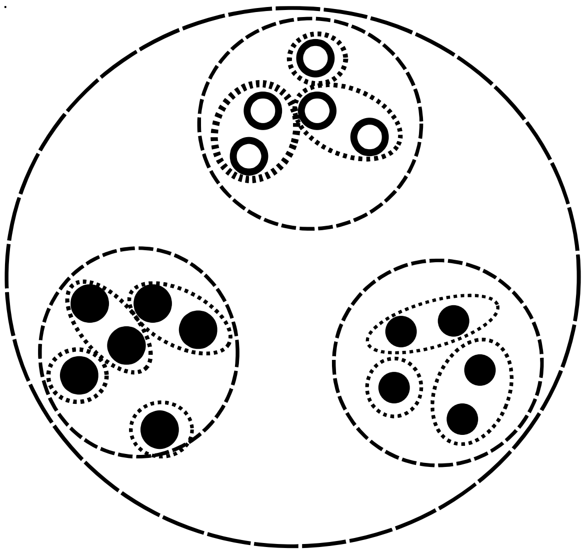

Document Clustering

Master of Science Thesis

Computer Science: Algorithms Languages and Logic

CHRISTOPHER ISSAL
MAGNUS EBBESSON

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, August 2010

Document Clustering
using entity extraction

CHRISTOPHER ISSAL
MAGNUS EBBESSON

© CHRISTOPHER ISSAL, August 2010
© MAGNUS EBBESSON, August 2010

Examiner: DEVDATT DUBHASHI

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover: Data points in different clusterings.

Department of Computer Science and Engineering
Göteborg, Sweden, August 2010

Abstract

Cluster analysis is a sub-field in artificial intelligence and machine learning that refers to a group of algorithms that try to find a natural grouping of objects based on some objective metric. In general this problem is hard because a *good* grouping might be subjective, two expert taxonomists can disagree on what they believe represents reasonable discriminatory features. The methods work directly on the data and are thus contained in the class of unsupervised algorithms contrary to classification algorithms whose bias is based on known classes. This report tries to give an overview to the application of clustering algorithms to text and how data might be processed.

Keywords: document clustering, text clustering, cluster analysis, cluster, unsupervised categorisation

Sammanfattning

Klusteranalys är ett delområde inom artificiell intelligens och maskininlärning som refererar till en grupp av algoritmer som försöker hitta naturliga grupperingar av objekt baserat på dess egenskaper. I allmänhet detta problem är svårt, eftersom en bra gruppering kan vara subjektiv, två experter inom taxonomi kan exmepelvis vara oense om vilka egenskaper de anser vara mest utmärkande. Dessa metoder som arbetar direkt på data och ingår därmed i klassen av oövervakade algoritmer vilka skiljer sig från mot klassificeringsproblemets algoritmer vars preferenser baseras på inlärd information. Denna rapport försöker ge en översikt över tillämpningen av kluster algoritmer till text och hur data kan bearbetas.

Keywords: dokumentklustring, textklustring, klusteranalys, kluster, oövervakad kategorisering

Acknowledgments

We would like to thank our colleagues at Findwise AB and especially our advisor Svetoslav Marinov for his enthusiastic support throughout the course of the project. We would also like to thank Martin Johansson and Pontus Linstörm for the friendly competition. Last but not least we would like to thank our supervisor Prof. Devdatt Dubahashi, Chalmers University of Technology, for taking us on.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

It is vital to remember that information — in the sense of raw data — is not knowledge, that knowledge is not wisdom, and that wisdom is not foresight. But information is the first essential step to all of these.

Arthur C. Clarke (2003) Humanity will survive information deluge

We are currently standing at the shoreline of the big sea of data that is the information age. And while this era's most precious resource is intellectual property and data there seems to be no bound to how much can be gathered. Digitalisation is booming and the big problem is how to interpret or analyse what is sampled, because the vast amounts surpasses what can be done by humans alone. Clearly we need powerful techniques to help us in this endeavor. The study of the much smaller problem only relating to text has spawned multiple fields in academia and industry such as computational linguistics, text data mining, information retrieval and news analytics etc.

Picture the hard working miner swinging his pickaxe, far below ground level, at tons after tons of rock in the pursuit precious jewels. This is the perfect analogy of the field of textual processing as it is today. Current methods have no concept of true *semantics*, the meaning of what it analyses, it does not understand words more than the symbols they are made of. Although it is debatable whether true meaning even exists and whether it is achievable by machines or whether it is just the resulting feeling of finding a match in the huge archives that is the human memory, this is better done elsewhere[42][12][23].

When processing text a lot of it is removed by sifting away useless rock to find the glittering bits. Because we have no concept of meaning except in very primitive cases our most sophisticated methods of today rely on single words and possibly their close neighbours while their most important information carriers, structure, grammar and semantics are mostly ignored. Even though there exist very good grammatical parsers, they can at most reconstruct syntactical structures to a degree.

When trying to understand any unknown data one of the most basic instincts for humans is look for patterns or structure. The leading question is *"What does these points have in common?* This leads us to our main topic of this report, grouping or unsupervised categorisation of textual data also known as document clustering. Clustering is one of the classic tools of our information age swiss army knife. Grouping is a blunt instrument in itself but it is a start that may lead to points of interest that require further analysis by humans. It narrows the search space because one is studying nearly structured data instead of a porridge random points. It can be thought of as the analogy of "lets plot and see what happens" method possible only when sampling data in a low dimensional space.

It may help as a guide when browsing or searching for knowledge [13] or serve as one of the core methods in automatic discovery of news articles[1]. Other known uses in industry is in market segmentation, plagiarism detection[31]etc.

[1]http://news.google.com

Chapter 2

Representation models

We have no idea about the 'real' nature of things ... The function of modeling is to arrive at descriptions which are useful.

Richard Bandler and John Grinder. (1979) Frogs into Princes: Neuro Linguistic Programming

The above quote sums everything up. A model is designed to represent the true nature of an object. As documents goes, they are quite well modeled to begin with. A document have has its topic, its sentences and its words that all together represents the documents A literate person could with ease make a good understanding of what the document represents, its meaning, the true nature of what it is about. Then repeat this for several other documents and group those documents that are alike. However in the world of computing, we have literate computers in the sense that they can parse text and read it back to the user, yet they have a hard time generalising the concept in the same manner as a human can. Whether or not this last is completely true can be debatable as there is no clear image of how humans do conceptualise. But that debate is outside the scope of this project.

Currently we do not have a good model of how humans conceptualise text, yet we have to find a representation that works for computers. The question is how? Can we assume that documents are bags of words and compare them with each other simply by looking at the contents of the bags? Bag of word perspective is the overall dominant document perspective in the field. While it is simple to implement and easy to work with, one must ask if it is enough? Text have structure, sentences, phrases that undoubtedly do contribute to the meaning of the text. Can these features be incorporated in our model in such a way that a computer can make sense of them?

In this chapter we will introduce different ways of modeling documents that somewhat addresses these questions.

2.1 Vector space

Introduced in the early seventies by Salton et al[40] as a model for automatic indexing, the vector space model has become *the* standard document model for document clustering.

In this model a document d can be interpreted as a set of terms $\{t_1 \dots t_n\}$. Each of these terms can be weighted by some metric (importance , occurrence etc). Given a weighting schema $W(d)$, d can represented by an n–dimensional vector $\mathbf{w}$.

The strongest motivation for a vector space representation is that it is easy to formulate and word with.

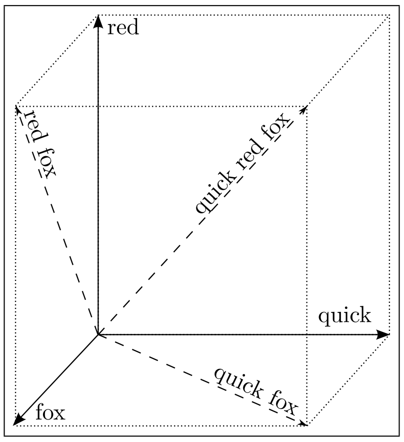

Figure 2.1: Illustration of a 3D vector space

In figure 2.1 illustrates a 3 dimensional vector space with the dimensions 'fox','red','quick'. The three clauses, 'red fox' 'quick red fox' and 'quick fox' represents points in that vector space. Example 2.1 demonstrate how three documents, $\{d_1, d_2, d_3\}$ can be calculated by using a simple term frequency weighting scheme.

Example 2.1:

d_1 : *The fox chased the rabbit*

d_2 : *The rabbit ate the cabbage*

d_3 : *The fox caught the rabbit.*

These three documents can be represented by a *co-occurrence matrix* as show in table 2.1. Each document is represented by a column vector the table.

	d_1	d_2	d_3
the	2	2	2
fox	1	0	1
rabbit	1	1	1
chased	1	0	0
caught	0	0	1
cabbage	0	1	0
ate	0	1	0

Table 2.1: Term-Document Co-occurrence matrix

The model is simple but it comes with a few drawbacks. Note that the words *the* and *rabbit* occur with the same frequency in all documents. As far as document features go, these terms are not discriminatory features for any of documents. As a consequence, they do not contribute to the separation of the documents. Increased separation will help distinguish similar documents. Imagine organising a bag of coins: Each coin is made out of the same metal and they all have a monetary value imprinted on them. It is quite obvious that the metal feature will not help significantly in the organisation of the coins, while the monetary value will.

To tackle the issue with non-discriminatory features it is common to apply a more delicate weighting scheme. From information retrieval we borrow *Term Frequency Inverse Document Frequency* or *tf-idf* for short. The *tf-idf* score for a term at position i in document j is computed as

$$(tf\text{-}idf)_{ij} = tf_{ij} \times idf_i \tag{2.1}$$

Where tf_{ij} is the term frequency for term i in document j. idf_i, is the inverse document frequency for a term t_i expressed as

$$idf_i = log \frac{|D|}{|\{d : t_i \in d\}|} \tag{2.2}$$

$|D|$ denotes the total number of documents in the corpus and the denominator, $|\{d : t_i \in d\}|$ are the number of documents in which therm t_i exists.

The *tf-idf*–weighting scheme will increase the weight of terms that have frequent occurrence in a smaller set of documents and lower the weight of those terms that are frequently occurring over the entire corpus. *tf-idf* is just one out of many different weighting schemes. We investigate how different weighting schemes affects clustering in section 5.5. [25, 27] give a good introduction to different weighting schemes.

The observant reader will notice that, even though each of the three documents only contains four unique terms, the length of the context-vectors are equal to the total number of unique terms in the corpus. This causes an apparent space efficiency problem. With each new unique term, the matrix grows by the size of the corpus. This calls for a more efficient representation for the model. From table 2.1 we can observe that we have more zero weighted features than non-zero features. These are call *sparse vectors.* The number of zero weighted terms will increase with the number of new terms entered in the document. Rather than using a matrix representation, we store only the non-zero terms in a *sparse matrix.* The sparsity of the model can be explained by the how the words are distributed in a language. The most frequently occurring word, occurs almost twice as frequent as the second most frequent word and so forth[39]. This phenomena was described by Zipf in 1949 and has known as *Zipf's law.*

A bit problem with high dimensional spaces is that points become more separated with each new dimension. This complicates similarity measure as points that appear to be similar in one subspace, might not be similar in another subspace. We can address this problem by reducing our feature space. This topic is further addressed in section 3.

2.1.1 Extensions

Standard vector model makes the assumption that all axis are pairwise orthogonal, i.e. terms are linearly independent. This assumption makes modeling simple but it badly reflexes natural languages, where terms (in general) are not linearly independent (synonyms etc). Figure 2.2 demonstrates this flaw.

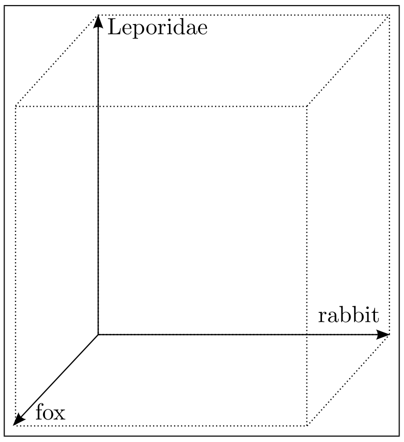

Figure 2.2: Leporidae is the Latin family name for rabbit, but is in the standard vector model assumes that rabbit is as close to fox as Leporidae

The consequence of the assumption be comes apparent when measuring similarities between documents. Documents that discuss to a common theme, but use different vocabulary, will be treated as dissimilar, when in reality the contrary is true. The *general vector space model* [46, 44] was introduced to address this. The idea is to expand the vector space model with a *term to term correlation* weight. When computing similarities between the documents, we introduce the correlation weight into the similarity metric. This allows us to overcome the orthogonal assumption.

Topic-based Vector Model (TBVM) introduces the concept of fundamental topics. Fundamental topics are defined to be orthogonal and assumed to be independent form each other. For our small fox and rabbit example, such topics might be *animals,vegetables,retrieval* etc.

This is a major difference to the standard vector space model. TBVM transforms documents to a d dimensional space R. Each term $t_i \in T$ is assigned a relevance towards each of these fundamental topics thus each term is represented by a fundamental topic vector, $\mathbf{t}_i$ in R. A term's weight corresponds to the length of $\mathbf{t}_i$. Now the model for a document d in a corpus D can be represented in *TBVM*, by a document vector $\mathbf{d} \in R$ (after being normalized to unit length) as follows:

$$\forall d \in D : \mathbf{d} = \frac{1}{|\boldsymbol{\delta}|}\boldsymbol{\delta}$$

where

$$\boldsymbol{\delta} = \sum_{t_i \in T} e_{ij}\mathbf{t}_i$$

where e_{ij} is the number of occurrences of term i in document j.

However while the original paper[5] proposes guidelines of how these fundamental topics should be picked, nothing specific is given. This was later addressed in the *Enhanced Topic-based Vector Space Model*[35] by using ontologies and several other linguistic techniques.

2.2 Graph model

While the popular vector space model can be considered the standard representation model, some authors suggest a different approach. [41, 19, 47] propose using graphs as representation. A graph is a set of vertices (or nodes) and edges usually denoted as

$G = \{V, E\}$, where V is the vertices and E is the set of edges. Edges represent some relation between vertices.

In contrast to the bag-of-words perspective used by the vector model, the common denominator among the different graph approaches is that the structural integrity of the documents in some sense, is preserved. The motivation behind analysing the structure becomes apparent when one considers the phrases in example 2.2.

Example 2.2:

"The fox chases the rabbit"

"The rabbit chases the fox"

The two phrases are semantically different but they have an equal bag-of-words representation *{the,fox,chases,rabbit}*.

Hammonda[19] proposes a *Document Index Graph* (DIG). A DIG is an directed graph. Each vertex v_i, represents a unique word in corpus. Each edge is between an ordered pair of vertices (v_i, v_j) if and only if v_j follows v_i in document in the corpus. Vertices within the graph keep track of which document the word occur in. Sentence path information i.e., what edge goes where and in what document is held in a separate index. Following this definition, a path from $v_1$to v_n represents a sentence of length n. Sentences that share sub-phrases will have shared paths in G. The degree of phrase matching between documents within the directed graph is later used to determine documents' similarities.

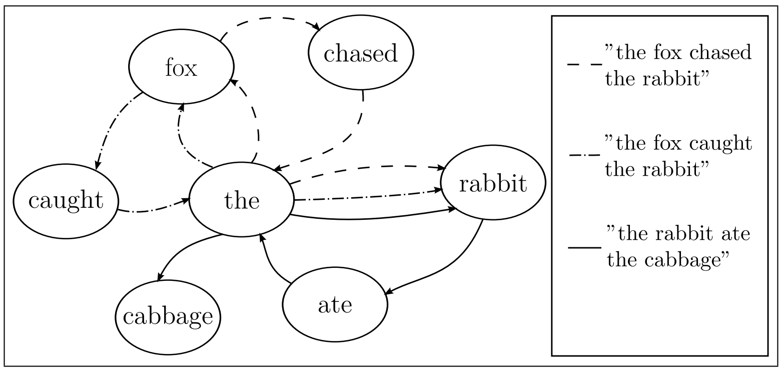
fox
chased
caught
the
rabbit
cabbage
ate
_ _"the fox chased the rabbit"
_._"the fox caught the rabbit"
___"the rabbit ate the cabbage"

Figure 2.3: A document index graph over three document

A variation on the same theme as DIG is the Suffix Tree Clustering. Introduced by Zamir and Ezzioni[47] as a mean for clustering web snippets from search engine results. As with DIG, the idea is to work with the phrase structure documents, or snippets. A suffix tree is a rooted directed tree. Documents are regarded as strings of words rather than characters. This structure holds the following properties:

- Each internal node has at least 2 children.
- Each edge on the tree is a labeled with a unique phrase from a document with in the corpus.
- No two edges from an edge begin with the same word.
- For each suffix s with in a document there exist suffix-node whose label is s.
- At leaf contains information about where its phrase originated from.

An example of a suffix tree is given in figure 2.4.

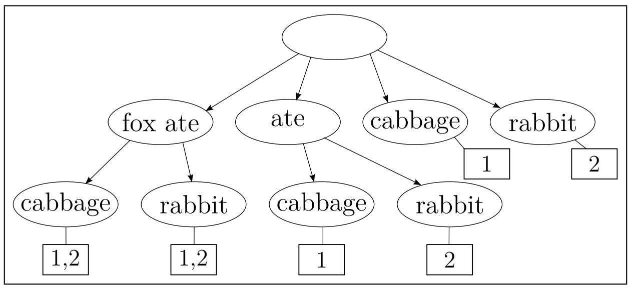

Figure 2.4: A STC over 'fox ate cabbage' 'fox ate rabbit'

While Zamir and Ezzioni focused on short web snippets, others such as Schenker[41] have focused on entire HTML pages. He proposes three different models to represent web documents. Unlike *DIG* and *STC,* each document is represented by a directed graph. Thus a corpus is a set of graphs. Each unique term in a document is represented by a node. A edge from vertex v_i to v_j exist if and only if term t_j succeeds term t_i in the text.

```
    For the Schenker's standard model, Schenker utilizes meta data from the HTML-tags to crea
```

In Schenker's simple model the dependencies are based on word order. The more sophisticated *n–Distance* model applies a *n* word look ahead (n-gram) where the labels on introduced edges are the distance in numbers of words from the first word in the sentence to the next node. These two models are illustrated in figure 2.5.

Edge labels are an important key in Schenker's way to cluster later on.

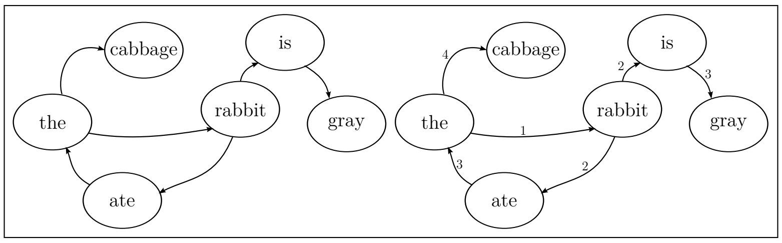

Figure 2.5: Left:Schenker simple model Right:Schenker *n*–gram model

2.3 Probabilistic topic models

Another perspective to document modeling is that they have been generated through some random process. Imagine that a document is constructed by first choosing a distribution of topics that your corpus shall cover e.g.$\{fox : 0.4, rabbit : 0.1, hunting : 0.3, cooking : 0.2\}$. Then from this distribution select a topic at random. Now each topic represents a distribution of words. By some mean select words to your document by drawing words from the topic. *E.g.,* let say we randomly pick the fox topic, then we would draw words such as *canid*, *sneaky*, fur etc. These words then make up you document. In general documents can be a generated from a mixture of topics. This *generative model* is the core idea behind topic models.

It was introduced by Hoffman in 1999[22], and has since been improved and modified[9, 10, 43].

The real work in topic modeling comes from that we usually do not know the topic distribution in a document nor the topic-word distribution. These unknown properties are called *hidden variables.* With the data that we have observed (the words of the document) we can use posterior inference methods to reveal the latent structure[9].

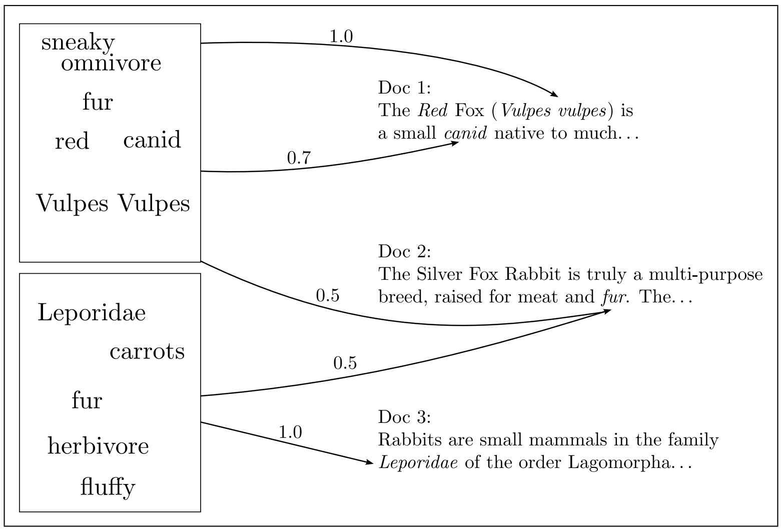
sneaky
omnivore
fur
red
canid
Vulpes Vulpes
Leporidae
carrots
fur
herbivore
fluffy
1.0
0.7
0.5
0.5
1.0
Doc 1:
The *Red* Fox (*Vulpes vulpes*) is a small *canid* native to much...
Doc 2:
The Silver Fox Rabbit is truly a multi-purpose breed, raised for meat and *fur.* The...
Doc 3:
Rabbits are small mammals in the family *Leporidae* of the order Lagomorpha...

Figure 2.6: Probabilistic generative process behind a topic model

Let us formalise (notation by [43]) the process to emphasize that it is a probabilistic model we are dealing with. Let $P(z)$ denote the distribution of topics z for a particular document d.

As previously mentioned, words within a document are assumed to be generated by first sampling a topic and then a word is drawn from that topic-word distribution. Let $P(z_i = j)$ denote the probability that the jth topic was sampled for the ith word in the document. $P(w_i|z_i = j)$ would then denote the probability that the word w_i was sampled under the jth topic as the ith word. Then the probability distribution over words within a document is denoted by

$$P(w_i) = \sum_{j=1}^{k} P(w_i|z_i = j)P(z_i = j) \tag{2.3}$$

k is the number of topics. In the literature, θ_d usually denotes the multinomial distribution of topics for a document d. This process is repeated for all documents in a corpus. To simplify this notation, graphical models (such as probabilistic topic models) are often described by a plate notation. The plate notation allows for demonstrating the conditional dependencies between an ensemble random variables. Example 2.1 demonstrates a simple conditional dependencies.

Example 2.1 The conditional dependence, with random variable $X = \{x_1, \ldots, x_n\}$ and $Y = y$

$$P(y, x_1, \ldots, x_n) = P(y) \prod_{n=1}^{N} P(x_n|y) \tag{2.4}$$

can be represented as

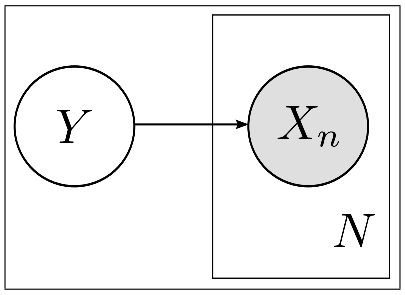

Figure 2.7: Plate notation of 2.4.

Nodes represent random variables. Shaded nodes are observed variables. Edges denotes the possible dependence. The plates represents the replicated structure. N is here the number of times the replicated structure is being repeated.

In general topic models follows the bag-of-word assumption. But some extensions have been proposed that handles word-order sensitivity. It has further been extended to be able to capture properties such as author and year of which the document might have been generated[9].

In some topic models documents are assumed to have been generated from a mixture of topics and can thus capture polysemy *i.e.,* word that have multiple meanings.

2.3.1 Latent Dirichlet Allocation

In *Latent Dirichlet Allocation* (*LDA*) the generative model is assumed to be as follows:

1. For each topic, pick a distribution over words: $\boldsymbol{\beta_i} \sim Dir_V(\eta)$ for $i \in \{1, \dots, k\}$
2. For each document d:
   (a) Draw topic distribution $\boldsymbol{\theta_d} \sim Dir_k(\boldsymbol{\alpha})$
   (b) For each word:
      i. Draw a topic assignment $Z_{d,n} \sim Mult(\boldsymbol{\theta_d}), Z_{d,n} \in \{1, \dots, k\}$
      ii. Draw a word $W_{d,n} \sim Mult(\boldsymbol{\beta_{Z_{d,n}}}), W_{d,n} \in \{1, \dots, V\}$

k is the number of topics and V is the size of the vocabulary. $\boldsymbol{\alpha}$ is a positive K–vector and η is a scalar. $Dir_V(\boldsymbol{\alpha})$ denotes a V–dimensional Dirichlet parametrised by $\boldsymbol{\alpha}$, and similar for $Dir_K(\eta)$. $Mult(\boldsymbol{\theta_d})$ is the *multinomial distribution* over topics for document d and $Mult(\boldsymbol{\beta_{Z_{d,n}}})$ is the multinomial distribution over words.

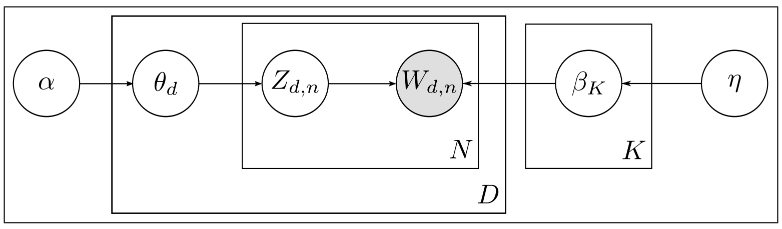

Figure 2.8: LDA process on plate notation

Recall that we have seen some of the elements before. What is added now for LDA is a specific assumption on how the multinomial distributions are generated. This is done by introducing a Dirichlet prior on $\boldsymbol{\theta_d}$. The Dirichlet distribution is *conjugate prior* for the multinomial distribution. This means that if our likelihood is multinomial with a Dirichlet prior, our posterior will also be a Dirichlet, thus simplyfing the statistical inference[43].

The elements $\{\alpha_1, \dots, \alpha_K\}$ of the hyper-parametric vector $\boldsymbol{\alpha}$ can be interpreted as a prior observation count for the number of times topic $j \in \{1, \dots, K\}$ that is sampled for a document (before having observed the words in the document!)[43]. It is convenient to assume that $\alpha_1 = \alpha_2 = \cdots = \alpha_K$, thus giving a symmetric Dirichlet distribution and even further reduce the inference complexity.

Similarly η can be interpreted as the prior observation count on the number of times words are sampled from a topic before having seen the words in the corpus. In the original paper suggests that values for $\boldsymbol{\alpha}$ and η can be given by a altering variational *expectation maximisation* (EM) procedure. [43] provides empirical values for α and η.

In section 4.9 we will present an inference method for *LDA*.

Chapter 3

Dimensionality reduction

The Cube which you will generate will be bounded by six sides, that is to say, six of your insides. You see it all now, eh? – Sphere explains Space to Square

Edwin A. Abbott. (1884) Flatland: A Romance of Many Dimensions.

High dimensional data can cause computational difficulties *e.g.,* similarities measuring. But how to deal with it? Does one really need M features to cluster N documents. In general $M \gg N$ for text. Perhaps not all features are needed? This is the assumption behind most dimensionality reduction strategies.

We can apply a supervised process that selects features matching certain criterions (usually by some filtering). We evaluate some of these techniques in sections (5.4) and (5.6). The supervised process is sometimes known as *feature selection.*

Alternatively we can apply an unsupervised process where features are extracted subject to some optimisation criterion. This unsupervised version is known to as *feature extraction.* Feature extraction performs a transformation from M–dimension space to a K–dimensional space ($M > K$). In the linear class we find *Factor analysis, Principle Component Analysis (PCA) Singular Value Decomposition*, *Latent Semantic Analysis, Random Projection* among others[15]. The non-linear methods include *Independent Component Analysis, IsoMap, kernel PCA* and *Kohonen Maps.* In this report we will address linear methods such as *PCA,SVD low-rank approximation* and *random projection.*

3.1 Feature selection

The literature is full of different ideas on what are good features for to select.

Stop-word-filtering is by far the most popular and perhaps simplest approach used in many document clustering applications. A stop word is usually a term that either is very frequently occurring such as *the, and* etc or has little or no contextual significance such as articles, prepositions etc. Stop-words are, from an information theoretic point of view, words with non or little information about the context of a document. The idea is than that by removing the stop-words, we can obtain a higher retrieval precision (5.2). A standard stop-word set's cardinality is the numbers of a few hundreds, hence it marginally reduces the dimensionality since languages (in general) contains a far greater number of words.

We can extend this this idea and assume that not just "frequent occurring words" are terms with high entropy, but that certain word classes or syntactic roles possesses a similar feature. *E.g.*

Example 3.1

> "The red fox caught the grey rabbit"

Here, the word "the", would be removed by simple stop word filtering. One can argue that we can even further reduce the sentence without losing a significant amount of the meaning example 3.1, by removing certain word classes. In this example the adjectives "red", "grey". Clearly we are able to understand the meaning of "fox caught rabbit" with the same ease. Similar assumptions can be made by certain syntactic roles such as accusative case.

We call this process of selecting word-classes for *part-of-speech tag* selection or *POS-tag* selection. From the field of linguistics we know that the smallest meaningful clause consists of a verb and a noun which carry the core meaning of a sentence. Additional classes only adds more nuance to the meaning. That is why in our dimensionality process we have chosen to retain only nouns and verbs.

We can extract words which have a certain syntactic role in the sentence such as subject, object and predicate etc. We call this process *SR-tag* selection.

Statistical reduction is a fairly common and simple approach. Remove those terms which are too common through out the corpus or too uncommon. The motivation for too common removal is equal to *tf-idf* motivation whereas too uncommon can be noisy data.

3.2 Principle component analysis

Principle component analysis or *PCA* for short dates back to 1901[34] and is a widely used technique for dimensionality reduction. It is a also known as *Karhunen-Loève* transformation. [7] gives two definition to *PCA* that boils down to the same algorithm. *PCA* can be interpreted as an linear projection that minimizes the average projection cost, where the average projection cost is the average squared distance between the data points and their projection[1]. The second interpretation is that *PCA* is the orthogonal projection of data onto a lower dimensional linear space (the principle subspace), such that the variance of the projected data is maximized.

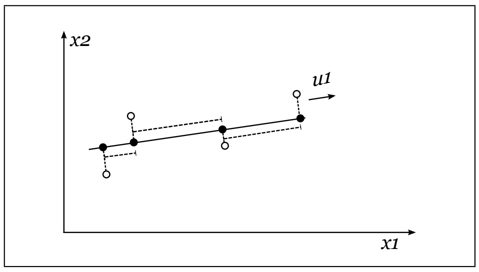

Figure 3.1: PCA from 2-d to 1-d. Either minimize the distance to u_1 or the maximize the variance between the points on the projection

PCA is about finding a set of k independent vectors to project or data on, but which one to choose? From derivations in A.1 we find that we need to calculate a covariance matrix representing the minimization error, given the first interpretation. Once we have the covariance matrix, calculate the k largest eigenvalues and their corresponding eigenvectors. These are the k first principle components.

[1]This is the definition given by Pearson[7, 34]

Computing all eigenvalues for a $D \times D$ matrix is expensive and has running time $\mathcal{O}(D^3)$, but since we only need the k first principle components, those can be computed in $\mathcal{O}(kD^2)$[7, 6]

3.3 Singular Value Decomposition

Singular value decomposition *(SVD)* can be used as a low-rank approximation for matrices.

[27] provides the following theorem defining *SVD*

Let r be the rank of an $M \times N$ matrix C then there is an singular valued decomposition of C on the form

$$C = U\Sigma V^T \tag{3.1}$$

where U is an $M \times M$ matrix whose columns are the orthogonal eigenvectors of CC^T. V is an $N \times N$ matrix whose columns are the eigenvectors of C^TC. An important property to note is that the eigenvalues $\lambda_1 \dots \lambda_r$ are the same for both CC^T and C^TC. Σ is an $M \times N$ diagonal matrix where $\Sigma_{ii} = \sigma_i$ for $i = 1 \dots r$ else 0 where $\sigma_i = \sqrt{\lambda_i}$, $\lambda_i > \lambda_{i+1}$. σ_i is called a singular value of C. In some literature Σ is expressed as a matrix of size $r \times r$ with the zero entries left out and the corresponding entries in U and V are also left out. This representation of the SVD is called *reduced SVD.*

We can use SVD to construct a low rank approximation, D_k, of D by following the Eckhart-Young theorem:

Algorithm 3.1 SVD low rank approximation

```
Construct the SVD of $D_r$ as in equation 3.1
From $\Sigma_r$ identify the $k$ largest singular values($k$ first, since $\lambda_i > \lambda_{i+1}$), and set the
$r - k$ other values to 0, yielding $\Sigma_k$
Compute $D_k$ from $U\Sigma_k V^T$. The reduced form is obtained by pruning row vectors of
length 0 in $D_k$
```

For optimality proof of D_k as an approximation of D see [27].

For sparse matrices of size $M \times N$ with c non-zero elements, an *SVD* can be computed in $O(cMN)$ time[6].

3.4 Random Projection

In random projection (RP) the goal is to project $M-$dimensional data on to a $k-$dimensional with the help of an *random matrix* R of size $k \times M$ whose columns are of unit length. If D is the original matrix of size $M \times N$, then

$$D_{RP} = R \times D \tag{3.2}$$

where D_{RP} is the $k \times N$ reduced data set.

While PCA and SVD are *data-aware,* i.e., they make reduction based on the original data, RP is said to be an *data-oblivious* technique as it does not assume any prior information about the data.[2].

Random projection relies on the *Johnson-Lindenstrauss lemma*[2, 6], which states that if points from a vector space are projected into a randomly selected subspace of substantially high dimension, the distances between the points are preserved with high probability[6]. If one is only interested in keeping the Euclidean properties of the matrix, it can be shown that the dimension size can be as low as the logarithm to the original matrix[2, 6]. With $k = log(N/\varepsilon^2)$ a distortion factor of $1 + \varepsilon$ can be achieved.

Strictly speaking, random projection is not a projection, since in general R is not orthogonal. Ensuring that R is orthogonal comes with a hefty computational price $O(n^3)$ for a $n \times n$ matrix.[17]. Not having orthogonality can cause some real distortions in the data set. However, due to the conclusions of Hecht-Nielsen[20] that in a high dimensional space there exists a much larger number of *almost* orthogonal than orthogonal directions, we can still apply the random projection with acceptable errors. Bringham experience mean square difference between $R^T R$ and I approximate to $1/k$[6]

When it comes to constructing R several different approaches have been proposed. The most general produces R by selecting k random M-dimensional unit vectors from some distribution, usually Gaussian or uniformed.[6].

Given R on the previous format, applying R to any vector cost $O(kN)$[2]. To address this Achlioptas proposes a much simpler distribution for r_{ij}[1, 2]:

$$r_{ij} = \begin{cases} -\sqrt{3}/N \; with\; probability & 1/6 \\ 0 & 2/3 \\ \sqrt{3}/N & 1/6 \end{cases} \tag{3.3}$$

In practice any zero mean,unit variance distribution of r_{ij} would give a mapping that satisfies the Johnson-Lindenstrauss lemma[6]. The speed up comes from the relative sparseness of R that allows for smart bookkeeping of the non-zero elements of R. This yields a three times speed up[2] compared to a Gaussian distribution, without marginal losses in performance[6]. Unfortunately the sparsity does not work well for all input. If input data is very sparse, D_{RP} might be null, due to the multiplications of $r_{ij}D_j$ might be zero.

Chapter 4

Clustering methods

But from a planet orbiting a star in a distant globular cluster, a still more glorious dawn awaits. Not a sunrise, but a galaxy rise. A morning filled with 400 billion suns, the rising of the milky way. – Carl Sagan

Carl Sagan. (1980) Cosmos: A Personal Voyage, episode 9, "The Lives of Stars"

A group of elements can be organised in a lot of different ways just like mathematical sets. Any specific set of clusters (a clustering) can be described using a few properties. Memberships can be either *exclusive* or *overlapping*, *fuzzy* or *binary.* Exclusive in this context means that an element can be a member of at most one cluster (this type of clustering is called *partitional*) while overlapping allows multiple (possibly nested) clusters. Fuzzy membership is a gradual quantity described by real value in the interval $[0, 1]$. This value could possibly be interpreted as a classification probability or closeness relative to the given cluster.

The problem of clustering can be expressed as an optimisation problem where one tries to minimise the distortion of choosing each cluster as a quantisation vector for its members. Solving this problem optimally is NP-hard (even in the simplest cases where $k = 2$, arbitrary d (see [3]) or $d = 2$, arbitrary k (see[26]) where k is the number of clusters and d is the number of dimensions) which means that in practice we have to approximate solutions. Hence, every algorithm described in the following section is heuristic, some with better guarantees than others. Most methods will also require a parameter k describing the sought number of clusters.

One can further organise clusters into hierarchies where higher order sets contain multiple, more detailed sets and so on as illustrated in figure 4.1. Algorithms usually work on data in an either *divisive* (top down) or *agglomerative* (bottom up) manner.

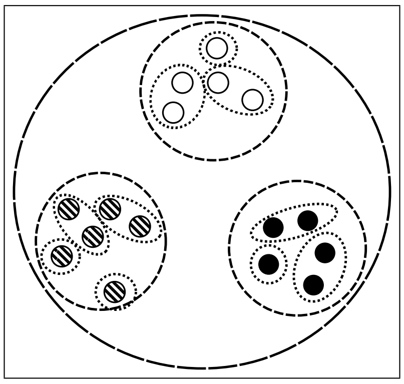
Figure 4.1: Example of hierarchical clustering

4.1 Outliers

When examining data in our n–dimensional space we may find elements whose measured features deviate greatly from the rest (see figure 4.2). This could indicate that they do not belong to any specific grouping or that portions of input is missing. These anomalies could be interesting in their own right depending on the users needs. They play a central role in fraud and intrusion detections systems.

Outliers do cause problems when performing *complete* clusterings, *i.e.,* when every point is assigned to some cluster, because true members are mixed with erroneous ones. This increases the cluster boundaries and may lead to merging of clusters bridged by outliers or inclusion of members from other clusters. In greedy methods such as single-link and complete-link described in section 4.5 our clustering would suffer the most. This error can be mitigated somewhat by using mean based criteria functions because this spreads it over all included members. If one knows that data contains a lot of noise it might be wise to run methods designed specifically for this purpose such as DBSCAN[14] or detect and remove outliers early.

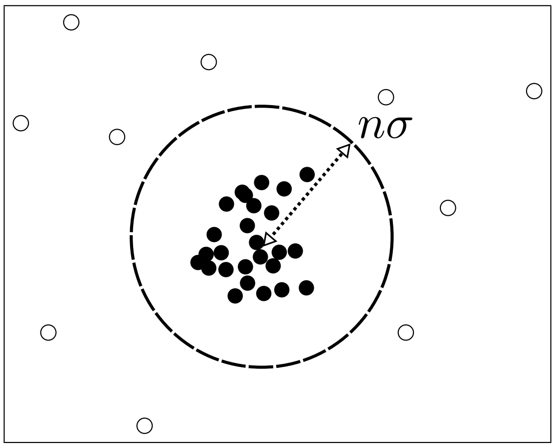

Figure 4.2: Example of outliers

4.2 Batch, online or stream

We can consider data in different ways before passing judgment on how we can cluster the data. The standard k–means (see section(4.4)) consider all data points at once and then calculates all clusters in one *batch.* .

Alternatively we can let the clusters do a bit of competition. Considering one input at a time, the clusters will compete for that input. The competition is usually based on some metric, e.g. closest distance. The winning cluster will be allowed to adjust itself to respond more tightly with the given input, making it more likely to capture similar inputs. When clustering is performed in this fashion it is called *online.* Compared to batch case, online methods are less sensitive to the initial clustering and they are never "done" in some sense.

We can also assume that data arrives in streams. We buffer a certain of the data, apply our clustering algorithm on this selection. That clustering can then be compacted in to a history vector that is weighted in at the next set of data, until all data is exhausted. Clustering on streams allows for non-stationary data in a much higher degree than the batch and online versions, which assumes stationary data.

4.3 Similarity measure

The choice of proximity function is a significant one because it will define what to interpret as clusters in the n–dimensional space that our documents reside in. We would like to achieve a good separation while keeping generality so we do not get either too small of too large clusters. The most intuitive metric is probably the Euclidian distance also known as the ℓ^2 norm of the difference vector. This is the direct distance between two objects in a linear space defined by

$$d_2(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2} \tag{4.1}$$

where $\mathbf{p}, \mathbf{q}$ are either points or vectors.

Other metrics based on linear space distances, although less common, are the the taxi cab distance (also known as the manhattan distance or ℓ^1 norm) and Chebyshev distance (ℓ^∞ norm or chessboard distance).

$$d_1(\mathbf{p}, \mathbf{q}) = \sum |p_k - q_k| \tag{4.2}$$

$$d_{chebyshev}(\mathbf{p}, \mathbf{q}) = \max_k |p_k - q_k| \tag{4.3}$$

More sophisticated distances include Mahalanobis distance that exploit correlations of the data set (which are unfortunately somewhat expensive to calculate). All above mentioned functions measure the *dissimilarity* between vectors which are all examples of the more general concept Bregman divergence.

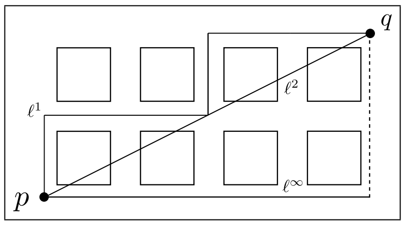

Figure 4.3: Geometric interpretation of ℓ^1, ℓ^2 and ℓ^∞norms of $\overrightarrow{pq}$ in two dimensions

The most common example of a true *similarity* function is the cosine similarity defined by the dot product between each feature vector

$$s_c(\mathbf{p},\mathbf{q}) = \mathbf{p}\cdot\mathbf{q} = \sum_{k=1}^{n} p_k q_k \tag{4.4}$$

An attractive property when micro-optimising is its simplicity or sparse vectors where you can skip any non-zero coordinates of either vector. In documents the number of nonzero elements can be as low as 0.1% or less, depending on the corpus, which makes this metric very cheap. The geometric interpretation is cosine of the angle between the position vectors of each point. This is commonly done with unit position vectors *i.e.,* each point resides on an n–dimensional unit hypersphere. Note that if no projectional dimensionality reduction is performed every point will reside on the strictly positive hyperoctant.

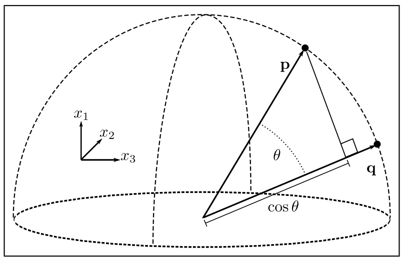

Figure 4.4: Geometric interpretation cosine similarity

A concept borrowed from statistics is Jaccard index or Jaccard similarity coefficient that measures similarity between sets of samples. The idea is that the intersection between the sets measure commonality which is normalised with the union of both sets.

$$J(A,B) = \frac{|A\cap B|}{|A\cup B|} \tag{4.5}$$

Another slightly modified weighting gives us a statistic called the Sørensen similarity coefficient which is usually called Dice's coefficient in IR literature or some variant of their names combined.

$$S(A,B) = \frac{2|A\cap B|}{|A|+|B|} \tag{4.6}$$

This idea of sets can be extended to feature vectors if we interpret nonzero features from each vector as a binary membership attribute. The intersection would then include dimensions where both features are nonzero while the union would include either and both. By combining cosine similarity with this way of thinking we get the Tanimoto coefficient

$$T(\mathbf{p},\mathbf{q}) = \frac{|\mathbf{p}\cdot\mathbf{q}|^2}{|\mathbf{p}|^{\mathbf{2}} + |\mathbf{q}|^{\mathbf{2}} - |\mathbf{p}\cdot\mathbf{q}|^{\mathbf{2}}} \tag{4.7}$$

which is a variant that yields the Jaccard similarity for binary attribute vectors.

4.4 Basic k–means

The most basic and probably intuitive way of clustering is to interpret each m–dimensional feature vector as a point in space. The basic idea is to use cluster prototypes called *centroids* to represent each partitions center in space and then assign elements based on their closeness. Each centroid is then updated with respect to the mean of these elements and the procedure is repeated until no elements are reassigned. Centroids, being means would not represent any object in our input set but some kind of continuous blend. In figure 4.5 the centroids are denoted by "+". A variant of this is if we require that centroids represent a real object then this algorithm becomes k–medioid.

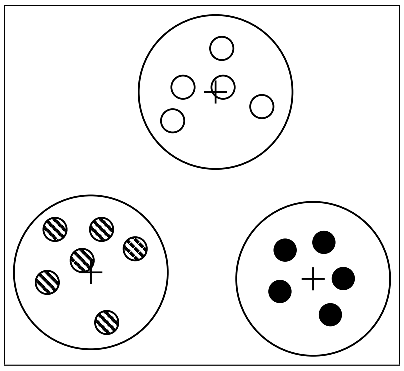
Figure 4.5: Three clusters with centroids

Algorithm 4.1 Basic k–means algorithm

```
Initialize $k$ centroids
repeat
  for all objects in input do
    Assign each element to its closest centroid
  end for
  for all centroids do
    Compute the mean of the assigned points
    This mean now becomes the new centroid
  end for
until all centroids remains unchanged or other termination criteria
```

The majority of the runtime is spent computing vector distances as can be seen from the description above. Every distance or similarity is computed *once* and then compared to the current best each pass of the assignement step. The new means consider every vector once as well. This leads us to believe that k–means is linear to the number of documents and the dimensionality. The centroids scale linearly as well with k. If we run it with a fixed number it iterations i the total runtime complexity would become $\mathcal{O}(iknd)$. In practice this is cheaper for document clustering because our vectors are very sparse, the true dimensionality of each vector is much lower than d.

By using a radius-based assignment scheme this algorithm will prefer clusters that are globular and hyperspherical in some dimension. Figure 4.6 demonstrates groupings that are not globular. This implies that k-means will fail to correctly group piecewise similar elements like the one below.

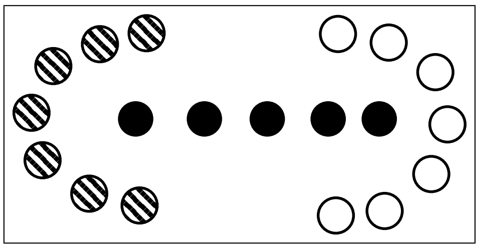
Figure 4.6: Groupings that are not globular

Being a heuristic method there are no guarantees that it will give an optimal clustering. The only guarantee we have is that it will not yield worse results than the initial conditions. Unfortunately this also implies that bad initial conditions might lead to a local optima like the example illustrated in figure 4.7.

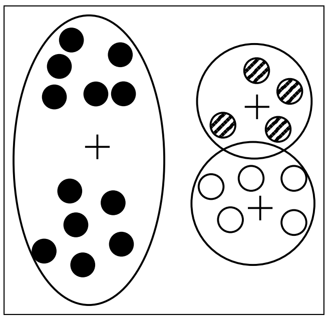
Figure 4.7: k–means stuck in a local optima

There have been a few proposals how to remedy this problem. Among them to perform a hierarchical clustering on a small subset to discover which data points are central to the found clusters and use these as initial centroids. Others suggest performing a number of trial runs each pass and choosing the one with that satisfies the global criteria best. One can also do a local search and try swapping points between clusters to see if that helps minimize the total distortion.

4.4.1 Bisecting k–means

This variant on k-means treats the input data as a single cluster and repeatedly splits the “worst” cluster in some sense until k clusters has been reached. Having a more general criterion function gives us some control over how we would like to decide what we considers the worst current cluster. Typical criteria are cluster size, total distortion or some other function of the current state. Splitting on size gives us more balanced cluster sizes.

If we assume the worst case in each split of n documents, *i.e.,* one document in one of the clusters and $n - 1$ documents in the other. This would mean that we have $k - 1$ splits and $\mathcal{O}(nk)$ similarity calculations. So in worst case this algorithm is as bad as normal k-means. But in practice this would be a very rare case indeed, with more balanced clusters should run faster, quite a bit faster.

Algorithm 4.2 Bisecting k–means

```
repeat
  Pick a cluster to split according to a criterion
  for i = 1 to $N$ do
    Bisect into two sub-clusters using $k$–means for $k = 2$
    Keep track of best candidate
  end for
until $k$ clusters remains
```

4.5 Hierarchical Agglomerative Clustering (HAC)

By treating each object as a cluster and then successively merging them until we reach a single root cluster we have organised the data into a tree. The pairwise grouping requires that we know the current best (according to some criteria) clusters which either forces us to calculate the similarity each pass or do it once through memorisation. The former method is not realistic in practice so calculating this *similarity matrix* is required and costs $\mathcal{O}(n^2)$ runtime and memory.

Algorithm 4.3 Hierarchical Agglomerative Clustering

```
Compute the similarity matrix
repeat
  Find two best candidates according to criterion
  Save these two in the hierarchy as sub clusters
  Insert new cluster containing elements of both clusters
  Remove the old two from the list of active clusters
until $k$ or one cluster remains
```

Single–link clustering is the cheapest and most straightforward merge criterion to use. We use the closest point in the both clusters to figure out the cluster similarity locally. In other words, the two most similar objects represent the similarity between the clusters as a whole. By sorting the values of the similarity matrix the merging phase can be done in linear time. This means that a total single link clustering runtime is only bound by the similarity matrix in $\mathcal{O}(n^2)$.

In a *complete–link clustering* the greedy rule instead tries to minimize the total cluster diameter. This makes the two furthest points in each cluster the interesting ones. This is a global feature that depends on the current structure and requires some extra computation in the merging phase. Running time for a complete–link rule is $\mathcal{O}(n^2 log\ n)$

The names single link and complete link come from their graph theoretic interpretations. We can define s_i as the similarity between the clusters merged at step i and $G(s_i)$ as the graph that links all clusters with similarity at least s_i. In a single–link clustering the state at step i are all the connected components of $G(s_i)$ and in complete–link the maximal cliques of $G(s_i)$.

Average–link clustering is a criterion that takes into account all similarities in each considered cluster instead of just the edges of each clusters. In other words the greedy rule tries to maximize cluster cohesion instead of diameter or local similarity. This however requires that we have the more information than just the similarity matrix because we need to calculate the mean of each cluster. In literature this method is also known as *group–average clustering* or *Unweighted Pair Group Method with Arithmetic Mean* (UPGMA). UPGMA is defined as

$$\frac{1}{|C_1| \cdot |C_2|} \sum_{\mathbf{x} \in C_1} \sum_{\mathbf{y} \in C_2} dist(\mathbf{x}, \mathbf{y}) \tag{4.8}$$

where C_1, C_2 are two separate clusters. Running time for UPGMA is $\mathcal{O}(n^2 log\ n)$

A great strength of HAC is its determinism, *i.e.,* always yields the same result from the same input. This means that we can at least depend on it to not fail miserably from bad initial conditions. The generated tree contains a clustering of every $k = 1, 2, \ldots, n$ so there is no need for this to be specified *a priori*. Unfortunately the memory requirement of $\mathcal{O}(n^2)$ is a big obstacle for this algorithm to be used in document clustering because it does not scale with large corpora. Some remedies of this has been suggested such as the Buckshot algorithm (used in the Scatter/Gather document browsing method) where one only clusters a sample size of $\mathcal{O}(\sqrt{nk})$[13].

4.6 Growing neural gas

This method stems from a group of algorithms drawing their concepts from self–organizing neural networks which can create a mapping from a high dimensional signal space into some (lower dimensional) topological structure. Being an extension of the Neural Gas [28] (NG), Growing Neural Gas [16] (GNG) is a competitive method that learns a distribution by drawing samples and then reinforcing the best matching particle; moving it closer to the signal.

The particles are represented as nodes in a graph where edges correspond to a topological closeness. The exploration is done in an incremental manner, instead of starting with k particles like NG one starts with only two and add a particle every λ steps. This new particle is inserted where it is currently "needed" the most, *i.e.,* at the node with the largest accumulated error and its worst neighbour.

Outliers are dealt with using an age associated with each edge which deprecates outdated network structures. When a node is selected as a best representative for a given signal one also increments the age of every neighbor edge. Outlier nodes are seldom selected as a best matching unit therefore it's edges will decay with time and become totally unconnected, after which the algorithm removes it totally. This also makes the method capable to learn moving distributions however some modifications are needed to make it more efficient at this task .

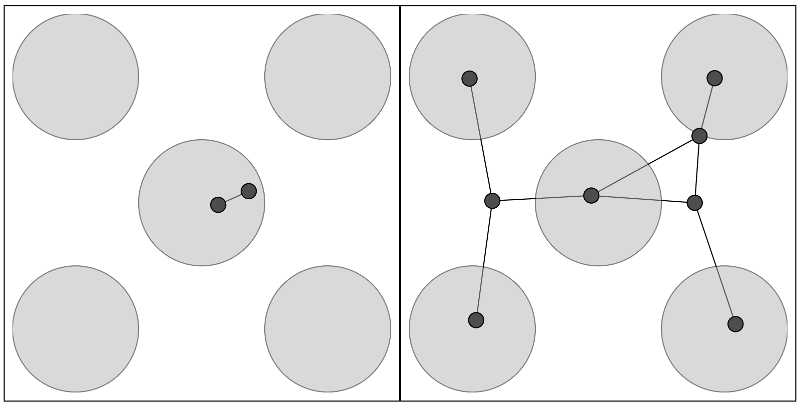

Figure 4.8: Growing neural gas after after initialization and after 1000 iterations

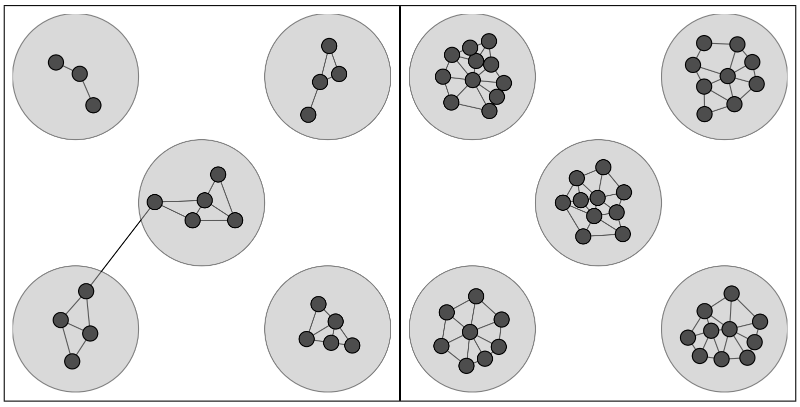

Figure 4.9: Growing neural gas after 3000 and 7000 iterations

Algorithm 4.4 Growing neural gas

```
Start with two neurons
repeat
  Draw signal from distribution
  Find best and second best matching units (bmu) and connect them
  Move bmu and neighbors closer to signal
  Age all touched local edges
  Remove edges older than $T_{max}$ and unconnected neurons
  if iteration is multiple of $\lambda$ then
    Find node with largest accumulated error
    In this node find its worst neighbor
    Insert new node between the connected nodes
    Connect these nodes to the inerted node and remove old edge
    Set error of new edges to half the largest and age to zero
  end if
  Decay error
until termination criterion
```

4.7 Online spherical k–means

This extension to the standard k–means uses document vectors normalised to unit length and cosine similarity to measure document similarity. When vectors are of unit length, maximising cosine similarity and minimising mean square distance are equivalent[48]. To ensure that our centroids remain on the hypersphere, the centroid must be normalised at the assignment step. In general centroids are not sparse, hence the normalisation step will cost $\mathcal{O}(m)$.

We can further extend our modification with an online scheme rather than a batch version.

With these two modifications of the standard k–means we have the *online-spherical k-means*(*OSKM*). The *OSKM* applies a winner take all strategy for its centroids. The centroid μ_i, being closest to the input vector $\mathbf{x}_i$ will be updated as

$$\mu_i' \leftarrow \frac{\mu_i + \eta(t)\mathbf{x}_i}{|\mu_i + \eta(t)\mathbf{x}_i|} \tag{4.9}$$

where $\eta(t)$ is the learning rate depending on iteration t

Algorithm 4.5 Online spherical k–means algorithm

```
Initialize unit length $k$ centroid vectors $\{\mu_1, \dots, \mu_k\}$,$t \leftarrow 0$
while termination criteria not reached do
  for each data vector $\mathbf{x}_i$ do
    find closest centroid $\mu_i = \arg\max_k \mathbf{x}_i^T \mu_k$
    Update $\mu_i$ accordingly
    $t \leftarrow t + 1$
  end for
end while
```

OSKM is essentially an incremental gradient ascent algorithm.

Zhong[48] describes three different learning rates. One which is inversely proportional to the cluster size, generating balanced clusters. The second is a simple flat rate ($\eta = 0.05$) and the third follows a decreasing exponential scheme:

$$\eta(t) = \eta_0 (\frac{\eta_f}{\eta_0})^{\frac{t}{NM}} \tag{4.10}$$

where N is the number of input values and M is the number of batch iterations. In [48] the later is empirically shown to lead to better clustering results.

The computational bottleneck is the normalisation of μ. If M batch iterations are performed the total time complexity for estimating all centroids is $\mathcal{O}(MNm)$. However by some cleaver bookkeeping and deferring of normalisation, the over all running time of $OSKM$ can be reduced to $\mathcal{O}(MN_{nz}k)$[48] where N_{nz} denotes the number of non-zero elements in the term-document matrix. To further speed up $OSKM$, Zhong suggest a sampling scheme. At each m, $(m < M)$ batch, sample $\frac{mN}{M}$ data points and adjust centroids after those. The motivation is that with an annealing learning rate (as earlier proposed), the centroids will move around much in the beginning and as the learning rate declines, the centroids will adjust more smoothly to the local data structure. With the sampling technique, Zhong achieves reduces runtime cost by 50%, without any significant clustering performance loss.

4.8 Spectral Clustering

Depending on one's background this method can be interpreted in a few different ways. What is clear though is that it works on a graph / matrix duality of the pairwise similarity. The *similarity graph* can be constructed using a few different methods generating different resulting graphs.

By only connecting points with a pairwise similarity greater than some threshold ε one gets the *ε-neighbourhood graph.* Because the resulting similarities are pretty homogeneous one usually ignores them and constructs an unweighted graph.

One can keep the k best matches for each vertex and construct a graph called the *k-nearest neighbour graph.* This graph becomes directed because this relationship might not be mutual. Either one can ignore the direction of the edges or enforce that only the vertices with a mutual k-best similarity are connected.

The last case is the fully connected graph where one keeps every cell of the similarity matrix and uses the closeness as the edge weights. The implications of the choice of graph construction method is still an open question however each of them are in regular use according to [45].

It was discovered that there exists a correspondence between bi-partitions and eigenvectors of the graph Laplacian. The Laplacian a matrix calculated from the similarity graph matrix and another feature called the *degree matrix* which is a diagonal matrix with the degree of each vertex in their respective cells.

With the similarity graph constructed one intuitively wants to find the minimum– or sparsest cuts of this graph to isolate k clusters. Depending on which cut one tries to approximate the last step has some variations. They all calculate k vectors that are fed into a k–means algorithm to produce the final clustering.

Algorithm 4.6 Spectral Clustering

```
  Compute the similarity matrix
  Compute the laplacian matrix L
  Find the $k$ first eigenvectors of L
  Construct matrix $A$ using eigenvectors as columns
  Partition into $k$ clusters with $k$–means using the rows of $A$ as initial centroids
```

4.9 Latent Dirichlet Allocation

With the representation given in 2.3.1 we now proceed with the inference step.

From clustering point of view the hidden variable θ_d is of most interest. We can either use θ to calculate similarities between documents and then use the similarities to cluster the documents or allow for the topics to be interpreted as clusters. In the later θ_d can then be interpreted as a soft clustering membership[9].

We infer its and the other hidden variables' *posterior distribution* given D observed documents from the following equation [9]

$$P(\theta_{1:D}, z_{1:D,1:N}, \beta_{1:K}|w_{1:D,1:N}, \alpha, \eta) = \frac{P(\theta_{1:D}, z_{1:D}, \beta_{1:K}|\mathbf{w}_{1:D}, \alpha, \eta)}{\int_{\beta_{1:K}} \int_{\theta_{1:D}} \sum_{\mathbf{z}} P(\theta_{1:D}, \mathbf{z}_{1:D}\beta_{1:K}|\mathbf{w}_{1:D}, z_n, \beta_{1:K})} \tag{4.11}$$

However, due to the coupling of the hidden variables θ and β, the denominator is intractable for exact inference[10]. Instead we have to approximate it.

Instead of directly estimating β and θ for each document one can estimate the posterior distribution over z, the per-word topic assignment, given $\mathbf{w}$, while marginalizing out β and θ. The reason is that they can be seen as statistics of the association between the observed words and the corresponding topic assignment[21].

Blei[10, 9] uses a mean field variational inference method for estimating z. The basic idea behind mean field variational inference is to approximate the posterior distribution with a simpler distribution containing free variables, hence turning it into an optimisation problem. This is done by decoupling the hidden variables and create independence between them. The independence is governed by a variational parameter. Each hidden variable can now be described by its own distribution.

$$Q(\theta_{1:D}, z_{1:D,1:N}, \beta_{1:K}) = \prod_{k=1}^{K} Q(\beta_k|\lambda_k) \prod_{d=1}^{D} \left(Q(\theta_{d_d}|\gamma_d) \prod_{n=1}^{N} Q(z_{d,n}|\phi_{d,n}) \right) \tag{4.12}$$

where λ_k is a V–Dirichlet distribution,γ_d is a K–Dirichlet and $\phi_{d,n}$ is aK–multinomial. As before V is the size of the vocabulary and K is the number of topics

With this at hand the optimisation problem is to minimise the *Kullback-Leibler* of the approximation and the true posterior.

$$\underset{\gamma_{1:D},\lambda_{1:K},\phi_{1:D,1:N}}{\arg\min} \quad KL(Q(\theta_{1:D}, z_{1:D,1:N}, \beta_{1:K})||P(\theta_{1:D}, z_{1:D,1:N}, \beta_{1:K}|w_{1:D,1:D}) \tag{4.13}$$

Algorithm 4.7 Mean field variational inference for LDAΨ is the digamma function, the first derivative of the log Γ function

```
$\gamma_d^0 := \alpha_d + N/k$ for all $d$
$\phi_{d,n}^0 := 1/k$ for all $n$ and $d$
repeat
  for all topic $k$ and term $v$ do
    $\lambda_{k,v}^{(t+1)} := \eta + \sum_{d=1}^{D}\sum_{n=1}^{N} 1(w_{d,n} = v)\phi_{n,k}^{(t)}$
  end for
  for all documents d do
    $\gamma_d^{(t+1)} := \alpha_k + \sum_{n=1}^{N} \phi_{d,n,k}^{(t)}$
    for all word $n$ do
      $\phi_{d,n,k}^{(t+1)} := exp\{\Psi(\gamma_{d,k}^{(t+1)}) + \Psi(\lambda_{k,w_n}^{(t+1)}) - \Psi(\sum_{v=1}^{V} \lambda_{k,v}^{(t+1)})$
    end for
  end for
until convergence for function 4.13 (see [9] for more details)
```

Each iteration of the mean field variational inference algorithm performs a coordinate ascent update. One such update has time complexity of $\mathcal{O}(knm + nm)$.

[10] gives (based on empiric studies) a value for the numbers of iterations needed until convergence. For a single document is in order of the numbers of words in the document. This gives approximate running time of $\mathcal{O}(m^2n)$

Once the algorithm has converted we can return estimate for θ_d,β_k and $z_{d,n}$.

$$\hat{\beta}_{k,v} = \frac{\lambda_{k,v}}{\sum_{v'=1}^{V} \lambda_{k,v'}} \tag{4.14}$$

$$\hat{\theta}_{d,k} = \frac{\gamma_{d,k}}{\sum_{k'=1}^{K} \gamma_{d,k'}} \tag{4.15}$$

$$\hat{z}_{d,n,k} = \phi_{d,n,k} \tag{4.16}$$

[43] uses Gibbs sampling to estimate z and then provides a estimate for θ_d and β_k. A thorough describtion of their procedure is given in [21].

Expectation propagation and collapsed variational inference are other approximation methods that have been evolved for *LDA*. The choice of approximation inference algorithm amount to trading of speed, complexity, assurance and conceptual simplicity[9].

Chapter 5

Results

The definition of insanity is doing the same thing over and over and expecting different results.

Rita Mae Brown. (1983) Sudden Death

In this section we would like to evaluate what kind of text processing that yields good results and how it affects the input space for the algorithms. As we will find later, just applying any clustering algorithm straight on the term frequency vector is not a very good idea for more than one reason.

5.1 Measuring cluster quality

While the clustering algorithms aims to optimise some target function, it is not clear whether this target function always divides the data into clusters that reflect the true nature of the data. The model might not be fully representative or biased in some negative way, or the documents express a greater depth than the algorithm can cover. Due to the nature of the problem it also follows that the perfect clustering is rarely know *a priori* (unless synthetic data) for general data. Thus there is often nothing to compare the clustering with. If we do have something to compare against, perfect clustering is highly subjective in the eye of the beholder. Especially when it applies to natural language data.

The literature provides a broad spectrum of evaluation methods and they can be either be supervised or unsupervised.

Unsupervised evaluation methods evaluate the internal structure of the clustering. One can calculate the density of the clusters by calculating the *cohesion* of each cluster with some distance function. Arguably a good clustering yields dense clusters. Alternatively one can investigate the average *separation* between clusters. A good clustering should provide a good separation of internal and external objects.

The good thing about unsupervised evaluation is that we do not require any complicated and detailed heuristic to evaluate the clustering. Unfortunately unsupervised method are highly algorithmic-dependent, *e.g.,* k–means returns globular clusters, then the *square-sum-error* might be good cohesion measure whereas applying the same metric on a clustering from a density based algorithm, the SSE might be catastrophic and vise versa.

Provided that we have some categorisation *a prioi* — a *gold standard* so to say, we can make more accurate judgments about the clustering provided by the algorithm. This is the supervised branch.

5.1.1 Recall, precision and F-measure

From information retrieval comes metrics such as *recall* and *precision.* We can interpret a clustering as a retrieved set documents given a query, then recall is defined as the proportion of relevant documents retrieved out of all relevant documents. Precision can be interpreted as the proportion of retrieved and relevant documents out of the retrieved documents[4].

An alternative interpretation is that clustering is a series of decisions, one for each pair of documents in the corpus[27]. The decision is whether or not to assign the two documents to the same cluster given their calculated similarity. A *true positive (TP)* is the decision that has assigned two similar documents to the same cluster. A *true negative (TN)* assignment is when we have assigned two dissimilar documents to two different clusters. If we assign two dissimlar documents to the same cluster we have a *false positive (FP)* and two similar documents to different clusters we have a *false negative (FN).*

With these interpretations in mind we can formally define recall as

$$recall = \frac{|retrieved \cap relevant|}{|relevant|} = \frac{TP}{TP + FN} \tag{5.1}$$

and precision

$$precision = \frac{|retrieved \cap relevant|}{|retrieved|} = \frac{TP}{TP + FP} \tag{5.2}$$

The observant reader will notice that one can achieve perfect recall by simply gathering all documents within one cluster. To cope with this flaw. It is practice use precision and recall in a combined metric known as *F-measure* or *F-score.*

$$F_\beta = \frac{(\beta^2 + 1) recall \cdot precision}{recall + \beta^2 \cdot precision} \tag{5.3}$$

where β controls the penalisation of false negatives, by selecting $\beta > 1$.

To apply F-measure to clustering we assume that we have a perfect clustering *i.e.,* a set of classes $\mathbb{C}$ and set of clusters Ω given by the clustering. The F-measure for a cluster j can be calculated as[4]:

$$F_\beta = \sum_{c \in \mathbb{C}} \frac{|c|}{|D|} \arg\max_{\omega \in \Omega} F_\beta(\omega, c) \tag{5.4}$$

where $|D|$ denotes the total size of the corpus and $|c|$ denotes the size of the class. F-measure addresses the total quality of the clustering from an information retrieval perspective. A perfect F-score (1.0), indicate a perfect match between the classification set and the clustering. While F-measure is quite popular in the information retrieval community its application to evaluate clustering can be questioned. F-measure does not address the composition of the clusters themselves[4]. It also requires that $|\mathbb{C}| = |\Omega| = k$ and this puts limitations on the clustering algorithm – it must return a fixed number of clusters.

5.1.2 Purity, entropy and mutual information

From the information theoretic field comes *purity, entropy* and *mutual information.*

Purity measures the dominance of the largest class per cluster.

$$purity(\Omega, \mathbb{C}) = \frac{1}{|D|} \sum_{\omega \in \Omega} \max_{c \in \mathbb{C}} |\omega \cap c| \tag{5.5}$$

A perfect clustering will yield a purity of 1 and bad close to $1/k$. It can be shown that purity encourage a clustering with high cardinality. If $|\Omega| = |D|$ we will get a perfect purity, but intuitively we would appreciate lower cardinality on our clustering.

Entropy is a measure on uncertainty about the distribution of an random variable. The literature provides several, slightly different interpretations of entropy in relation to clustering. We can either consider the probability of a document being in a specific cluster and class or by the probability of a document being in a specific cluster regardless of class. The former definition can be expressed as [4]:

$$entropy(\Omega, \mathbb{C}) = -\sum_{\omega \in \Omega} \frac{|\omega|}{|D|} \sum_{c \in \mathbb{C}} P(c, \omega_j) \log P(c, \omega_j) \tag{5.6}$$

where $P(c, \omega_j)$ is the probability of a document j from cluster ω belongs to classification c. The second interpretation given by[27] does not consider the actual class. It can be written as:

$$H(\Omega) = -\sum_{\omega \in \Omega} P(\omega_j) \log P(\omega_j) \tag{5.7}$$

where $P(\omega_j)$ is the probability of a document being in cluster ω.

The *mutual information* (MI) between a clustering and a classification is a metric on the amount of information Ω and $\mathbb{C}$ share. Given the information about a document being in a particular cluster gives us some information about what class that cluster might be. As we learn more about what documents are in that specific cluster the more certain we get about its actual class. MI measures how much our knowledge about classes increases as we learn about the clustering. This works in both ways. Knowing more about one of the input variables reduces the uncertainty about the second and vice versa.

It has strong relations to entropy and can be expressed in terms of entropy. [27] defines mutual information as

$$MI(\Omega, \mathbb{C}) = \sum_{\omega \in \Omega} \sum_{c \in \mathbb{C}} P(c, \omega_j) \log \frac{P(c, \omega_j)}{P(\omega_j) P(c)} \tag{5.8}$$

Minimum mutual information is 0. Then the documents in a specific cluster cannot say anything about what classification that cluster is. Maximum mutual information is reached when the clustering exactly matches the reference classification. More over, if Ω is divided into sub-clusters of the perfect matching we will have maximum mutual information. Consequently MI, does not penalise low cardinality. To address this, it is practical to use the normalised version of mutual information *normalised mutual information (NMI)*. NMI ensures that large cardinality is penalised. The literature describes geometric interpretations on the normalisation factor $\sqrt{H(\Omega) \cdot H(\mathbb{C})}$[48, 4], but also as a arithmetic mean between the entropy of the Ω and $\mathbb{C}$, *i.e.,*$|H(\Omega) + H(\mathcal{C})|/2$[27]. The normalisation factor also restricts the metrics upper value to 1.

$$NMI(\Omega, \mathbb{C}) = \frac{MI(\Omega, \mathbb{C})}{\sqrt{H(\Omega) H(\mathbb{C})}} \tag{5.9}$$

It has been empirically demonstrated that *NMI* is the superior to purity and entropy as a measurement for document clustering[4].

With these evaluation methods available, we have chosen work with *NMI* for evaluation in combination with purity as we for most of the evaluated algorithms have to provide a fixed k clusters.

5.1.3 Confusion matrix

Table 5.1 is a confusion matrix over clusters A to K. The category distribution is given by the columns of the matrix.

	Clusters										
Categories	A	B	C	D	E	F	G	H	I	J	K
News	327	3	109	3	2	9	1	90	26	170	19
Economy	137	1	2	-	-	-	-	3	2	5	3
Consumer	120	3	2	142	7	-	212	-	-	35	-
Entertainment	30	-	-	-	-	3	3	6	216	77	216
Food	3	138	9	35	100	-	2	-	-	24	-
Living	26	-	-	-	1	-	1	-	-	67	-
Sports	4	-	2	-	-	384	3	33	-	15	5
Travel	-	1	-	1	-	-	-	-	-	25	-
Job	18	-	-	-	-	-	-	-	1	33	-
Auto	14	-	-	-	-	-	20	-	-	78	-
Fashion	2	-	-	-	-	1	3	-	-	14	-

Table 5.1: Confusion matrix for GP for clustering with NMI 0.50

The confusion matrix above demonstrates an important caveat in all supervised clustering evaluation. We have to assume the categories are separable and disjoint. The above clustering is on a collection of news articles. The confusion matrix reflects that categories like “News” and “Economy” are similar and will use the similar features. A similarity that human readers can agree to whereas “Sport” can be easily identified a different vocabulary.

5.2 Data sets

In the scope of our project we have chosen to work with two rather different corpora — one Swedish newspaper corpus and one English newsgroup corpus. It has been argued that Swedish is more difficult to cluster due to language specific features of Swedish (compounds,homonyms etc)[38] and this has been something that we would like to cover as well.

The English data set is the popular collection called *20 Newsgroup*[1]. The original collection contains 19997 Usenet discussions crawled from 20 different newsgroup boards with topics covering computer science, politics, religion, etc.

The documents are almost evenly distributed over the different newsgroups. There is an modified version where duplicated and cross-post have been removed, which we have used throughout our work. The headers only contains the *from* and *subject* fields and this reduced data set contain 18828 documents. We consider the discussion board topics as our golden standard clustering.

[1]Available at `http://people.csail.mit.edu/jrennie/20Newsgroups/`

Topic	#
alt.atheism	799
comp.graphics	973
comp.os.ms-windows.misc	985
comp.sys.ibm.pc.hardware	982
comp.sys.mac.hardware	961
comp.windows.x	980
misc.forsale	972
rec.autos	990
rec.motorcycles	994
rec.sport.baseball	994

Topic	#
rec.sport.hockey	999
sci.crypt	991
sci.electronics	981
sci.med	999
sci.space	987
soc.religion.christian	997
talk.politics.guns	910
talk.politics.mideast	940
talk.politics.misc	755
talk.religion.misc	628

Table 5.2: Category distribution NG20

Our second corpus is crawled from the online version of the local news paper *Götebors-Posten*[2] *(GP).* GP contains 3049 documents distributed over 11 categories about news, economics, sport etc. Documents have been manually categorised by the editor of the news paper. The categories have been selected from the topology of the web site. In contrast to NG20, the GP corpus is fairly unbalanced in the distribution of documents per category.

Topic	#
Living	97
Economy	153
Job	52
Consumer	519
Entertainment	553
Food	311

Topic	#
Fashion	20
Auto	112
News	759
Travel	27
Sports	446

Table 5.3: Category distribution GP

	GP	NG20
N	3049	18828
$\sum n_d$	1307287	7141855
$\frac{1}{N}\sum n_d$	429 ± 399	379 ± 1182
$(\min n_d, \max n_d)$	$(4, 7425)$	$(7, 71337)$
Balance	0.03	0.63
NNZ	653813	3055221
dim	93656	177868
k	11	20

Table 5.4: Statistics about the corpora

In table 5.4, the notion of *balance* follows from [48] and is defined as the ratio between the smallest category and the largest category. It can give a hint on how hard the corpus is to cluster (depending on algorithmic preference). The lower balance value is worse. D denotes the set of documents in the corpus. N is the cardinality of D. The cardinality of a document $d \in D$ is denoted by n_d and it corresponds to the number of terms within that document. *dim* is the number of unique terms throughout the corpus and *NNZ* are

[2] `http://www.gp.se`

the total number of non-zero elements in the corresponding term-document matrix. k are the number of classes.

After textual processing (described in 5.3) we retain slightly different statistics about the corpora as shown in table 5.5.

	GP	NG20
N	3049	18828
$\frac{1}{N}\sum n_d$	89 ± 64	63 ± 70
$(\min n_d, \max n_d)$	$(1, 514)$	$(1, 1236)$
NNZ	272677	1193420

Table 5.5: Statistics about the corpora after textual processing

5.3 Experimental setup

In each experiment run two types of tests. First we do a baseline and see how each parameter affects the input space and/or the cluster solution with any other processing turned off *i.e.*, the only textual processing is tokenisation and word counting. These runs do in general not perform very well, but the intent is to show the relative improvements not any absolutes. Please note that the parameters investigated are probably **not** linear in cluster quality with each other there seems to be some dependencies. This is why we do a second run with some more "sane" values to see how each parameter affects a "real world" clustering.

The second type of setup involves "good" values for all parameters, some of which are supported in literature others that we have discovered work well. The following settings are used except for the parameter investigated in the experiment at hand. We remove terms occurring in more than 60% of the corpus (section 5.4.3) as well as terms occurring in less than 7 documents (section 5.4.2). Stop words are removed and only terms in the lexical classes noun, proper noun and verbs are kept (section 5.6). All frequencies are weighted with the $\sqrt{tf} \times idf$ scheme (see section 5.5). After this only the 7000 most common terms are kept to keep down runtime costs (section 5.4.1).

For each experiment we generate the document matrix and then perform 100 clusterings using the bisecting k–means with 10 trials each pass. The motivation for bisecting k–means is its runtime performance and consistency.

5.4 Simple statistical filtering

As our first experiment we would like to empirically test how some statistical analysis can reduce the dimensionality.

5.4.1 The N most common terms

A few articles suggest a selection based on the N most common words appearing in the whole corpus. This cuts down the dimensionality of the vocabulary vector to length N and the document matrix of m documents to $N \times m$.

N	Purity	NMI	NNZ (%)	dim (%)
1000	0.163 ± 0.004	0.054 ± 0.002	56.49	0.56
3000	0.166 ± 0.005	0.054 ± 0.002	70.90	1.69
7000	0.168 ± 0.005	0.055 ± 0.001	80.65	3.94
10000	0.169 ± 0.005	0.055 ± 0.001	84.09	5.62
15000	0.169 ± 0.004	0.055 ± 0.001	87.54	8.40
30000	0.169 ± 0.005	0.055 ± 0.002	92.27	16.87
177868	0.170 ± 0.004	0.055 ± 0.001	100.00	100.00

Table 5.6: Keeping only the N most common words of the vocabulary, NG20

N	Purity	NMI	NNZ (%)	dim (%)
1000	0.531 ± 0.006	0.143 ± 0.001	50.15	1.07
3000	0.533 ± 0.003	0.147 ± 0.001	63.91	3.20
7000	0.542 ± 0.010	0.148 ± 0.001	74.19	7.47
10000	0.549 ± 0.011	0.148 ± 0.001	78.25	10.68
15000	0.554 ± 0.010	0.150 ± 0.002	82.59	16.02
30000	0.556 ± 0.010	0.150 ± 0.002	89.10	32.03
93656	0.555 ± 0.008	0.150 ± 0.002	100.00	100.00

Table 5.7: Keeping only the N most common words of the vocabulary, GP

Even though this reduces the dimensionality drastically it does not reduce the number of nonzero values more than half at one hundredth of the dimensionality. As the only textual filter this parameter has very little impact on the clustering results. Unfortunately the clustering results are so bad that no other conclusions can be drawn from them.

5.4.2 Terms less common than u

Instead of removing all but the N most common one can limit the terms to only those which exist in at most u texts of the input. This reduces the connectedness between clusters that share words that are not discriminatory. We have seen values of 60–90%, in literature Another idea we tried was to set this limit to $1/k$ and even smaller to remove all smudging factors.

u (%)	Purity	NMI	NNZ (%)	dim (%)
0.1	0.198 ± 0.012	0.110 ± 0.008	13.27	92.37
0.5	0.605 ± 0.018	0.525 ± 0.011	25.88	97.81
1	$\mathbf{0.663 \pm 0.016}$	$\mathbf{0.574 \pm 0.009}$	33.58	98.81
5	0.595 ± 0.013	0.553 ± 0.006	55.14	99.77
10	0.535 ± 0.010	0.479 ± 0.005	64.14	99.89
30	0.301 ± 0.014	0.178 ± 0.004	77.48	99.96
50	0.210 ± 0.005	0.122 ± 0.001	84.38	99.98
70	0.189 ± 0.005	0.093 ± 0.003	90.36	99.99
90	0.210 ± 0.008	0.088 ± 0.003	95.77	100.00
100	0.1700 ± 0.004	0.055 ± 0.001	100.00	100.00

Table 5.8: Removing terms occurring in more than fraction u documents, NG20

u (%)	Purity	NMI	NNZ (%)	dim (%)
0.1	0.362 ± 0.023	0.114 ± 0.015	15.39	80.46
0.5	0.611 ± 0.028	0.397 ± 0.019	29.54	94.54
1	0.674 ± 0.026	0.454 ± 0.017	37.59	97.16
5	$\mathbf{0.784 \pm 0.016}$	$\mathbf{0.505 \pm 0.015}$	57.65	99.43
10	0.766 ± 0.018	0.487 ± 0.012	66.36	99.71
30	0.610 ± 0.016	0.283 ± 0.007	80.44	99.91
50	0.512 ± 0.001	0.203 ± 0.001	88.72	99.96
70	0.476 ± 0.001	0.179 ± 0.001	92.22	99.98
90	0.536 ± 0.027	0.188 ± 0.008	98.23	100.00
100	0.555 ± 0.008	0.150 ± 0.002	100.00	100.00

Table 5.9: Removing terms occurring in more than fraction u documents, GP

Some very interesting results spring forth from this simple filter. It seems that by heavily removing the most common terms one can get reasonably good clustering results on these specific corpora. Note that while a lot of the matrix elements are removed the dimensions stay roughly the same. In other words, this operation produces a yet more sparse environment possibly making clusters more separated.

5.4.3 Terms more common than L

The last statistical feature we filter by a lower bound, words must exist in at least L documents to not get filtered out. If a word only exists in one or two documents it does not help to generalize those specific documents into any group and could therefore be considered noise in a sense.

L	Purity	NMI	NNZ (%)	dim (%)
0	0.167 ± 0.004	0.055 ± 0.001	100.00	100.00
3	0.169 ± 0.005	0.055 ± 0.001	95.13	29.85
5	0.168 ± 0.006	0.055 ± 0.002	93.17	19.86
7	0.169 ± 0.004	0.055 ± 0.001	91.82	15.58
15	0.170 ± 0.005	0.055 ± 0.001	88.14	9.10
30	0.168 ± 0.004	0.055 ± 0.001	83.70	5.39

Table 5.10: Removing terms occurring in less than L documents, NG20

L	Purity	NMI	NNZ (%)	dim (%)
0	0.555 ± 0.008	0.150 ± 0.002	100.00	100.00
3	0.557 ± 0.005	0.150 ± 0.002	87.35	25.91
5	0.546 ± 0.012	0.148 ± 0.002	82.53	15.91
7	0.549 ± 0.011	0.148 ± 0.001	79.21	11.64
15	0.540 ± 0.010	0.148 ± 0.001	71.14	5.78
30	0.533 ± 0.003	0.146 ± 0.001	62.81	2.93

Table 5.11: Removing terms occurring in less than L documents, GP

By only applying this filtering we see no real improvement in the resulting clusters. There is however a big reduction in dimensionality already with a requirement of occurrence frequency being more than 5. In literature we have seen frequency values of 3 to

30. Further investigation is required to see what the consequences of clustering quality is in combination with other processing.

5.5 Term weighting schemes

The assignment of weights to words is commonly used in the vector space model. Our next experiment tries to demonstrate the impact of different weighting schemes on the resulting clusters. The first setup is unprocessed in every way except for word counting and tokenisation. The schemes used are tf, the baseline of this test that uses the word count directly. Manning[27] suggests a values normalized by the maximum term of each vector $ntf = \alpha + (1 - \alpha)tf/tf_{max}$. The two last functions use a sub-linear scheme, $1 + \log_2 tf$ and $\sqrt{tf}$ which smoothes the vector coordinates towards 1, toning down the extreme frequencies. The reasoning here is that a term of frequency ten is not necessarily 10 times more important than a term of frequency 1.

	GP		NG20	
Scheme	Purity	NMI	Purity	NMI
tf	0.555 ± 0.008	0.150 ± 0.002	0.170 ± 0.004	0.055 ± 0.001
ntf	0.555 ± 0.009	0.150 ± 0.002	0.170 ± 0.004	0.055 ± 0.001
$\log_2 tf$	0.574 ± 0.001	0.237 ± 0.001	0.221 ± 0.023	0.125 ± 0.002
$\sqrt{tf}$	$\mathbf{0.631 \pm 0.010}$	$\mathbf{0.304 \pm 0.009}$	$\mathbf{0.238 \pm 0.017}$	$\mathbf{0.138 \pm 0.002}$

Table 5.12: Weighting schemes

As can be seen in tables 5.12 and 5.13 it seems using a smoothing function for the frequency does make sense in practice. Both sub-linear functions perform better than no weighting. Surprisingly the max-tf scheme does not affect the results at all.

	GP		NG20	
Scheme	Purity	NMI	Purity	NMI
$tf \times idf$	0.605 ± 0.006	0.205 ± 0.006	0.206 ± 0.005	0.102 ± 0.002
$ntf \times idf$	0.605 ± 0.004	0.204 ± 0.002	0.204 ± 0.005	0.102 ± 0.002
$\log_2 tf \times idf$	0.699 ± 0.018	0.329 ± 0.017	0.403 ± 0.015	0.254 ± 0.005
$\sqrt{tf} \times idf$	$\mathbf{0.710 \pm 0.020}$	$\mathbf{0.368 \pm 0.013}$	$\mathbf{0.450 \pm 0.022}$	$\mathbf{0.347 \pm 0.014}$

Table 5.13: Weighting schemes with *idf*

Combining these weighting schemes with the inverse document frequency described in section 2.1 we get an increase of cluster quality across the board. It does seem to impact the larger corpus NG20 a lot more than GP. These scores are rather low however which means we need more textual processing than just counting unique terms.

5.5.1 Standard settings

In our second experiment we see the real world importance of weighting with idf.

	GP		NG20	
Scheme	Purity	NMI	Purity	NMI
tf	0.597 ± 0.006	0.271 ± 0.005	0.279 ± 0.011	0.155 ± 0.004
ntf	0.598 ± 0.008	0.271 ± 0.007	0.281 ± 0.009	0.155 ± 0.004
$\log_2 tf$	0.699 ± 0.005	0.319 ± 0.005	0.228 ± 0.011	0.152 ± 0.004
$\sqrt{tf}$	$\mathbf{0.667 \pm 0.003}$	$\mathbf{0.333 \pm 0.002}$	$\mathbf{0.273 \pm 0.017}$	$\mathbf{0.169 \pm 0.005}$

Table 5.14: Weighting schemes with textual processing

	GP		NG20	
Scheme	Purity	NMI	Purity	NMI
$tf \times idf$	0.795 ± 0.009	0.514 ± 0.012	0.708 ± 0.017	0.617 ± 0.007
$ntf \times idf$	$\mathbf{0.796 \pm 0.010}$	$\mathbf{0.515 \pm 0.011}$	0.707 ± 0.018	0.617 ± 0.007
$\log_2 tf \times idf$	0.792 ± 0.011	0.505 ± 0.015	0.729 ± 0.013	0.627 ± 0.007
$\sqrt{tf} \times idf$	0.773 ± 0.023	0.492 ± 0.011	$\mathbf{0.742 \pm 0.011}$	$\mathbf{0.639 \pm 0.004}$

Table 5.15: Weighting schemes with *idf* and textual processing

Yet again we see a much bigger impact on the NG20 corpus when using the inverse document frequency. Compared to the results where no processing occurred, these result scores are actually quite acceptable and comparable to studies in literature. Interesting to note here is that the sub-linear weighting methods $\log_2 tf$ and $\sqrt{tf}$ actually make the clusterings worse on our GP corpus while the opposite effect is seen on NG20.

5.6 Language based filtering (LBF)

In this experiment we wanted to investigate the impact of language based filtering or language based feature selection.

It is standard procedure in most information retrieval application to prune stop-words from the document in the corpus. Here we use the stop-word list provided by the *Natural Language Tool Kit*[3]. In extension to stop-word filtering, we apply Part of speech-tag (POS-tag) selection and syntactic role selection (SR-selection).

Word classes, or *part-of-speech* tags are obtained by running the corpus through the HunPos[18][4] tagger. HunPos is a hidden Markov model based part of speech tagger. The tagger uses a trained model for each language. The English model is trained on Penn Tree Bank II*[18]*; an annotated corpus consisting of text from the Wall Street Journal. For Swedish the model is trained on the Swedish equivalent, SUC-corpus and uses their annotation.

Due to the probabilistic nature of the underlying algorithms in HunPos, it does not have a 100% accuracy. [29] reports an accuracy of 95.90% on a 1M Swedish token data set, while [18] reports an overall accuracy for English of 96.58% on the 1M token WSJ data set.

The syntactic roles tags are extracted with the use of MaltParser[5][33] dependency parser. It is a data driven parser generator that makes use a support vector machine *(liblinear)* to make classification. It too uses trained model for each language. The two language specific trainings sets are the same for MaltParser as for HunPos. Nivre[32] reports a tagging accuracy of 96.1% and 95.6% for English and Swedish respectively.

[3]`http://nltk.googlecode.com/svn/trunk/nltk_data/packages/corpora/stopwords.zip`
[4]`http://code.google.com/p/hunpos/`
[5]`http://www.maltparser.org`

The input format for MaltParser requires that the text data is tagged with POS-tags. We feed the MaltParser the POS-tagged data produced by HunPos.

We apply feature selection when filtering, *i.e.,* extracting those words which are of interest, rather than removing unwanted words (except for stop words). A lists of allowed part of speech tags and syntactic role tags are located in appendix B.1 and B.2 respectively.

In the tables 5.16 through 5.19 let

s : stop-word filtering

p : POS-tag selection

r : SR-tag selection

5.6.1 LBF on NG20

Filter	Purity	NMI	NNZ (%)	dim (%)
$base$	0.170 ± 0.004	0.055 ± 0.001	100.00	100.00
s	0.121 ± 0.003	0.026 ± 0.001	77.83	99.93
p	0.157 ± 0.007	0.083 ± 0.002	59.47	83.19
r	0.152 ± 0.010	0.051 ± 0.001	15.30	16.18
$s + p$	$\mathbf{0.309 \pm 0.013}$	$\mathbf{0.152 \pm 0.002}$	55.24	83.13
$s + r$	0.186 ± 0.013	0.102 ± 0.002	11.03	16.16
$s + t + r$	0.178 ± 0.009	0.102 ± 0.003	**10.63**	**15.34**

Table 5.16: Language Based Filtering NG20

Filter	Purity	NMI	NNZ (%)	dim (%)
$base$	0.699 ± 0.010	0.600 ± 0.006	61.07	3.94
s	0.700 ± 0.011	0.603 ± 0.006	52.59	3.94
p	0.742 ± 0.011	0.638 ± 0.005	40.83	3.94
r	0.396 ± 0.017	0.325 ± 0.005	9.49	2.20
$s + p$	$\mathbf{0.744 \pm 0.012}$	$\mathbf{0.640 \pm 0.004}$	39.06	3.94
$s + r$	0.402 ± 0.018	0.325 ± 0.005	9.13	2.18
$s + t + r$	0.396 ± 0.013	0.322 ± 0.005	**8.82**	**2.13**

Table 5.17: Language Based Filtering NG20 with textual processing

In table 5.16 we see that our stop-word filtering in the unprocessed stage truly reduces the number of non-zero entries and there by increase separation between documents. However it worsens the cluster purity and NMI compared to the base line. It has been speculated that perhaps stop word filtering is better suited for information retreival tasks than clustering[38]. Here we get some hint that it might be the case.

While it might be a good idea to filter on stop-words for short queries, it has little effect on total size of term-document matrix. Our experiments show that stop-word filtering can works well as a booster for other techniques.

Both POS-tag and SR selection do present a significant dimensionality and non-zero element reduction, with the SR selector being the most strict filtering policy. Just allowing nouns, proper nouns and verbs, we achieve a boost in clustering quality, while reducing the dimensionality. From the above experiment we can conclude that the best quality clusters are generated when we combine stop-word filtering and POS-tagging.

Just from a small change in the vocabulary (106 terms) allows us to double the clustering quality.

5.6.2 LBF on GP

Filter	Purity	NMI	NNZ (%)	dim (%)
$base$	0.555 ± 0.008	0.150 ± 0.002	100.00	100.00
s	0.585 ± 0.013	0.142 ± 0.002	82.90	99.88
p	0.559 ± 0.009	0.166 ± 0.003	56.43	82.82
r	0.449 ± 0.009	0.114 ± 0.001	36.40	55.32
$s+p$	0.604 ± 0.006	0.272 ± 0.006	53.91	82.77
$s+r$	$\mathbf{0.620 \pm 0.018}$	$\mathbf{0.325 \pm 0.012}$	32.17	55.28
$s+t+r$	0.613 ± 0.020	0.302 ± 0.008	**28.98**	**50.95**

Table 5.18: Language Based Filtering GP

Filter	Purity	NMI	NNZ (%)	dim (%)
$base$	0.787 ± 0.003	0.468 ± 0.002	65.50	7.47
s	0.757 ± 0.004	0.485 ± 0.004	55.74	7.47
p	$\mathbf{0.803 \pm 0.010}$	0.498 ± 0.012	43.64	7.47
r	0.774 ± 0.015	0.496 ± 0.011	23.31	5.57
$s+p$	0.775 ± 0.022	0.494 ± 0.013	41.71	7.47
$s+r$	0.768 ± 0.012	$\mathbf{0.512 \pm 0.013}$	20.71	5.55
$s+t+r$	0.772 ± 0.016	0.509 ± 0.010	**19.65**	**5.25**

Table 5.19: Language Based Filtering GP with textual processing

The results for GP is similar to NG20. Except for the combined setting, where $s + r$ has a small advantage over $s + p$.

5.6.3 LBA over all

The over all results indicate that the single best policy to boost the clustering quality is POS-tag selection for both languages. When combining with stop-word filtering, the results disagree. Assuming that the parser performs equally good on both data sets (previous results indicate similar accuracy) and that we apply equivalent selection preferences, the difference is how much the policies reduces the data set. In NG20 only 16.16% of the vocabulary remain when using $s + r$, compared to 55.28% for the same policy in GP. Besides the obvious difference in languages, the two corpora are produced in different ways. GP contains professionally written articles, while NG20 is peoples discussions about various topics. This might play a part in why there is such a significant difference in quality.

5.7 Lemmatisation, synonyms and compounding

The idea here is to insert some sense of language semantics in the textual processing. The first problem we consider is where we get multiple terms from a single underlying word inflected in different ways. A typical example is "walk", "walks", "walked", "walking" and so on. These words convey similar semantic information to the text but are considered different terms. By solving this problem one loses some syntactic nuances but it gives us an arguably more compact representation of the semantics of a processed text. What we would like to do is *lemmatise* the group of words into a single item. This concept is similar to *stemming* where one does suffix removal based on a set of rules. The de facto stemmer out there today for English is the Porter stemmer (see [36]) that works quite well on Germanic languages. Some problems arise however when using a stemmer, sometimes different words will assume the same root and a stemmer will fail at finding the lemma of the word "better" because it requires a dictionary look-up. Our implementation uses lexical look-ups, in WordNet[30] for English and in SALDO[11] for Swedish, with some heuristics to remove inflections.

Different words that carry the same or very similar meaning are called synonyms. These might pose a possible problem when trying to find a grouping. In extreme cases two documents might share the exact same high level idea but do not share a single information carrying word. One possible remedy for this problem is to make a look-up for each word and try to unify words carrying the same meaning. Euphemism poses a different but related problem where an expression such as "kicked the bucket" (when referring to a person) usually doesn't have anything to do with neither kicks or buckets. In this case the correct substitution would be "died" or possibly the root word "death".

Compounded words carry information regarding two or more different semantic concepts merged into a single term. The norm in Germanic languages is to prepend the main word with a descriptive word thereby forming the concatenation. These are usually called *solid compounds* whereas some languages separates each component with a space which are called *open compound* (*e.g.,* English). The impact of splitting solid compounds in Swedish is investigated in [37]. We would rather like to investigate the opposite problem of joining open compounds. More specifically names of places, persons and entities. For this we use a snapshot of the Wikipedia index from DBpedia[8]. This decreases the descriptive power of the component words but also avoids mistakes like different persons sharing family name with no blood relations, or company names with meanings totally unrelated to their business etc.

In the tables 5.20 through 5.21 let

l : lemmatisation

d : dbpedia

s : synsets

Filter	Purity	NMI	NNZ (%)
base	0.744 ± 0.012	0.640 ± 0.004	39.06
l	0.738 ± 0.009	0.644 ± 0.004	39.16
d	0.745 ± 0.011	0.638 ± 0.004	38.14
s	0.742 ± 0.011	0.641 ± 0.005	39.39
$l + d$	0.736 ± 0.008	0.643 ± 0.004	38.22
$l + s$	0.739 ± 0.009	0.641 ± 0.003	39.42
$d + s$	0.746 ± 0.011	0.641 ± 0.005	38.46
$l + d + s$	0.740 ± 0.009	0.643 ± 0.004	38.47

Table 5.20: Lexical analysis NG20

All these experiments were run with full textual processing mentioned in section 5.3. What we can clearly see here is that neither filter yields any significant improvements but rather minor changes in quality within the standard deviation.

Filter	Purity	NMI	NNZ (%)
base	0.775 ± 0.022	0.494 ± 0.013	41.71
l	0.795 ± 0.009	$\mathbf{0.547 \pm 0.013}$	43.43
d	0.784 ± 0.015	0.509 ± 0.018	41.23
s	0.767 ± 0.021	0.484 ± 0.018	41.97
$l + d$	0.797 ± 0.009	$\mathbf{0.547 \pm 0.013}$	42.92
$l + s$	0.784 ± 0.009	0.530 ± 0.011	39.68
$d + s$	$\mathbf{0.802 \pm 0.024}$	0.513 ± 0.017	41.44
$l + d + s$	0.782 ± 0.007	0.524 ± 0.009	39.00

Table 5.21: Lexical analysis GP

In our Swedish corpus the changes are a lot more vivid. Lemmatisation shows real promise while dbpedia look-ups not so much. Please note that the dbpedia look-up was in the English language version for more completeness because the Swedish version does not have more than roughly one tenth of the number of articles. This should hopefully not impact our main targets too much: names, places and corporations. The synonym replacement method we performed was worse than base which suggests that our specific method of replacement could use some improvements.

5.8 Keyword extraction

In modern search application it is common to generate keywords from the indexed data and display these when presenting the query results. The keywords are a set terms or noun phrases that represent a document. The idea behind this experiment is that we shall preprocess our data and extract keywords from our documents and allow for these keywords to represent the documents. From our previous exploration of the data sets we have found that after our text process reduction schema have been applied, we had 69 and 89 words on average remaining for NG20 and GP respectively. This brings up and interesting question: If we are able to extract an equal amount of keywords, how would these keywords perform be compared to our other textual processing for clustering? Would one be able to just cluster on a M number of keywords per document?

To extract keywords we the software developed by Johansson and Linström[24]. They report precision of 0.31 and recall of 0.51, with a total F-measure of 0.39 .

From this we have taken two different approaches on how to evaluated the extracted keywords. The first aspect is that the keywords are well represented for the documents and all play an equally important role in the representation for the document. We therefor apply a boolean weighting schema for the keywords.

Filter	Purity	NMI	NNZ (%)	dim (%)
base	$\mathbf{0.744 \pm 0.012}$	$\mathbf{0.640 \pm 0.004}$	39.06	**3.94**
69	0.168 ± 0.013	0.072 ± 0.003	22.04	171.03
50	0.165 ± 0.012	0.074 ± 0.003	19.72	171.03
30	0.205 ± 0.014	0.109 ± 0.008	6.05	171.03
15	0.222 ± 0.013	0.111 ± 0.007	8.80	171.03
10	0.205 ± 0.014	0.109 ± 0.008	6.05	171.03
5	0.211 ± 0.014	0.106 ± 0.006	**3.07**	171.03

Table 5.22: Keyword extraction NG20 with equal weight

Filter	Purity	NMI	NNZ (%)	dim (%)
base	$\mathbf{0.775 \pm 0.022}$	$\mathbf{0.494 \pm 0.013}$	41.71	**7.47**
89	0.478 ± 0.016	0.175 ± 0.007	26.76	104.07
50	0.483 ± 0.020	0.166 ± 0.006	18.68	104.07
30	0.471 ± 0.022	0.183 ± 0.010	12.55	104.07
15	0.478 ± 0.031	0.197 ± 0.014	6.75	104.07
10	0.437 ± 0.024	0.171 ± 0.016	4.57	104.07
5	0.398 ± 0.023	0.138 ± 0.015	**2.32**	104.07

Table 5.23: Keyword extraction GP with equal weight

Apparently this point of view on keywords are inferior as documents representation in clustering. Evaluating keywords extracted from a random documents, we see that some keywords occur with a much higher frequency than expected. In NG20 the word “subject” occur in almost all documents. A more thorough investigation would probably reveal that a small set of keywords are dominating the corpus and and this would cause a skewed feature space, where document separation is more complicated.

A second possible reason why the keywords perform worse than expected can be related to the trained models that the keyword extractor uses. Those are trained on mostly medical literature. The increased dimensionality is a consequence from that keywords can be noun-phrases, hence the number of unique features increases. We can see that when the number of keywords are 15 we seem to reach some maxima on clustering quality for keyword clustering.

Alternatively we can assume that keywords are the result of a prior dimensionality reduction and that the keywords generated are bag-of-word representation for the documents. Given the previous results of the automatically generated keywords, bag-of-words aspect is motivated by the fact the we do not have perfect keywords. If we use a more appropriate weighting schema, we should be able to reduce weight on those terms with high idf score and there by increase the clustering quality.

Filter	Purity	NMI	NNZ (%)	dim (%)
base	$\mathbf{0.744 \pm 0.012}$	$\mathbf{0.640 \pm 0.004}$	39.06	3.94
69	0.721 ± 0.009	0.629 ± 0.004	22.47	3.94
50	0.722 ± 0.009	0.626 ± 0.004	20.74	3.94
30	0.716 ± 0.011	0.618 ± 0.004	17.09	3.94
15	0.661 ± 0.014	0.567 ± 0.007	12.13	3.94
10	0.586 ± 0.018	0.493 ± 0.009	9.07	3.94
5	0.330 ± 0.015	0.263 ± 0.007	**4.95**	**2.81**

Table 5.24: Keyword extraction NG20 as dim reduction

Filter	Purity	NMI	NNZ (%)	dim (%)
base	0.775 ± 0.022	0.494 ± 0.013	41.71	7.47
89	0.795 ± 0.018	$\mathbf{0.528 \pm 0.008}$	19.92	5.62
50	$\mathbf{0.800 \pm 0.019}$	0.519 ± 0.010	12.62	4.02
30	0.763 ± 0.011	0.478 ± 0.007	7.78	2.71
15	0.633 ± 0.021	0.355 ± 0.015	3.52	1.38
10	0.582 ± 0.032	0.295 ± 0.015	2.06	0.85
5	0.438 ± 0.023	0.177 ± 0.015	**0.81**	**0.35**

Table 5.25: Keyword extraction GP as dim reduction

Interestingly this seems to prove itself quite useful. We achieve a good dimensionality reduction and yet high quality clustering is achieved. Applying a more 'fair' weighting schema reduces the score of the most frequent occurring keywords and increase the count on the unique terms. The keywords beats the base line on the GP corpus but is unable to beat the NG20 base line. A probable reason for this could that the underlying functionality behind the keyword extractor performs with different quality on the different languages[24]. Even though the keyword extracted data performs roughly equally good as our other text processing settings the running time overhead in producing the keywords is far greater than our previous feature selection methods.

5.9 Comparing clustering methods

In a final experiment we evaluate the reviewed clustering algorithms with respect to document clustering. Here our objective is to demonstrate how different classes of clustering algorithms performs using our textual processing. To achieve a broad spectrum of documents we have used external clustering tools such as CLUTO[6] as well as own implementations of well described algorithms such as OSKM and GNG. CLUTO is an tool developed for efficient clustering of high dimensionality data clustering. The package contains various clustering algorithms, of which we have chosen to use the following:

RB is an repeating bisecting k–means. Here default settings are used, with using cosine similarity and random initialisation seed

graph performs a k–way spectral clustering. We use asymmetric graph linkage and the Jaccard similarity metric

bagglo is an bias-agglomerative method. It first compute an heuristic by using the repeating bisection method and then applies this to an agglomerative algorithm. By default, $UPGMA$ is the criterion function

agglo is a standard agglomerative method, however apply different criterion functions. *slink* which is single link. *clink* is complete linkage and *upgma* as .

All CLUTO methods run with 'colmodel=none' since we when already perform our own term weighing.

We have written our own implementation of the *online spherical k means.*

OSKM We follow the implementation details in the original paper. The data sample procedure implemented as well as the exponentially decaying learning rate. The termination criterion is different, instead of ending after M batches we continue until no document changes cluster between batches. To make sure that we sample all documents, the algorithm will run for at least M iterations. The parameters used are: M =10, $\eta_0 = 1.0$ and $\eta_f = 0.01$

Topic modeling is also represented in the survey by an LDA implementation that uses Gibb's sampling. It is called GibbsLDA++[7].

LDA We cluster simply by using θ_d and assigns the documents to its most dominant topic, *i.e.,* creating a hard clustering. We assume that each category within the corpus is topic. The number of topics is equal to k.

To represent the self-organizing clustering methods we have implemented our own growing neural gas

[6]`http://glaros.dtc.umn.edu/gkhome/views/cluto`
[7]`http://gibbslda.sourceforge.net/`

GNG We ran the GNG with settings recommended by [16]

Each algorithm ran 20 times

	GP		NG	
Algorithm	Purity	NMI	Purity	NMI
RB	0.771 ± 0.024	0.494 ± 0.015	0.746 ± 0.012	0.640 ± 0.004
graph	0.733 ± 0.024	0.522 ± 0.014	0.629 ± 0.018	0.518 ± 0.008
bagglo	0.791 ± 0.000	0.480 ± 0.000	0.646 ± 0.000	0.635 ± 0.000
agglo_{slink}	$\mathbf{0.932 \pm 0.000}$	0.027 ± 0.000	$\mathbf{0.953 \pm 0.000}$	0.018 ± 0.000
agglo_{clink}	0.408 ± 0.000	0.138 ± 0.000	0.181 ± 0.000	0.044 ± 0.000
agglo_{upgma}	0.672 ± 0.000	0.279 ± 0.000	0.310 ± 0.000	0.354 ± 0.000
OSKM	0.808 ± 0.021	$\mathbf{0.525 \pm 0.011}$	0.730 ± 0.029	$\mathbf{0.680 \pm 0.014}$
LDA	0.712 ± 0.019	0.499 ± 0.011	0.581 ± 0.027	0.588 ± 0.015
GNG	0.821 ± 0.010	0.504 ± 0.010	0.575 ± 0.016	0.468 ± 0.010

Table 5.26: Clustering methods results

The single link criterion produces pure clusters but has a poor NMI. The high purity can be explained by how we measure purity. Our purity is the unweighted mean over all clusters and since single link produces one huge “blob” as a main cluster and leaving the rest of the clusters fairly sparse. The small clusters will have a good purity given their small size, hence the unweighted mean is high. See appendix C for more details.

Over all OSKM produces the best clustering given our data sets and preprocessing. When comparing our results for OSKM with the result given in the original paper for the same data set we have achieved about 10% better clustering result thanks to our textual processing.

Chapter 6

Conclusion

The project set out to perform a survey over document clustering techniques. This has involved a broad investigation from the underlying data models to various algorithms. Our main focus has been investigating different text processing methods in order to enhance the clustering results. Motivation has been on the one hand to reduce dimensionality in order to keep running times low and on the other to enhance clustering results. We have focused on feature selection methods, rather than feature extraction. Feature extraction is very well formulated mathematically but their heavy computational burden makes them almost infeasible in any real application. However with the benefit of working with texts is we can apply more sophisticated feature selection methods.

We investigated the impact of how well simple statistical filtering affects the clustering results. Results from our experiments showed that by filtering heavily on words that occur frequently over the entire corpus, we achieve acceptable clustering results.

Term weighting makes a huge impact on the clustering quality. Though well explored in information retreival, not as well explored in the document clustering literature, where the standard $tf \times idf$ dominates. We demonstrated the impact of inverse document frequency and that in a stand alone case, when no other textual process is used, the $tf \times idf$ is inferior to $\sqrt{tf} \times idf$. This result is unfortunately not consistent when combined with our other processing.

Stop-word filtering is almost always applied in all clustering applications. We show this can be extended to more delicate levels by filtering on certain part of speech tags and/or syntactic role tags. The stop-words themselves gave very poor results, but combined with the more complex feature selection methods, it acted as a boosting technique. When POS-tag filtering was combined with the stop-word filtering and the other textual processing techniques, we achieved our highest clustering quality.

The effect of lemmatisation was tested with some inconclusive results. We tried to "normalise" words further by replacing them with their synonyms, but with little success. The disambiguation issue makes the selection problem hard and it reflects the results.

We regarded keywords both as perfect document representations as well as reduced document representation for document clustering. When regarding the generated keywords as perfect document representatives we were fairly unsuccessful in clustering but the quality of keywords could questioned.

Last but not least we apply a cocktail of textual processing techniques and evaluate a different clustering algorithms set to represent different algorithm classes. The overall result are quite even with the Online Spherical k–means turning out victorious. However simply judging the clustering result by a metric is treacherous and can be deceiving as shown by some of the confusion matrices presented in appendix C. Categories can have similar attributes but be in different classes in the reference set. However this critique is in favour of the clustering algorithms as some categories are similar.

Over all document clustering presents a feasible way for making a rough categorisation of large corpora, infeasible to humans.

Bibliography

[1] Dimitris Achlioptas. Database-friendly random projections. In *PODS '01: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 274–281, New York, NY, USA, 2001. ACM.

[2] Nir Ailon and Bernard Chazelle. Faster dimension reduction. *Commun. ACM*, 53(2):97–104, 2010.

[3] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. Np-hardness of euclidean sum-of-squares clustering. *Mach. Learn.*, 75(2):245–248, 2009.

[4] Nicholas O. Andrews and Edward A. Fox. Recent developments in document clustering. Technical report, Computer Science, Virginia Tech, 2007.

[5] J. Becker and D. Kuropka. Topic-based vector space model. In *Proceedings of the 6th International Conference on Business Information Systems*, pages 7–12, Colorado Springs, July 2003.

[6] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250, New York, NY, USA, 2001. ACM.

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[8] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia : a crystallization point for the web of data. 2009.

[9] David M. Blei and Lafferty J. Topic models. In *Text Mining: Theory and Applications*. 2009.

[10] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[11] Lars Borin, Markus Forsberg, and Lennart Lönngren. Saldo 1.0 (svenskt associationslexicon version 2). Technical report, Språkbanken, Göteborg universitet, 2008.

[12] David Chalmers. *The conscious mind: in search of a fundamental theory*. Oxford University Press, Inc., New York, NY, USA, 1996.

[13] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–329, New York, NY, USA, 1992. ACM.

[14] Martin Ester, Hans-Peter Kriegel, Jörg S, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[15] Imola Fodor. A survey of dimension reduction techniques. Technical report, University of California, 2002.

[16] Bernd Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, 1995.

[17] Alan Genz. Methods for generating random orthogonal matrices. In *Monte Carlo and Quasi-Monte Carlo Methods 1998*, pages 199–213, Berlin, 1999. Springer-Verlag.

[18] Péter Halácsy, András Kornai, and Csaba Oravecz. Hunpos: an open source trigram tagger. In *ACL '07: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 209–212, Morristown, NJ, USA, 2007. Association for Computational Linguistics.

[19] Khaled M. Hammouda and Mohamed S. Kamel. Efficient phrase-based document indexing for web document clustering. *IEEE Transactions on Knowledge and Data Engineering*, 16:1279–1296, 2004.

[20] R. Hecht-Nielsen. Context vectors: General purpose approximate meaning representations self-organized from raw data. *Computational Intelligence: Imitating Life*, pages 43–56, 1994.

[21] Gregor Heinrich. Parameter estimation for text analysis. Technical report, University of Leipzig, 2008.

[22] Thomas Hofmann. Probabilistic latent semantic indexing. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, New York, NY, USA, 1999. ACM.

[23] Douglas R. Hofstadter. *Gödel, Escher, Bach: An Eternal Golden Braid.* Basic Books, Inc., New York, NY, USA, 1979.

[24] Martin Johansson and Lindstöm. Keyword extraction using machine learning, 2010.

[25] Man Lan, Chew-Lim Tan, Hwee-Boon Low, and Sam-Yuan Sung. A comprehensive comparative study on term weighting schemes for text categorization with support vector machines. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1032–1033, New York, NY, USA, 2005. ACM.

[26] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. In *WALCOM '09: Proceedings of the 3rd International Workshop on Algorithms and Computation*, pages 274–285, Berlin, Heidelberg, 2009. Springer-Verlag.

[27] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, 1 edition, July 2008.

[28] T. Martinetz and K. Schulten. A "neural-gas" network learns topologies. *Artificial Neural Networks*, I:397–402, 1991.

[29] Beata Megyesi. The open source tagger hunpos for swedish. In *NODALIDA 2009:Proceedings of the 17th Nordic Conference of Computational Linguistics*, volume 4, pages 239–241, Odense,Denmark, 2009. Northern European Association for Language Technology (NEALT).

[30] George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38:39–41, 1995.

[31] Lefteris Moussiades and Athena Vakali. Pdetect: A clustering approach for detecting plagiarism in source code datasets. *Comput. J.*, 48(6):651–661, 2005.

[32] Joakim Nivre, Johan Hall, and Jens Nilsson. Maltparser: A data-driven parser-generator for dependency parsing. In *In Proc. of LREC-2006*, pages 2216–2219, 2006.

[33] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135, 2007.

[34] Karl Pearson. On lines and planes of closest fit to system of points in space. *Philosophical Magazine*, 2:559–572, 1901.

[35] Artem Polyvyanyy and Dominik Kuropka. *A quantitative evaluation of the enhanced topic-based vector space model.* Universitätsverlag Potsdam, 2007.

[36] M. F. Porter. An algorithm for suffix stripping. pages 313–316, 1997.

[37] M. Rosell. Improving clustering of Swedish newspaper articles using stemming and compound splitting. In *Proc. 14th Nordic Conf. on Comp. Ling. – NODALIDA '03*, 2003.

[38] Magnus Rosell. Part of speech tagging for text clustering in swedish. In *NODALIDA 2009:Proceedings of the 17th Nordic Conference of Computational Linguistics*, volume 4, pages 150–157, Odense,Denmark, 2009. Northern European Association for Language Technology (NEALT).

[39] Magnus Sahlgren. *The Word-Space Model: using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces.* PhD thesis, Stockholm University, 2006.

[40] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, November 1975.

[41] Adam Schenker, Horst Bunke, Mark Last, and Abraham Kandel. Clustering of web documents using graph representations. In *Applied Graph Theory in Computer Vision and Pattern Recognition*, pages 247–265. 2007.

[42] John R. Searle. Minds, brains, and programs. *Behavioral and Brain Sciences*, 3:417–424, 1980.

[43] Mark Steyvers and Tom Griffiths. *Probabilistic Topic Models.* Lawrence Erlbaum Associates, 2007.

[44] George Tsatsaronis and Vicky Panagiotopoulou. A generalized vector space model for text retrieval based on semantic relatedness. In *EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 70–78, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[45] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4), 2007.

[46] S. K. M. Wong, Wojciech Ziarko, and Patrick C. N. Wong. Generalized vector spaces model in information retrieval. In *SIGIR '85: Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 18–25, New York, NY, USA, 1985. ACM.

[47] Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. pages 46–54, 1998.

[48] Shi Zhong. Efficient online spherical k-means clustering. In *IEEE Int. Joint Conf. Neural Networks (IJCNN 2005)*, pages 3180–3185, Montreal, Canada, 2005.

Appendix A

Derivations

A.1 PCA

Given an arbitrary *N-dimensional* vector x from the data set, represented as a linear combination of orthonormal basis vectors $[u_1 \perp u_2 \perp \cdots \perp u_N]$ as

$$x = \sum_{i=1}^{N} \alpha_i u_i$$

Assume that we want to represent x by only M $(M < N)$ basis vectors. This can be done by replacing $[\alpha_{M+1}, \ldots \alpha_N]^T$ by some preselected constants β

$$\hat{x}(M) = \sum_{i=1}^{M} \alpha_i u_i + \sum_{j=M+1}^{N} \beta_j u_j$$

The representation error, $\Delta x(M) = x - \hat{x}(M)$ is then

$$\Delta x(M) = \sum_{i=1}^{N} \alpha_i u_i - (\sum_{i=1}^{M} \alpha_i u_i + \sum_{j=M+1}^{N} \beta_j u_j) = \sum_{i=M+1}^{N} (\alpha_i - \beta_i) u_i$$

The error can be measured by the mean square magnitude of Δx

$$\begin{aligned} \bar{\varepsilon}^2(M) &= E[|\Delta x(M)|^2] \\ &= E\left[\sum_{i=M+1}^{N}\sum_{j=M+1}^{N} (\alpha_i - \beta_i)(\alpha_j - \beta_j) u_i^T u_j\right] \qquad \text{(A.1)} \\ &= \sum_{i=M+1}^{N} E[(\alpha_i - b_i)^2] \end{aligned}$$

Now among the basis vectors u_i and constants β_i chose those that minimizes A.1. Optimal value for β_i can be found by derivation the objective function with respect to β_i and setting the partial derivative to 0

$$\frac{\partial}{\partial \beta} E[(\alpha_i - \beta_i)^2] = -2(E[\alpha_i] - \beta_i) = 0 \rightarrow E[\alpha_i] = \beta_i \qquad \text{(A.2)}$$

A.2 shows that we can replace β_i by the expected value of α_i. Inserting this into A.1 one obtains

$$\begin{aligned}
\bar{\varepsilon}^2(M) &= \sum_{i=M+1}^{N} E[(\alpha_i - E[\alpha_i])^2] \\
&= \sum_{i=M+1}^{N} E[(xu_i - E[xu_i])^T(xu_i - E[xu_i])] \\
&= \sum_{i=M+1}^{N} u_i^T E[(x - E[x])(x - E[x])^T]u_i \qquad \text{(A.3)} \\
&= \sum_{i=M+1}^{N} u_i^T \Sigma_x u_i
\end{aligned}$$

Σ_x is here the covariance matrix

Continuing on the effort of minimizing A.3. Since we had the constraint that **u** was to be orthogonal basis ($u_i^T u_i = 1$), we incorporate this with *Lagrange multipliers* λ_i

$$\bar{\varepsilon}^2(M) = \sum_{i=M+1}^{N} u_i^T \Sigma_x u_i + \sum_{i=M+1}^{N} \lambda_i(1 - u_i^T u_i)$$

Compute the partial derivative with respect to **u** and set the function equal to 0. Since the covariance matrix is symmetric the following holds.

$$\frac{\partial}{\partial u_i}\bar{\varepsilon}^2(M) = \frac{\partial}{\partial u_i}\left[\sum_{i=M+1}^{N} u_i^T \Sigma_x u_i + \sum_{i=M+1}^{N} \lambda_i(1 - u_i^T u_i)\right] = 2(\Sigma_x u_i - \lambda_i u_i) = 0 \rightarrow \Sigma_x u_i = \lambda_i u_i$$

This tells us that u_i are eigenvectors of the covariance matrix Σ_x and λ_i are eigenvalues. We may now express the sum square error as

$$\bar{\epsilon}^2(M) = \sum_{i=M+1}^{N} u_i^T \Sigma_x u_i = \sum_{i=M+1}^{N} u_i^T \lambda_i u_i = \sum_{i=M+1}^{N} \lambda_i$$

Hence, in order to minimize the error, λ_i will have to be smallest eigenvalues. Therefore to represent x in M dimensions we shall pick the M eigenvectors u_i that corresponds to the M largest eigenvalues λ_i. The largest eignenvector is also know as *the first principle component.*

Appendix B

Feature selection schemes

B.1 Allowed Part-Of-Speech tags

Listed below are the allowed POS-tags for both English and Swedish. The English tags uses Penn Tree Bank annotation and the Swedish equivalent uses Stockholm-Umeå-corpus (SUC) annotation.

POS-tag	Description
NN	Noun
PM	Proper Noun
VB	Verb

Table B.1: Allowed POS-tags Swedish

Pos-Tag	Description
NN	Noun
NNS	Noun plural
NNP	Proper noun singular
NNPS	Proper noun plural
VB	Verb base form
VBD	Verb past tense
VBG	Verb gerund or present particle
VBN	Verb, past participle
VBP	Verb non 3rd person singular present
VBZ	Verb 3rd person singular present

Table B.2: Allowed POS-tags English

B.2 Allowed syntactic roles

Below are the allowed syntactic role tags for both English and Swedish. For English, we use the Penn Tree Bank tag set and for Swedish the Stockholm-Umeå-corpus (SUC) tag set.

Constituent-labels	Description
AG	Agent
EO	Logical object
ES	Logical subject
IO	Indirect object
OO	Other object
PA	Complement of preposition
SP	Subjective predicative complement
SS	Other subject
HD	Other head

Table B.3: Allowed SR-tags Swedish

Constituent-labels	Description
SBJ	subject
OBJ	object
PRD	predictive complement
OPRD	Predictice complement of rising/control verb
LGS	Logical subject of passive verb
DTV	Dative complement (to) in dative shift
PUT	Complement of the verb put
BNF	Benefactor complement for in dative shift

Table B.4: Allowed SR-tag English

Appendix C

Confusion matrices

Below are example confusion matrices for various clustering algorithms on the GP corpus. Rows represent the categories and the columns are the different clusters. Results from GNG is left out for practical reasons.

	A	B	C	D	E	F	G	H	I	J	K
News	332	1	3	16	3	93	9	109	22	168	2
Economy	137	-	-	3	1	3	-	2	2	5	-
Consumer	121	213	142	-	3	-	-	2	-	33	7
Entertainment	32	4	-	203	-	3	4	-	217	88	-
Food	3	1	36	-	138	-	-	9	-	24	100
Living	26	1	-	-	-	-	-	-	-	67	1
Sports	4	3	-	11	-	14	396	2	-	16	-
Travel	-	-	1	-	1	-	-	-	-	25	-
Job	18	-	-	-	-	-	-	-	1	33	-
Auto	14	20	-	-	-	-	-	-	-	78	-
Fashion	2	3	-	-	-	1	1	-	-	14	-

Table C.1: Confusion matrix: Bisecting k-means

	A	B	C	D	E	F	G	H	I	J	K
News	387	9	3	11	114	12	4	21	38	156	4
Economy	135			1	2	11		1	1	2	
Consumer	101	6	1	207	1	13	175	5	9	1	
Entertainment	32	12	2		90			129	15	272	1
Food	1	107	143	43	10			1	5	1	
Living	24	1	2			11	2	1	56		
Sports	2	24	2		2		2	27	2	10	375
Travel	2		1	1		1			22		
Job	48								3	1	
Auto	2					105	1	1	3		
Fashion	1	1	1			2	1	6	2	5	1

Table C.2: Confusion matrix: Spectral

	A	B	C	D	E	F	G	H	I	J	K
News	427	3	108	17	2	8	4	140	31	17	2
Economy	102		1				47	1		2	
Consumer	506	2			7		1	3			
Entertainment	70		1	106		1	1	5	7	224	138
Food	59	140	9		103						
Living	96				1						
Sports	28		2	30		385	1				
Travel	26	1									
Job	48					1				2	1
Auto	111										1
Fashion	20										

Table C.3: Confusion matrix: Bagglo

	A	B	C	D	E	F	G	H	I	J	K
News	756				1			1		1	
Economy	153										
Consumer	519										
Entertainment	548	1	1				1		1		1
Food	311										
Living	97										
Sports	445			1							
Travel	27										
Job	51										
Auto	112										
Fashion	19					1					

Table C.4: Confusion matrix: Agglo-Slink

	A	B	C	D	E	F	G	H	I	J	K
News	53	186	179	54	113	5	6	12	8	89	63
Economy	12	17	20	22	37	19			1	18	7
Consumer	25	42	55	144	63	2	10		2	36	140
Entertainment	110	39	113	41	79	85	5	20	5	52	4
Food	19	2	77	10	151	21				7	24
Living	2	37	8		10	1	5	1	2	4	27
Sports	40	66	75	19	95		1	16		129	5
Travel	8	11	1	2	4						1
Job	3	23	3	5	15					1	2
Auto		10	2	1	9	3				4	83
Fashion	3	1	6	1				2		7	

Table C.5: Confusion matrix: Agglo-Clink

	A	B	C	D	E	F	G	H	I	J	K
News	3	3	95		2	644	4	1	3	1	3
Economy			1			148	2				2
Consumer					7	508		1			3
Entertainment		1	12	1		533	3		2		1
Food			10		111	12					178
Living			3			94					
Sports		2	2		1	440	1				
Travel						27					
Job						52					
Auto						112					
Fashion						19				1	

Table C.6: Confusion matrix: Agglo-UPGMA

	A	B	C	D	E	F	G	H	I	J	K
News	55	26	11	176	2	51	11	52	206	143	26
Economy	1	5	1	1	1	2		1	134	6	1
Consumer		17	42		126	10	205	4	81	33	
Entertainment	287	5		178	1	14	2	13	12	35	6
Food		1	155		1		1	151		2	
Living		12	5			57	2		19	1	
Sports	3	2		2	1	1	2	11	3	43	378
Travel						23		2		2	
Job						1	1		8	42	
Auto		101				1	4		4	2	
Fashion	8	4				1	3	1		3	

Table C.7: Confusion matrix: LDA

	A	B	C	D	E	F	G	H	I	J	K
News	122	201	109	3	45	3	2	29	3	13	229
Economy	1	107	2	1	2			2			38
Consumer	2	13		344	1		6		2		151
Entertainment	5	19			42	2		236		216	33
Food		2	9	33			105		139		22
Living		2		2			2	1			90
Sports	7	3	2	2	68	352				1	11
Travel								2			24
Job		5						1			46
Auto								1			111
Fashion		3		1	2			4		1	9

Table C.8: Confusion matrix: OSKM