been assigned. Let $a(i)$ the average dissimilarity between i and all the other object in cluster C_j. Consider any cluster C_k different to cluster C_j , and define $d(i, C_k)$ as the average dissimilarity of i to all objects of C_k and compute:

$$b(i) = \min_{c_k \neq C_j} d(i, C_k) \qquad k = 1, 2, ..., n; k \neq j$$

The cluster C_b for which this minimum is attained is called the neighbor of object i. This is like the second best choice for object i.

Then, the silhouette width is defined as:

$$
\begin{aligned}
s(i) &= 1 - \frac{a(i)}{b(i)} \quad if\ a(i) < b(i) \\
s(i) &= 1 - \frac{b(i)}{b(i)} \quad if\ a(i) > b(i) \\
s(i) &= 0 \qquad\qquad\quad if\ a(i) = b(i)
\end{aligned}
$$

Or equivalently

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

From the definition we can see that $-1 \leq s(i) \leq 1$. A value of $s(i)$ close to 1 is obtained when the within dissimilarity $a(i)$ is much smaller than the smallest between dissimilarity $b(i)$. Therefore we can say that object i is well clustered. On the other hand, if $s(i)$ take values close to -1 implies that $a(i)$ is much larger that $b(i)$. In this case we can say that object i has been misclassified, so object i may be reassigned. If $a(i)$ and $b(i)$ have similar values then $s(i)$ is about zero. In this situation object i lies equally far away from both cluster C_j and C_k.

Also it is possible to obtain the following summary values:

1. A cluster silhouette S_j, called average silhouette width, which represent the heterogeneity of cluster C_j. This quantitative measure is obtained by:

$$S_j = \frac{1}{m} \sum_{i=1}^{m} s(i)$$

2. The overall or global silhouette width, denoted by GS_U, and defined as:

$$GS_U = \frac{1}{c} \sum_{j=1}^{c} s(j)$$

Where U is any partition $U \longleftrightarrow C : C_1 \cup C_2 \cup ... \cup C_c$

This global silhouette value is used as a validity index for U. In order to choose the optimal number of clusters for a data set using this index, choose the partition U with the maximum GS_U.

Chapter 5

Experimental results

5.1 Data sets used in this work

We used three well known data sets among researchers in text mining and information retrieval. These datasets are described bellow.

5.1.1 Reuters-1 (RCV1)

In 2000, Reuters Ltd made available a large collection of Reuters News stories for use in research and development of natural language processing, information retrieval, and machine learning systems. This corpus, known as "Reuters Corpus, Volume 1" or RCV1, is significantly larger than the older, well-known Reuters-21578 collection heavily used in the text classification community. This is distributed on two CDs and contains about 810,000 Reuters, English Language News stories. It requires about 2.5 GB for storage of the uncompressed files.

In Fall of 2004, NIST took over distribution of RCV1 and any future Reuters

Corpora. These datasets can be obtained from NIST upon request.

This data set is described in detail by Lewis et. al. [19].

5.1.2 20 Newsgroups

The 20 Newsgroups data set was collected by Ken Lang, consists of 20017 articles divided almost evenly among 20 different UseNet discussion groups.

Some of the newsgroups are very closely related to each other (e.g. comp.sys.ibm.pc.hardware / comp.sys.mac.hardware), while others are highly unrelated (e.g misc.forsale / soc.religion.christian). Here is the list of the 20 newsgroups, partitioned according to subject matter:

comp.graphics

comp.os.ms-windows.misc

comp.sys.ibm.pc.hardware

comp.sys.mac.hardware

comp.windows.x.

rec.autos

rec.motorcycles

rec.sport.baseball

rec.sport.hockey

sci.crypt

sci.electronics

sci.med

sci.space

talk.politics.misc

talk.politics.guns

talk.politics.mideast

talk.religion.misc

alt.atheism

soc.religion.christian

misc.forsalehre

This data set is available from many online data archives such as CMU Text Learning Group, the UCI KDD Archive and more.

Many of the categories fall into confusable clusters.

5.1.3 WebKB

The WebKB data set described at (Craven et al) [4] contains 8145 web pages gathered from universit computer science departments. The collection includes the entirety of four departments, and additionally, an assortment of pages from other universities. The pages are divided into seven categories: student, faculty, staff, course, project, department and other.

5.2 Experimental Procedures

5.2.1 Sparse Representation

Let $|D|$ the number of documents in the collection D and $|V|$ the total number of different words (i.e the vocabulary lenght). In order to store the document vectors, we should need a matrix of order $|D| * |V|$.

However, most documents contain a small subset of the dictionary's words. In the case of text classification, a text corpus might have thousands of word types. Each individual document, however, has only a few hundred unique tokens. So, in the numerical vectors, almost all of the entries for that document will be zero. Rather than store all the zeros, it is better to represent the matrix as a set of sparse vectors, where a row is represented by a list of pairs, one element of the pair being a column number and the other element being the corresponding nonzero feature value. By not storing the zeros, savings in memory can be immense. Processing programs can be adapted to handle this format.

In this thesis, we use a relational database to store the sparse representation of the vectors. In this database, we have a main table that stores the document vectors called DocWord:

DocWord(Doc, Word, Weight) where:

Doc: A positive integer value corresponding to the Reuters-assigned document id.

Word: A positive integer word id representing one type of term.

Weight: The numeric feature value, i.e. within document weight, assigned to this term for this document.

Let nz the non zero weight values of the entire collection. With this representation, the storage growth is in the order of $O(3nz)$ which is more efficient than store all the matrix, since in this sparse context we have that $nz << |D| * |V|$.

The vectors weights for EM algorithm were term frequencies. To use in PAM algorithm, the document vectors were normalized to unit length and we used the $tfidf$

format, since we need to calculate a dissimilarity measure using the cosine function.

5.2.2 Preprocessing

In order to create the vectors from the text documents, we used the T2K (Text to Knowledge) library for Text Mining of D2K (Data to Knowledge) application environment for data mining. This software was developed by the Automated Learning Group (ALG) at the National Center of Supercomputing Applications (NCSA) of the University of Illinois. It is available for free under an academic license to academic and public institutions that have relationships with ALG at NCSA from this site: *http://alg.ncsa.uiuc.edu/do/downloads/d2k.*

The vectors obtained from this application were then converted to our sparse format using a Visual C++ code and exported to a relational database.

5.2.3 Dimension reduction

Even with attempts to develop efficient representational models, each document in a collection is usually made up of an exceedingly large number of features. The large number of features required to represent documents in a collection affects almost every aspect of a text mining system's approach, design, and performance.

For even the most modest document collections, the number of word-level features required to represent the documents in these collections can be exceedingly large. For example, in an extremely small collection of 15,000 documents culled from Reuters news feeds, more than 25,000 nontrivial word stems could be identified [7].

The data sets were preprocessed to remove stop words. Additionally, low-frequency words appearing in less than 0.2% of the documents, and high-frequency words appearing

in more than 15% were eliminated, as suggested in [7].

5.2.4 Clustering algorithms Implementation

We develop most of the entire experimental process using (Structure Query Language) SQL implementations in a MS SQLServer database. We implemented each algorithm as a stored procedure that make use of some views and other stored procedures. Also we create a function for the calculation of the cosine of two document vectors.

Once we have the data preprocessed, the algorithms were applied k times, to generate from 1 to k clusters, and then evaluate the best clustering.

The computer used for our experiments was a workstation DELL Precision 690. This system has a Intel Pentium Xeon processor running at 3.00GHz with 16GB RAM.

In order to run PAM algorithm it is necessary the creation of the dissimilarity matrix. This process is very expensive since we have to create a nxn matrix where n is the number of documents of the collection, and the entries of this matrix are the cosine between each document. Although, we just need to create the dissimilarity matrix once, in order to run the PAM algorithm several times over the same dataset.

In general, PAM algorithm along with the creation of dissimilarity matrix is more expensive in time than EM algorithm.

The applied dimension reduction techniques improved the running time since they reduced the number of words in the dictionary.

5.3 Validation Results

In this section, we give experimental results of the clustering process using the validation measures described in Chapter 4, Section 4.6.

Since our goal is to compare the performance of the two implemented algorithms and the effects of feature reduction and stop words removal, we run each algorithm to perform from 2 to 25 clusters and we reported the clustering validation measures for each cluster: silhouette index, entropy and purity.

Then, we report a summary in order to analize the better clustering results.

5.3.1 20 Newsgroups

For our experiments with the 20 Newsgroups dataset, we randomly selected 2000 documents from the entire collection. The sets were created with equal numbers of documents per class.

The clustering results for the value of k (number of clusters) from 2 to 25 are reported in table 5.1 for the PAM algorithm and in table 5.2 for the EM algorithm.

Table 5.3 shows that the better results were obtained with EM. Silhouette and Purity validity results suggests that the performance of the clustering improve removing stop words, althought the minimum Entropy was encountered just reducing dimensionality by removing 0.2 % of low frequency words and words that occur in more than 85% of documents.

A fact that it is important to note is that the better results closely occur with the actual number of clusters, 20 in this case, as we can see in table 5.3.

5.3.2 Reuters

For our experiments with the Reuters dataset, we randomly selected 1981 documents from the entire collection, wich have the particularity that belong to just one topic. The sets were created with the numbers of documents proportional to the entire collection.

The clustering results for the value of k (number of clusters) from 2 to 25 are reported in table 5.4 for the PAM algorithm and in table 5.5 for the EM algorithm.

Results show that the best results were obataining aplying EM algorithm. In this case the external validity measures suggest that it is not always a good idea to remove words. Entropy gives the best result considering all words and Purity gives the best result without removing stop words.

5.3.3 Web-Kb

For our experiments with the WebKb dataset, we used the four largest categories (excluding the category "other"): student, faculty, course and project; all together containing 4199 pages.

The clustering results for the value of k (number of clusters) from 2 to 10 are reported in table 5.7 for the PAM algorithm and in table 5.8 for the EM algorithm.

In this particular dataset, we found better results when not use a stoplist; the explanation for this is that the nature of the documents need this words to discriminate for example, "my" is an excellent indicator of a student homepage and is the fourth-ranked word by information gain [21].

PAM									
k	With all words			Without 0.2 low and 15 high			Stop words removed		
	Silhouette	Purity	Entropy	Silhouette	Purity	Entropy	Silhouette	Purity	Entropy
2	0.541	0.100	0.966	0.184	0.084	0.974	0.172	0.080	0.980
3	0.536	0.105	0.956	0.247	0.100	0.967	0.254	0.103	0.967
4	0.364	0.104	0.956	0.185	0.098	0.970	0.185	0.107	0.969
5	0.442	0.109	0.952	0.295	0.109	0.955	0.296	0.116	0.957
6	0.344	0.109	0.955	0.257	0.110	0.956	0.276	0.113	0.956
7	0.310	0.113	0.951	0.277	0.115	0.953	0.245	0.124	0.941
8	0.294	0.116	0.948	0.259	0.121	0.953	0.259	0.129	0.941
9	0.277	0.131	0.942	0.236	0.127	0.947	0.247	0.166	0.922
10	0.309	0.147	0.943	0.212	0.163	0.925	0.286	0.178	0.922
11	0.307	0.148	0.945	0.255	0.175	0.927	0.270	0.178	0.921
12	0.299	0.149	0.946	0.249	0.183	0.917	0.264	0.226	0.882
13	0.280	0.147	0.944	0.232	0.182	0.915	0.254	0.221	0.884
14	0.262	0.145	0.944	0.219	0.186	0.909	0.249	0.220	0.886
15	0.276	0.159	0.946	0.207	0.188	0.906	0.236	0.228	0.877
16	0.277	0.166	0.945	0.211	0.189	0.910	0.235	0.226	0.880
17	0.283	0.165	0.943	0.210	0.221	0.884	0.226	0.259	0.855
18	0.271	0.165	0.945	0.215	0.224	0.884	0.221	0.275	0.845
19	0.263	0.168	0.943	0.208	0.226	0.883	0.214	0.276	0.851
20	0.276	0.181	0.943	0.203	0.227	0.884	0.208	0.275	0.853
21	0.269	0.182	0.941	0.197	0.227	0.887	0.200	0.271	0.857
22	0.282	0.186	0.941	0.195	0.256	0.866	0.194	0.288	0.850
23	0.276	0.187	0.942	0.190	0.256	0.867	0.191	0.289	0.852
24	0.265	0.187	0.941	0.182	0.278	0.850	0.183	0.283	0.853
25	0.258	0.185	0.943	0.177	0.275	0.852	0.180	0.282	0.855
	0.541	**0.187**	**0.941**	**0.295**	**0.278**	**0.850**	**0.297**	**0.289**	**0.845**

Table 5.1: Clustering results for 20 Newsgroups dataset using PAM algorithm.

EM									
k	With all words			Without 0.2 low and 15 high			Stop words removed		
	Silhouette	Purity	Entropy	Silhouette	Purity	Entropy	Silhouette	Purity	Entropy
2	0.349	0.103	0.946	0.100	0.100	0.948	0.052	0.094	0.967
3	0.272	0.117	0.923	0.095	0.111	0.894	0.095	0.108	0.896
4	0.204	0.142	0.905	0.149	0.132	0.892	0.109	0.132	0.875
5	0.156	0.281	0.838	0.202	0.149	0.892	0.109	0.175	0.871
6	0.217	0.227	0.875	0.111	0.215	0.856	0.048	0.205	0.863
7	0.158	0.266	0.840	0.196	0.209	0.850	0.165	0.210	0.846
8	0.234	0.258	0.861	0.122	0.273	0.822	0.137	0.184	0.871
9	0.217	0.283	0.834	0.141	0.201	0.881	0.148	0.232	0.830
10	0.154	0.259	0.851	0.164	0.275	0.809	0.134	0.272	0.841
11	0.212	0.273	0.860	0.183	0.266	0.828	0.177	0.285	0.815
12	0.273	0.331	0.860	0.211	0.246	0.851	0.151	0.264	0.828
13	0.154	0.112	0.956	0.156	0.315	0.832	0.047	0.525	0.828
14	0.257	0.231	0.898	0.132	0.284	0.836	0.423	0.372	0.828
15	0.244	0.369	0.898	0.182	0.286	0.832	0.142	0.281	0.817
16	0.182	0.307	0.898	0.397	0.400	0.832	0.499	0.316	0.817
17	0.201	0.296	0.898	0.163	0.330	0.819	0.191	0.385	0.817
18	0.235	0.300	0.898	0.154	0.392	0.819	0.050	0.525	0.817
19	0.267	0.334	0.898	0.156	0.373	0.761	0.067	0.525	0.817
20	0.236	0.400	0.898	0.143	0.407	0.761	0.080	0.525	0.817
21	0.237	0.320	0.898	0.172	0.360	0.761	0.146	0.379	0.817
22	0.276	0.270	0.860	0.194	0.381	0.780	0.668	0.365	0.817
23	0.218	0.389	0.860	0.185	0.374	0.780	0.388	0.554	0.817
24	0.220	0.289	0.883	0.152	0.384	0.780	0.047	0.525	0.817
25	0.266	0.308	0.868	0.177	0.346	0.820	0.218	0.419	0.817
	0.349	**0.400**	**0.834**	**0.397**	**0.407**	**0.761**	**0.668**	**0.554**	**0.815**

Table 5.2: Clustering results for 20 Newsgroups dataset using EM algorithm.

	Silhouette	k	Purity	k	Entropy	k
PAM all words	0.541	2	0.187	24	0.941	22
EM all words	0.349	2	0.400	20	0.834	9
PAM after frequence reduction	0.295	5	0.278	24	0.850	24
EM after frequence reduction	0.397	16	0.407	20	**0.761**	**20**
PAM stop words removed	0.296	5	0.289	23	0.845	18
EM stop words removed	**0.668**	**22**	**0.554**	**23**	**0.815**	**11**
Best Results	**0.668**	**22**	**0.554**	**23**	**0.761**	**20**

Table 5.3: Summary of clustering results for 20 Newsgroups dataset.

PAM									
k	With all words			Without 0.2 low and 15 high			Stop words removed		
	Silhouette	Purity	Entropy	Silhouette	Purity	Entropy	Silhouette	Purity	Entropy
2	0.017	0.424	0.911	0.024	0.425	0.909	0.024	0.425	0.909
3	0.022	0.419	0.914	0.030	0.416	0.910	0.030	0.416	0.910
4	0.030	0.421	0.912	0.029	0.421	0.912	0.029	0.421	0.912
5	0.043	0.413	0.916	0.047	0.411	0.915	0.048	0.409	0.914
6	0.044	0.416	0.914	0.033	0.426	0.908	0.044	0.419	0.913
7	0.045	0.416	0.912	0.038	0.419	0.913	0.038	0.419	0.913
8	0.037	0.421	0.910	0.040	0.421	0.910	0.037	0.424	0.909
9	0.043	0.423	0.906	0.045	0.424	0.903	0.055	0.439	0.894
10	0.057	0.437	0.894	0.052	0.422	0.905	0.071	0.436	0.894
11	0.076	0.430	0.904	0.053	0.421	0.903	0.067	0.434	0.896
12	0.075	0.430	0.903	0.052	0.422	0.904	0.052	0.422	0.903
13	0.082	0.426	0.904	0.072	0.418	0.911	0.071	0.418	0.910
14	0.083	0.426	0.903	0.072	0.420	0.909	0.071	0.420	0.909
15	0.080	0.424	0.905	0.072	0.421	0.909	0.072	0.421	0.908
16	0.080	0.427	0.902	0.070	0.419	0.910	0.070	0.419	0.909
17	0.083	0.427	0.901	0.069	0.416	0.910	0.088	0.428	0.900
18	0.083	0.424	0.901	0.069	0.418	0.909	0.086	0.426	0.902
19	0.081	0.428	0.901	0.069	0.419	0.907	0.085	0.425	0.900
20	0.082	0.426	0.899	0.070	0.420	0.907	0.086	0.424	0.900
21	0.083	0.425	0.899	0.071	0.421	0.906	0.087	0.425	0.901
22	0.084	0.426	0.899	0.074	0.423	0.902	0.090	0.428	0.897
23	0.086	0.429	0.897	0.076	0.421	0.900	0.091	0.430	0.896
24	0.086	0.429	0.896	0.077	0.424	0.903	0.092	0.427	0.901
25	0.084	0.432	0.898	0.078	0.432	0.895	0.093	0.431	0.903
	0.086	**0.437**	**0.894**	**0.078**	**0.432**	**0.895**	**0.093**	**0.439**	**0.894**

Table 5.4: Clustering results for Reuters dataset using PAM algorithm.

EM									
k	With all words			Without 0.2 low and 15 high			Stop words removed		
	Silhouette	Purity	Entropy	Silhouette	Purity	Entropy	Silhouette	Purity	Entropy
2	0.027	0.430	0.909	0.029	0.427	0.910	0.029	0.428	0.909
3	0.027	0.427	0.907	0.025	0.429	0.908	0.023	0.427	0.910
4	0.023	0.427	0.909	0.019	0.430	0.908	0.029	0.431	0.907
5	0.027	0.435	0.904	0.028	0.423	0.913	0.021	0.430	0.906
6	0.037	0.435	0.901	0.028	0.430	0.907	0.026	0.432	0.905
7	0.034	0.427	0.897	0.028	0.423	0.912	0.037	0.426	0.904
8	0.058	0.445	0.891	0.028	0.431	0.904	0.032	0.430	0.905
9	0.045	0.465	0.906	0.024	0.417	0.911	0.029	0.430	0.905
10	0.089	0.427	0.916	0.051	0.712	0.911	0.186	0.379	0.955
11	0.058	0.459	0.881	0.246	0.462	0.956	0.043	0.429	0.906
12	0.070	0.464	0.888	0.044	0.712	0.956	0.138	0.462	0.956
13	0.083	0.461	0.888	0.038	0.712	0.956	0.051	0.712	0.956
14	0.073	0.503	0.888	0.051	0.712	0.956	0.138	0.462	0.956
15	0.022	0.426	0.897	0.275	0.462	0.956	0.390	0.453	0.961
16	0.095	0.461	0.908	0.051	0.712	0.956	0.495	0.586	0.961
17	0.074	0.569	0.908	0.498	0.586	0.956	0.056	0.712	0.961
18	0.086	0.468	0.886	0.198	0.712	0.956	0.476	0.712	0.961
19	0.100	0.551	0.886	0.053	0.712	0.956	0.056	0.712	0.961
20	0.143	0.453	0.923	0.059	0.712	0.956	0.239	0.462	0.956
21	0.128	0.583	0.923	0.069	0.512	0.941	0.499	0.586	0.956
22	0.201	0.559	0.923	0.049	0.712	0.941	0.239	0.462	0.956
23	0.123	0.531	0.923	0.038	0.712	0.941	0.060	0.712	0.956
24	0.120	0.502	0.923	0.059	0.712	0.941	0.239	0.462	0.956
25	0.120	0.502	0.923	0.246	0.462	0.956	0.038	0.712	0.956
	0.201	0.583	0.881	0.498	0.712	0.904	0.499	0.712	0.904

Table 5.5: Clustering results for Reuters dataset using EM algorithm.

	Silhouette	k	Purity	k	Entropy	k
PAM all words	0.085715349	23	0.43703356	10	0.894307854	10
EM all words	0.201358794	22	0.583299024	21	0.880573667	11
PAM after frequence reduction	0.078063999	25	0.432261766	25	0.895119103	25
EM after frequence reduction	0.498423276	17	0.712481051	18	0.903612557	20
PAM stop words removed	0.093476609	25	0.439239835	9	0.893738154	9
EM stop words removed	0.499435811	21	0.712373737	13	0.903747485	7
Best Results	0.499435811	21	0.712481051	18	0.880573667	11

Table 5.6: Summary of clustering results for Reuters dataset.

PAM									
k	With all words			Without 0.2 low and 15 high			Stop words removed		
	Silhouette	Purity	Entropy	Silhouette	Purity	Entropy	Silhouette	Purity	Entropy
2	0.66	0.46	0.89	0.07	0.43	0.91	0.11	0.45	0.89
3	0.35	0.43	0.91	0.12	0.45	0.90	0.10	0.42	0.91
4	0.36	0.46	0.89	0.10	0.46	0.89	0.10	0.43	0.90
5	0.41	0.48	0.87	0.10	0.47	0.88	0.11	0.42	0.90
6	0.34	0.47	0.87	0.11	0.45	0.89	0.11	0.43	0.90
7	0.29	0.47	0.86	0.11	0.46	0.88	0.12	0.42	0.91
8	0.29	0.48	0.85	0.12	0.44	0.89	0.12	0.42	0.90
9	0.27	0.49	0.86	0.13	0.47	0.86	0.11	0.44	0.89
10	0.26	0.50	0.85	0.13	0.44	0.88	0.12	0.46	0.86
	0.66	0.50	0.85	0.13	0.47	0.86	0.12	0.46	0.86

Table 5.7: Clustering results for WebKb dataset using PAM algorithm.

EM									
k	With all words			Without 0.2 low and 15 high			Stop words removed		
	Silhouette	Purity	Entropy	Silhouette	Purity	Entropy	Silhouette	Purity	Entropy
2	0.71	0.47	0.88	0.18	0.45	0.88	0.19	0.45	0.88
3	0.53	0.47	0.87	0.20	0.42	0.87	0.16	0.47	0.88
4	0.32	0.49	0.88	0.16	0.48	0.87	0.12	0.45	0.89
5	0.35	0.50	0.85	0.13	0.49	0.84	0.12	0.58	0.77
6	0.30	0.54	0.81	0.11	0.59	0.75	0.12	0.53	0.79
7	0.27	0.49	0.84	0.10	0.58	0.76	0.10	0.56	0.77
8	0.27	0.50	0.84	0.10	0.62	0.73	0.08	0.55	0.80
9	0.24	0.55	0.77	0.10	0.54	0.80	0.09	0.53	0.79
10	0.25	0.53	0.80	0.09	0.54	0.77	0.09	0.54	0.79
	0.71	0.55	0.77	0.20	0.62	0.73	0.19	0.58	0.77

Table 5.8: Clustering results for WebKb dataset using EM algorithm.

	Silhouette	k	Purity	k	Entropy	k
PAM all words	0.663	2	0.499	10	0.853	10
EM all words	0.706	2	0.547	9	0.774	9
PAM after frequence reduction	0.134	9	0.471	9	0.862	9
EM after frequence reduction	0.198	3	0.618	8	0.726	8
PAM stop words removed	0.123	7	0.457	10	0.861	10
EM stop words removed	0.192	2	0.580	5	0.769	5
Best Results	0.706	2	0.618	8	0.726	8

Table 5.9: Summary of clustering results for WebKb dataset.

Chapter 6

Conclusions and Future Work

In this thesis, we studied and implemented SQL versions of two document clustering algorithms: PAM and EM with multinomial Naive Bayes Mixtures. We compared the performance of these algorithms in three benchmark data sets: 20 Newsgroups, Reuters and WebKb under two dimension reduction techniques: removing low and high frequency words and removing stop words. For the evaluation of the clustering we used three cluster validity techniques: Silhouette, Purity and Entropy.

Like in data mining, in text clustering it is difficult to measure the performance of a particular clustering algorithm. In the case of Text Mining we can see that it depends on the data and the preprocessing technique applied.

The large number of features required to represent documents in a collection affects almost every aspect of a text mining system's approach, design, and performance.

In our experiments, we found that in general EM have better accuracy than PAM and in most cases removing low frequency and high frequency words and stop words improve

the accuracy. However, the full complexity of real world text data cannot be completely captured by known statistical models and dimension reduction efficiency depends on the nature of the documents

Among future work that can be done based on this thesis are:

in text clustering includes:

1. Improve the efficiency and scalability of the algorithms.

2. The dot product computation between the document vectors is the computational bottleneck in the generation of the dissimilarity matrix for the PAM algorithm and Silhouette measure, this process could be parallelized in order to improve time efficiency.

3. Investigate techniques like "truncation" and "Latent Semantic Indexing" in order to reduce the dimensionality and improve the performance and efficiency.

4. Parallelize the tokenization and vectorization.

5. Parallelize the EM algorithm for text clustering.

Bibliography

[1] P. Berkhin, Survey of Clustering Data Mining Techniques, Accrue Software, 2002.

[2] C. Borgelt and A. Nürnberger. Fast fuzzy clustering of web page collections. In Proc. of PKDD Workshop on Statistical Approaches for Web Mining (SAWM), Pisa, Italy, 2004.

[3] D. Cai, X. He, and J. Han. Document clustering using locality preserving indexing. IEEE Trans. Knowledge and Data Engineering, 2005.

[4] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. In Proceedings of the Fifteenth National Conference on Artificial Intellligence (AAAI-98). 1998.

[5] D. Cutting, D. Karger, J. Pedersen, and J. Tukey. Scatter-gather: A cluster-based approach to browsing large document collections. In Proceedings of SIGIR'92, 1992.

[6] S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. harshman. Indexing by latent semantic analysis. Journal of the American Society of Information Science, 41(6):391–407, 1990.

[7] I. Dhillon, J. Fan and Y. Guan. Efficient Clustering of Very Large Document Collections.In Data mining for scientific and engineering applications (pp. 357-381). Kluwer Academic Publishers.2001.

[8] I. Dhillon, S. Mallela, and D.S. Modha. Information-theoretic coclustering. In Proc. of the ninth ACM SIGKDD int. conf. on Knowledge Discovery and Data Mining, pages 89–98. ACM Press, 2003.

[9] W. Fan, L. Wallace, S. Rich, Z. Zhang. Tapping the Power of Text Mining. Communications of the ACM, 49(9):76-82, 2006.

[10] R. Feldman and J. Sanger. The Text Mining Handbook. Advanced Approaches in Analyzing Unstructured Data. Cambridge University Press, 2007.

[11] R. Feldman and I. Dagan. KDT - Knowledge discovery in texts. In Proc. of the First Int. Conf. on Knowledge Discovery (KDD), pages 112–117, 1995.

[12] N. Fuhr. Probabilistic models in information retrieval, The Computer Journal, v.35 n.3, p.243-255, June 1992

[13] M. Gonzales. A Comparison In Cluster Validation Techniques. Msc. Thesis, Mathematics Department, University of Puerto Rico at Mayaguez. 2005.

[14] Hearst, M. A. and Pedersen J. O. Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results. Proceedings of 19th ACM SIGIR Conference on Research and Development in Information Retrieval. 1996.

[15] A. Hotho, A. Nürnberger and G. Paaß. A Brief Survey of Text Mining, GLDV-Journal for Computational Linguistics and Language Technology, 20:1, pp: 19-62. 2005

[16] L. Kaufman and P. Rousseeuw. Finding groups in data: An introduction to cluster analysis. John Wiley & Sons, 1990.

[17] M. Konchady. Text Mining Application Programming. Charles River Media, 2006.

[18] J. Larocca, N. Alexandre, C. Kaestner, A. Freitas. Document Clustering and Text Summarization. Pontificia Universidade Catolica do Parana.Proceedings of the 4th International Conference on Practical Applications of Knowledge Discovery and Data Mining. 2000.

[19] D. D. Lewis; Y. Yang;T. Rose; and F. Li RCV1: A New Benchmark Collection for Text Categorization Research. Journal of Machine Learning Research, 5:361-397, 2004. http://www.jmlr.org/papers/volume5/lewis04a/lewis04a.pdf?? .

[20] D. Losada. A Logical Model of Information Retrieval based on Propositional Logic and Belief. PhD thesis. Universidad de la Coruña. 2001.

[21] K. Nigam, A. McCallum, and T. Mitchell. Semi-Supervised Text Classification Using EM in Semi-Supervised Learning, Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien (eds.), MIT Press, 2006.

[22] C. Ordonez and P. Cereghini. SQLEM: Fast Clustering in SQL using the EM Algorithm. In ACM SIGMOD Conference, pages 559–570, 2000.

[23] K. Papineni. Why inverse document frequency? In Proceedings of the NAACL, 2001.

[24] G. Salton, J. Allan, and C. Buckley. Automatic structuring and retrieval of large text files. Communications of the ACM, 37(2):97–108, Feb 1994.

[25] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. Information Processing & Management, 24(5):513–523, 1988.

[26] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. Communications of the ACM, 18(11):613–620, 1975.

[27] A.Tombros, R. Villa and C. van Rijsbergen. The effectiveness of query-specific hierarchic clustering in information retrieval. Information Processing and Management, 38(4), 559–582. 2002.

[28] M. Steinbach, Ge. Karypis, and V. Kumara. A comparison of document clustering techniques. In KDD Workshop on Text Mining, 2000.

[29] M. Steinbach, L. Ertoz, and V. Kumar. Challenges of clustering high dimensional data. In L. T. Wille, editor, New Vistas in Statistical Physics– Applications in Econophysics, Bioinformatics, and Pattern Recognition. Springer-Verlag, 2003.

[30] C. J. van Rijsbergen. A non-classical logic for information retrieval. The Computer Journal, 29(6):481–485, 1986.

[31] S. Weiss, N. Indurkhya, T. Zhang, and F. Damerau. Text Mining: Predictive Methods for Analyzing Unstructured Information. Springer, 2004.

[32] W. Xu and Y. Gong. Document clustering by concept factorization. In Proc. 2004

Int. Conf. on Research and Development in Information Retrieval (SIGIR'04), pages 202–209, Sheffield, UK, July 2004.

Appendix A

Source Code in SQL

```
CREATE PROCEDURE [PAM]
      -- Add the parameters for the stored procedure here
      @k int, -- number of clusters
      @max_iter int, -- maximum number of iterations
      @last_swap_cost float OUTPUT
AS
BEGIN
      delete from PAM_Medoids;

      update PAM_DocSel set Medoid = null, diss = null;

      DECLARE @i int;

      DECLARE     @oldMedoid int,
                  @newMedoid int,
                  @swapCost float;

      SET NOCOUNT ON;
      -- Build similarity matrix based on Document Selected
      -- in table PAM_DocSel
      EXEC PAM_BUILD @k;

      SET @i = 0;
      WHILE (@i < @max_iter)
      BEGIN
            -- Find the distance between objects and the second closest
medoid
            EXEC PAM_SetSecondMedoid;

            EXEC  PAM_EvaluateSwap
                        @oldMedoid OUTPUT,
                        @newMedoid OUTPUT,
                        @swapCost OUTPUT
            IF (@swapCost >= 0) BREAK;

            -- Carry out the swap
            EXEC PAM_SWAP @oldMedoid, @newMedoid;

            SET @i = @i + 1;
      END
      SET @last_swap_cost = @swapCost;
END

CREATE PROCEDURE [PAM_MakeDissMatrix]
AS
BEGIN
      SET NOCOUNT ON;
    DELETE FROM PAM_DissMat;
    INSERT INTO PAM_DissMat (Doc1, Doc2, Diss)
      SELECT D1.Doc, D2.Doc, (1 - [WebKb].[dbo].[Cosine](D1.Doc,
D2.Doc))
      FROM Pam_DocSel AS D1 CROSS JOIN Pam_DocSel AS D2
      WHERE D1.Doc < D2.Doc;
    INSERT INTO PAM_DissMat (Doc1, Doc2, Diss)
      SELECT Doc2, Doc1, Diss
      FROM PAM_DissMat;
END
```

```
CREATE PROCEDURE [PAM_BUILD]
    @k int
AS
BEGIN
    DECLARE @NewMedoid int;
      DECLARE @i int;
      SET NOCOUNT ON;

    DELETE FROM PAM_Medoids;
    -- Insert first medoid which the sum of the dissimilarities to all
      other objects is as small as possible
      SET @NewMedoid = (SELECT TOP 1 Doc1
                              FROM PAM_DissMat
                              GROUP BY Doc1
                              ORDER BY SUM(Diss));

      INSERT INTO PAM_Medoids (Medoid) VALUES (@NewMedoid);
      -- Assign first medoid to all other objects and calculate their
dissimilarity
      UPDATE PAM_DocSel
      SET   Medoid = @NewMedoid,
            Diss = D.diss
      FROM PAM_DocSel S inner join PAM_dissmat as D
            ON S.Doc = D.Doc1
      WHERE D.Doc2 = @NewMedoid;

    SET @i = 1;
    WHILE (@i<@k)
      BEGIN
            SET @NewMedoid = (
            SELECT TOP 1 Doc1
            FROM PAM_DissMat AS M inner join PAM_DocSel AS S
                  ON M.Doc2 = S.Doc
            WHERE M.Doc1 not in (SELECT Medoid FROM PAM_Medoids)
           AND M.Doc2 not in (SELECT Medoid FROM PAM_Medoids)
               AND S.Diss > M.Diss
            GROUP BY Doc1
            ORDER BY SUM (S.Diss - M.Diss ) DESC);

            INSERT INTO PAM_Medoids (Medoid) VALUES (@NewMedoid);
            -- Update table of Selected Docs with new medoid
            UPDATE PAM_DocSel
            SET Medoid = @NewMedoid,
                  Diss = M.diss
            FROM PAM_DocSel S inner join PAM_dissmat as M
                  ON S.Doc = M.Doc1
            WHERE M.Doc2 = @NewMedoid
              AND S.Diss > M.Diss
            SET @i = @i + 1;
      END -- while

      UPDATE Pam_DocSel SET Medoid = Doc, diss = 0
    FROM Pam_DocSel S inner join PAM_Medoids M
      on S.Doc = M.Medoid;
END
```

```
CREATE PROCEDURE [PAM_EvaluateSwap]
      @i int OUTPUT,
      @h int OUTPUT,
      @Tih float OUTPUT
AS
BEGIN
      SET NOCOUNT ON;
    -- Seleccionar el swap con minimo costo
      SELECT TOP 1 @i = i, @h = h, @Tih = SUM(Cijh)
      FROM PAM_Swap_All
      GROUP BY i,h
      ORDER BY SUM(Cijh);

END

CREATE PROCEDURE [PAM_SWAP]
-- Swaps medoid i with object h
      @oldMedoid int,
      @newMedoid int
AS
BEGIN
      SET NOCOUNT ON;

      -- Insert new medoid
      INSERT INTO PAM_Medoids (Medoid) VALUES (@newMedoid);

      -- Delete old medoid
      DELETE FROM PAM_Medoids WHERE Medoid = @oldMedoid;
      EXEC PAM_SetFirstMedoid;

      UPDATE Pam_DocSel SET Medoid = Doc, diss = 0
      --WHERE Doc = @newMedoid;
      WHERE Doc in (SELECT Medoid FROM PAM_Medoids);


END

CREATE PROCEDURE [PAM_SetFirstMedoid]
AS
BEGIN
      SET NOCOUNT ON;
      UPDATE PAM_DocSel
       SET Diss = FM.MinDiss
       FROM PAM_DocSel as S inner join PAM_FirstMedoidDiss as FM
            on S.Doc = FM.Doc
      WHERE Diss <> FM.MinDiss;

      UPDATE PAM_DocSel
       SET Medoid = M.Medoid
       FROM   dbo.PAM_DocSel AS S INNER JOIN
           dbo.PAM_DissMat AS D ON S.Doc = D.Doc1 INNER JOIN
           dbo.PAM_Medoids AS M ON D.Doc2 = M.Medoid
    Where S.Diss = D.Diss
      and S.Medoid <> M.Medoid
      and S.Doc not in (SELECT Medoid From PAM_Medoids);
END
```

```
CREATE PROCEDURE [PAM_SetSecondMedoid]
AS
BEGIN
      SET NOCOUNT ON;
      UPDATE PAM_DocSel
       SET Diss2 = SM.MinDiss2
    FROM PAM_DocSel as S inner join PAM_SecondMedoidDiss as SM
            on S.Doc = SM.Doc;

END

CREATE PROCEDURE [EM]
      -- Add the parameters for the stored procedure here
      @k int, -- number of clusters
      @max_iter int, -- maximum number of iterations
      @threshold float, -- OUTPUT
      @last_logl float OUTPUT
AS
BEGIN
      DECLARE @i int;
    DECLARE @ant_logl float;
      DECLARE @logl float;

      EXEC EM_InitRandom @k;
      -- since initialization assigns clusters
      -- we have to calculate initial parameters
    EXEC EM_MStep @ant_logl OUTPUT;

      SET @i = 0;
      WHILE (@i < @max_iter)
      BEGIN
            -- Find the distance between objects and the second closest
medoid

            EXEC EM_EStep;

            EXEC EM_MStep @logl OUTPUT;

            IF (abs( @ant_logl - @logl) <= @threshold) BREAK;

        SET @ant_logl = @logl;
            SET @i = @i + 1;
      END
      SET @last_logl = @logl;
END

CREATE PROCEDURE [EM_InitRandom]
  @k int -- number of clusters
AS
BEGIN
  SET NOCOUNT ON;
  delete from EM_DocSel;

  delete from EM_Cluster;

  delete from EM_Word;
```

```
  delete from EM_WordCluster;

  declare @ndocs int;
  declare @i int;

  insert into EM_DocSel (Doc)
  select Doc
  from EM_DocWord
  group by Doc;

  set @ndocs = (select count (*) from EM_DocSel);
  set @ndocs =  ceiling(cast(@ndocs as float)/@k);

  set @i  = 1;
  while (@i <= @k )
  begin
   insert into EM_Cluster (Cluster)
   values(@i);

   update EM_DocSel
   set Cluster = @i
   from EM_DocSel inner join (
   select TOP (@ndocs) Doc
   from EM_DocSel
   where Cluster is null
   order by NewId()) as S on EM_DocSel.Doc = S.Doc;

   set @i  = @i + 1;
  end

   insert into EM_Word (Word)
   select Word
   from EM_DocWord
   group by Word;

   insert into EM_WordCluster (Word, cluster)
   select W.Word, C.Cluster
   from EM_Word W cross join EM_Cluster C;

END

CREATE PROCEDURE [EM_MStep]
      @norm_param float OUTPUT
AS
BEGIN
   SET NOCOUNT ON;
   --  calculo de Zcj
   declare @ndocs int;
   declare @nclusters int;
   declare @denom int;
   declare @nwords int;

   set @ndocs = (Select count(*) from EM_DocSel);
   set @nclusters = (Select count(*) from EM_Cluster);
   set @denom = @nclusters + @ndocs;

   Update EM_Cluster
```

```
   set Pcj = log(CAST((1+n.nDocs) as float)/@denom)
   from EM_Cluster C inner join EM_nDocsxCluster n on C.Cluster =
n.Cluster;

   -- calculo de Xwt|cj
   set @nwords = (select count(*) from EM_Word);

   EXEC EM_SetClusternWords;
   EXEC EM_SetWordClusterWordt;

   Update EM_WordCluster
   set Pwtcj = log(CAST((1 + tj.nWordt) as float)/(@nwords +
tj.nWords))
   from EM_WordCluster WC inner join EM_WordtCluster tj on WC.Cluster =
tj.Cluster
        and WC.Word = tj.Word;

   -- Dar la norma del vector de parametros

   set @norm_param = (select sqrt(sum(power(Zi,2))) from EM_params);

END

CREATE PROCEDURE [EM_EStep]
AS
BEGIN
   SET NOCOUNT ON;
   UPDATE EM_DocSel
   SET Cluster = Es.Cluster
   FROM EM_DocSel AS D INNER JOIN
                     EM_EStepNewCluster AS ES ON D.Doc = ES.Doc ;
END

CREATE PROCEDURE [EM_SetWordClusterWordt]
AS
BEGIN
   update EM_WordCluster set nWordt = W.nWordt
   from EM_WordCluster WC inner join (
      SELECT     TOP (100) PERCENT dbo.EM_DocWord.Word,
dbo.EM_DocSel.Cluster, SUM(dbo.EM_DocWord.Weight) AS nWordt
FROM         dbo.EM_DocSel INNER JOIN
                     dbo.EM_DocWord ON dbo.EM_DocSel.Doc =
dbo.EM_DocWord.Doc
GROUP BY dbo.EM_DocSel.Cluster, dbo.EM_DocWord.Word
ORDER BY dbo.EM_DocSel.Cluster, dbo.EM_DocWord.Word
      ) as W on WC.Cluster = W.Cluster and WC.Word = W.Word;

END

CREATE  PROCEDURE [EM_SetWords]
AS
BEGIN
   insert into EM_Word (Word)
   select word
   from EM_DocWord DW inner join EM_DocSel S on DW.Doc = S.Doc
   group by word;
```

```
   insert into EM_WordCluster (Word, Cluster)
   select W.word, C.cluster
   from EM_Word W cross join EM_Cluster C;

END

CREATE PROCEDURE [EM_SetClusternWords]
AS
BEGIN
   update EM_Cluster set nWords = W.nWords
   from EM_Cluster C inner join (
      SELECT    dbo.EM_Cluster.Cluster, SUM(dbo.EM_DocWord.Weight) AS
nWords
      FROM         dbo.EM_DocWord INNER JOIN
                                        dbo.EM_DocSel ON
dbo.EM_DocWord.Doc = dbo.EM_DocSel.Doc INNER JOIN
                                        dbo.EM_Cluster ON
dbo.EM_DocSel.Cluster = dbo.EM_Cluster.Cluster
      GROUP BY dbo.EM_Cluster.Cluster
      ) as W on C.Cluster = W.Cluster;
END
```