DIMENSIONALITY REDUCTION TECHNIQUES FOR SEARCH RESULTS CLUSTERING

by
STANISŁAW OSIŃSKI

Supervisor
YOSHI GOTOH

This report is submitted in partial fulfilment
of the requirement for the degree of
MSc in Advanced Software Engineering

Department of Computer Science
The University of Sheffield, UK

20 August 2004

Declaration

All sentences or passages quoted in this dissertation from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations which are not the work of the author of this dissertation have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this dissertation and the degree examination as a whole.

Name: Stanisław Osiński

Signature:

Date:

Abstract

Search results clustering is an attempt to automatically organise a linear list of document references returned by a search engine into a set of meaningful thematic categories. Such a clustered view helps the users to identify documents of interest more quickly. One search results clustering method is the *description-comes-first* approach, whereby using a dimensionality reduction technique a number of meaningful group labels are identified, which then determine the content of the actual clusters.

The aim of this project was to compare how three different dimensionality reduction techniques would perform as parts of the description-comes-first method in terms of quality of clustering and computational efficiency. The evaluation stage was based on the standard merge-then-cluster model, in which we used the Open Directory Project web catalogue as a source of human-clustered document references.

During the course of the project we implemented a number of dimensionality reduction techniques in Java and integrated them with our description-comes-first search results clustering algorithm. We also created a simple benchmarking application, which we used to gather data for further comparisons and analysis. Finally, we have chosen one dimensionality reduction technique that performed best both in terms of clustering quality and computational efficiency.

Table of Contents

List of Figures

List of Tables

1 Introduction

With the development of the Internet the well-known classic **Information Retrieval Problem**: *given a set of documents and a query, determine the subset of documents relevant to the query*, gained its modern counterpart in the form of the **Web Search Problem** described by [Selberg, 99]: *find the set of documents on the Web relevant to a given user query*. A broad range of so-called **web search engines** has emerged to deal with the latter task, Google[1] and AllTheWeb[2] being two examples of general-purpose services of this type. Available are also search engines that help the users to locate very specific resources, such as CiteSeer[3], which finds scientific papers and Froogle[4], which is a web shopping search engine. Practical and indispensable as all these services are, their functioning can still be improved.

1.1 Motivation

Low precision searches

The vast majority of publicly available search engines adopt a so-called **query-list paradigm**, whereby in response to a user's query the search engine returns a linear ranking of documents matching that query. The higher on the list, the more relevant to the query the document is supposed to be. While this approach works efficiently for well-defined narrow queries, when the query is too general, the users may have to sift through a large number of irrelevant documents in order to identify the ones they were interested in. This kind of situation is commonly referred to as a **low precision search** [Zamir, 99].

As shown in [Jansen et al., 00], more than 60% of web queries consist of one or two words, which inevitably leads to a large number of low precision searches. Several methods of dealing with the results of such searches have been proposed. One method is pruning of the result list, ranging from simple duplicate removal to advanced Artificial Intelligence algorithms. The most common approach, however, is **relevance feedback**, whereby the search engine assists the user in finding additional key words that would make the query more precise and reduce the number of returned documents. An alternative and increasingly popular method is also search results clustering.

Search results clustering

Search results clustering is a process of organising document references returned by a search engine into a number of meaningful thematic categories. In this setting, in response to a query "Sheffield", for example, the user would be presented with search results divided into such topical groups as "University of Sheffield", "Sheffield United", "Botanical gardens", "BBC Radio Sheffield" etc. Users who look for information on a particular subject will be able to identify the documents of interest much quicker, while those who need a general overview of all related topics will get a concise summary of each of them.

Some important characteristics of the task of search results clustering must be emphasised here. First of all, in contrast to the classical text clustering, search results clustering is based on short document excerpts returned by the search engine called **snippets** [Pedersen et al., 91] rather than the full-text source. Therefore, the algorithms must be prepared to deal with

[1] *http://www.google.com*
[2] *http://www.alltheweb.com*
[3] *http://citeseer.nj.nec.com*
[4] *http://froogle.google.com*

limited length and often low quality of input data. Secondly, as the clustering algorithm is intended to be a part of an on-line search engine, the thematic groups must be created *ad hoc* and fully automatically. Finally, as the main objective of search results clustering is to help the users to identify the documents of interest more quickly, the algorithm must be able to label the clusters in a meaningful, concise and unambiguous way.

The idea of web search results clustering was first introduced in the Scatter/Gather system [Hearst and Pedersen, 96], which was based on a variant of the classic K-Means algorithm. Scatter/Gather was followed by the Suffix Tree Clustering (STC) [Zamir, 99], in which snippets sharing the same sequence of words were grouped together. Finally, the Semantic Hierarchical On-line Clustering (SHOC) algorithm [Zhang, 02] uses the Singular Value Decomposition to identify clusters of documents. With their respective advantages such as speed and scalability, all these algorithms share one important shortcoming: none of them explicitly addresses the problem of cluster description, which often results in the poor quality of cluster labels they generate.

Lingo In our previous work [Osinski and Weiss, 04] we propose a search results clustering algorithm called Lingo in which special emphasis is placed on the quality of group labels. The main idea behind the algorithm is to *reverse* the usual order of the clustering process: Lingo first identifies potential cluster labels using a dimensionality reduction technique called Singular Value Decomposition (SVD), and only then assigns documents to these labels to form proper thematic groups. For this reason we will be also referring to this algorithm as to the **description-comes-first** approach. One conclusion of our work was that although SVD performs fairly successfully in the label discovery phase, alternative dimensionality reduction techniques may prove even more efficient.

There are several reasons for which dimensionality reduction techniques can work well in search results clustering tasks. First of all, they are viewed as a way of extracting *conceptual* components out of multidimensional data. As shown in [Lee and Seung, 99] the Non-negative Matrix Factorisation (NMF) applied to human face images yields components that clearly correspond to such parts of face as eyes, mouth or nose. We believe that in search results clustering these components should reveal different topics present in the input snippets. Secondly, some dimensionality reduction techniques order the discovered components according to their significance. This fact can be potentially exploited in the cluster ranking formula. Finally, as the majority of dimensionality reduction methods can be implemented using iterative improvement algorithms, the quality vs. time efficiency trade-off can be easily balanced in either direction.

1.2 Project goal and scope

The aim of this project is to compare how different dimensionality reduction techniques will perform as parts of the description-comes-first search results clustering algorithm. In particular, three techniques will be evaluated: Singular Value Decomposition (SVD), Non-negative Matrix Factorization (NMF) and Local Non-negative Matrix Factorisation (LNMF). Additionally, a dimensionality reduction algorithm based on the K-Means clustering method, often referred to as Concept Decomposition [Dhillon and Modha, 01], will be used as the baseline for comparisons. A separate version of Lingo should therefore be designed and implemented for each of these techniques.

Evaluation of the above variants of Lingo will be carried out using the merge-then-cluster approach, in which documents related to various topics are mixed together and the algorithm is expected to reconstruct the original groups. In this project the Open Directory Project[5] (ODP) web catalogue will serve as a source of pre-clustered document snippets.

[5] *http://dmoz.org*

1.3 Thesis structure

The structure of this thesis is the following. Chapter 2 deals with background information related to recent approaches to search results clustering and the dimensionality reduction techniques investigated in this project. It also provides a more detailed description of Lingo, along with an illustrative example.

Chapter 3 gives details about the software components developed during the course of this project and the framework within which they operate. Selected implementation issues are also discussed.

In Chapter 4 we describe the problem of evaluation of search results clustering algorithms, and explain in detail the assessment methodology we have decided to use in this project.

Chapter 5 contains results and analyses of the experiments we have carried out. Comparisons involve several variants of Lingo based on different dimensionality reduction methods, but also Suffix Tree Clustering and Tollerance Rough Set Clustering which do not employ dimensionality reduction at all.

Chapter 6 concludes this thesis and gives directions for future work.

1.4 Typographic conventions

A number of typographic conventions are used throughout this thesis to make the reading of the text easier. Words or phrases *in italics* are particularly important for proper understanding of the surrounding context and should be paid special attention to. Italics are also used to place emphasis on a word. First occurrences of terms or definitions will be denoted by **bold face**.

Margin notes and references

Margin notes are used to mark important concepts being discussed in the paragraphs next to them. They are also to help the Reader scan the text. Reference numbers of all figures in the text are composed of the number of the chapter they appear in and the consecutive number of the figure within the chapter. Thus, the fifth figure in the fourth chapter would be referenced as Figure 4.5. Square brackets denote citations and references to other articles, books and Internet resources listed at the back of this thesis.

2 Background

This section provides background information related to our project. We start with explaining the principles of search results clustering, its benefits and the general requirements a successful method solving this problem is expected to meet. Then, we introduce the general structure of a search results clustering algorithm emphasising the text processing techniques employed there. We also describe and analyse a number of search results clustering methods proposed so far, giving special attention to the description-comes-first approach. Finally, we present an overview of the dimensionality reduction techniques investigated in this project.

2.1 Search results clustering

Clustering is a process of forming groups (clusters) of similar objects from a given set of inputs. Good clusters have this characteristic that objects belonging to the same cluster are "similar" to each other, while objects from two different clusters are "dissimilar". The idea of clustering originates from statistics where it was applied to numerical data. However, computer science and data mining in particular, have extended this notion to other types of data such as text or multimedia.

Search results clustering is an attempt to apply the idea of clustering to document references (snippets) returned by a search engine in response to a query. Thus, it can be perceived as a way of organising the snippets into a set of meaningful thematic groups. An example clustered view of Google's response to the query "data mining" is shown in Figure 2.1. There are several ways in which end users can benefit from such a clustered view:

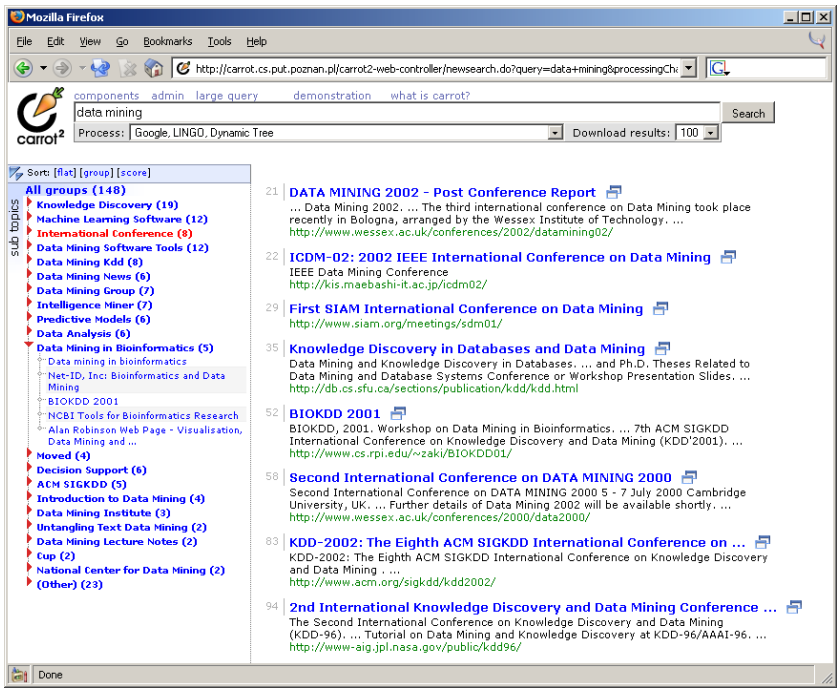
a. **Fast access to relevant documents** — Having documents divided into clearly described categories, users who look for documents on a particular subject can navigate directly to the groups whose labels indicate relevant content. Similarly, most irrelevant documents can be skipped easily, again relying only on cluster description.

b. **Broader view of the search results** — Some users issue general queries to learn about the whole spectrum of available sub-topics. These users will no longer be made to manually scan hundreds of references, and instead, they will be presented with a concise summary of all subjects dealt with in the results.

c. **Relevance feedback functionality** — With the ability to divide the results into sub-categories, search results clustering can also provide the functionality of relevance feedback, enabling the users to refine the initial query based on the labels of clusters generated for that query.

It is important to emphasise here that search results clustering is performed *after* the documents matching the query have been identified by the search engine. Thus, this type of clustering can be regarded as a form of post-processing and presentation of search results.

Figure 2.1
Clustered view of an example web query



Search results clustering involves a fairly new class of algorithms called **post-retrieval document clustering algorithms**. In [Zamir and Etzioni, 98] a list of requirements is given that such algorithms must meet:

Post-retrieval document clustering algorithm requirements [Zamir and Etzioni, 98]

a. **Relevance** — The algorithm ought to produce clusters that group documents relevant to the user's query separately from irrelevant ones.

b. **Browsable Summaries** — The user needs to determine at a glance whether a cluster's contents is of interest. We do not want to replace sifting through ranked lists with sifting through clusters. Therefore the algorithm has to provide concise and accurate descriptions of the clusters.

c. **Overlapping clusters** — Since documents have multiple topics, it is important to avoid confining each document to only one cluster [Hearst, 98].

d. **Snippet tolerance** — The algorithm ought to produce high quality clusters even when it only has access to the snippets returned by the search engines, as most users are unwilling to wait while the system downloads the original documents off the Web.

e. **Speed** — As the algorithm will be used as part of an on-line system, it is crucial that it does not introduce noticeable delay to the query processing. Clustering aims at allowing the user to browse through at least an order of magnitude more documents compared to a ranked list.

f. **Incremental processing** — To save time, the algorithm should start to process each snippet as soon as it is received over the Web.

2.1.1 Stages of search results clustering

Figure 2.2 shows the main stages of the search results clustering process. Below we describe each stage in more detail emphasising particular text processing techniques involved in them.

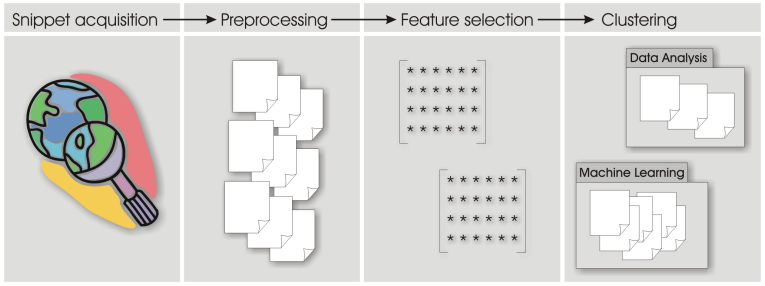


Figure 2.2 *Stages of a search results clustering process*

Snippet acquisition

The aim of this phase is to collect document snippets that have been returned by a search engine in response to the user's query. For web search results clustering the snippets can be obtained directly from a web search engine using an appropriate API, such as Google API[6], or by parsing the engine's HTML output and extracting all necessary data.

Preprocessing

The aim of the preprocessing phase is to transform the raw text of snippets to the form suitable for the clustering algorithm. This phase usually involves the following text processing techniques:

a. **Filtering** — During filtering sequences of characters that could introduce noise and thus affect the quality of clustering are removed from the input snippets. Typically special characters, e.g. '%', '$' or '#', is the content to be filtered out.

b. **Tokenization** — Tokenization is a process of identifying word and sentence boundaries in a text. The simplest tokenizer could use white space characters[7] as word delimiters and selected punctuation marks such as '.', '?' and '!' as sentence delimiters. In practical applications, however, this simplistic approach is usually not robust enough to handle "difficult" content such as document snippets. More elaborate tokenization techniques take into account such additional features of text as hyphens or HTML markup. In [Riboni, 02] evidence is provided that special treatment of e.g. document titles, indicated by the <title> HTML tag, can increase the accuracy of clustering.

c. **Stemming** — During stemming all words in a text are replaced with their respective stems. A **stem** is a portion of a word that is left after removing its affixes (i.e. suffixes and prefixes). Thus, with the use of a **stemmer** (short for a **stemming algorithm**) different grammatical forms of a word can be reduced to one base form. For example, the words: *connected*, *connecting*, *interconnection* should be transformed to the word *connect*.

   For the English language a variety of stemmers are available free of charge, Porter stemmer [Porter, 80] being the most commonly used algorithm. In case of other languages the choice of algorithms is more limited. Snowball[8] is a BSD-licensed stemming library that apart from English supports French, Spanish, Portuguese, Italian, German, Dutch, Swedish, Norwegian, Danish, Russian and Finnish.

d. **Stop word removal** — Stop word removal is a process of identifying and removing **stop words** from a text. Stop words, also referred to as function words, are words that on their own have no identifiable meaning and hence are of little use in some text processing tasks. For English, stop words are among others auxiliary

[6] *http://www.google.com/apis*
[7] Characters that represent space in the text, such as: line feed, carriage return, tabulation marks and regular space characters.
[8] *http://snowball.tartarus.org*

verbs, such as *have*, *be*, pronouns, such as *he*, *it* or prepositions, such as *to* and *for*.

In practical implementations stop word removal is based on a so-called **stop-list**, which is simply a list of stop words for a given language. While English stop-lists can be easily obtained from the Internet, stop words for other languages can be extracted from appropriate corpora.

e. **Language recognition —** Stop word removal, stemming and spell checking make an implicit assumption that the language of the processed text is known. While it may be the case with relatively small collections of documents, in general not only does the language remain unknown, but also it is rarely stated explicitly in the text. Thus, language recognition is an important part of a search results clustering system.

   In [Grefenstette, 95] two statistical methods of automatic language identification are compared. The trigram technique works on the premise that, for example, a word ending in *-ck* is more likely to be an English word than a French word, and similarly a word ending in *-ez* is more likely to be French. The intuition behind the small word technique is that stop words appear in almost all texts and hence are good clues for guessing the language.

Feature selection

The aim of the feature selection phase is to identify words in a text that are non-informative according to corpus statistics and can be omitted during clustering. The reasons for using feature selection in search results clustering are twofold. First of all, in most cases limiting the number of features considerably increases the time efficiency of the clustering algorithm. Secondly, feature selection can help to remove noise from a text, which may result in higher accuracy of clustering.

A large number of feature selection methods have been proposed in the literature, ranging from simple frequency thresholding to complex Information Theoretic algorithms. However, when reviewing the related work, close attention must be paid to the actual text processing problem for which particular selection strategies have been developed. A great majority of them are suitable for the text categorisation problem, and hence make an assumption that the class information and training data are available. Clearly, these assumptions do not hold for the document clustering problem, which is an unsupervised learning technique. This makes the choice of feature selection techniques for our project fairly limited.

Clustering-specific feature selection

In [Liu et al., 03] a number of feature selection strategies that are suitable for the clustering problem are reviewed and compared:

a. **Document Frequency (DF) —** Document Frequency feature selection method chooses only these words that appear in more than a given number of documents. This simple yet efficient selection strategy in a natural way scales to large numbers of documents.

b. **Term Strength (TS) —** Term Strength is computed for pairs of documents for which the similarity measure exceeds a predefined threshold. For a particular term, Term Strength is the conditional probability that this term occurs in one document, given that it occurs in the other document. Because the probabilities may need to be calculated for all possible pairs of documents, TS computational complexity is quadratic with respect to the number of documents.

c. **Term Contribution (TC) —** One disadvantage of DF is that it favours terms that have a high occurrence frequency, not considering the term's distribution among different classes. The Term Contribution strategy alleviates this problem by aggregating the term's contribution to document similarity.

According to the study carried out in [Liu et al., 03], for term reduction exceeding 90% TS and TC slightly outperform DF, at the cost, however, of higher computational demands.

Clustering

The last component in the process chain is the actual clustering algorithm. Several classes of algorithms have been used for the search results clustering task, ranging from adaptations of the classic numerical approaches, such as K-Means in Scatter/Gather [Hearst and Pedersen, 96], to purpose-built methods such as STC [Zamir, 99] and SHOC [Zhang and Dong, 04]. The majority of these algorithms employ the so-called **Vector Space Model (VSM)** of text. In the following sections we provide basic information on VSM as well as describe and analyse several search results clustering algorithms in more detail.

2.1.2 Vector Space Model

In the Vector Space Model (VSM) [Salton, 89], every document in the collection is represented by a multidimensional vector. Each component of such a vector reflects a particular key word or term connected with the given document. The value of each component depends on the degree of relationship between its associated term and the respective document. Many schemes for measuring this relationship, very often referred to as **term weighting**, have been proposed. In the following subsection we review the three most popular.

Term weighting

Term weighting is a process of calculating the degree of relationship (or association) between a term and a document. As the VSM requires that the relationship be described by a single numerical value, let a_{ij} represent the degree of relationship between term i and document j.

Binary weighting

In the simplest case the association is binary: a_{ij}=1 when key word i occurs in document j, a_{ij}=0 otherwise. The binary weighting informs about the *fact* that a term is somehow related to a document but carries no information on the *strength* of the relationship.

Term frequency weighting

A more advanced term weighting scheme is the **term frequency**. In this scheme $a_{ij}=tf_{ij}$ where tf_{ij} denotes how many times term i occurs in document j. Clearly, the term frequency is more informative than the simple binary weighting. Nevertheless, its drawback is that it focuses on local word occurrences only, not considering the global distribution of terms between documents.

Tf-idf weighting

The ***tf-idf*** (term frequency inverse document frequency) scheme aims at balancing the local and the global term occurrences in the documents. In this scheme $a_{ij}=tf_{ij} \cdot \log(N/df_i)$ where tf_{ij} is the term frequency, df_i denotes the number of documents in which term i appears, and N represents the total number of documents in the collection. The $\log(N/df_i)$, which is very often referred to as the *idf* (inverse document frequency) factor, accounts for the global weighting of term i. Indeed, when a term appears in all documents in the collection, $df_i=N$ and thus the balanced term weight is 0, indicating that the term is useless as a document discriminator.

Query matching

The term-document matrix

In the Vector Space Model, a collection of d documents described by t terms can be represented as a $t \times d$ matrix A, hereafter referred to as the **term-document matrix**. Each element a_{ij} of the term-document matrix represents the degree of relationship between term i and document j by means of one of the presented term weighting schemes. The column vectors of A, called **document vectors**, model the documents present in the collection, while the row vectors of A, called **term vectors**, represent the terms used in the process of indexing the collection. Thus, the document vectors *span* the document collection, which means they contain the whole semantic content of the collection.

In the Vector Space Model, a user query is represented by a vector in the column space of the term-document matrix. This means that the query can be treated as a pseudo-document that is built solely of the query terms. Therefore, in the process of query matching, documents must be selected whose vectors are geometrically closest to the query vector. A common

measure of similarity between two vectors is the cosine of the angle between them. In a $t \times d$ term-document matrix A, the cosine between document vector a_j and the query vector q can be computed according to the following formula:

Figure 2.3 The cosine distance formula

$$\cos\theta_j = \frac{a_j^T q}{\| a_j \| \| q \|} = \frac{\sum_{i=1}^{t} a_{ij} q_i}{\sqrt{\sum_{i=1}^{t} a_{ij}^2} \sqrt{\sum_{i=1}^{t} q_i^2}}$$

where a_j is the jth document vector, t is the number of terms and $||a||$ denotes the Euclidean length of vector a. Values of the cosine similarity range from 0.0 in case of no similarity between q and a to 1.0 in case of strict equality between them. Documents whose similarity to the query exceeds a predefined threshold (e.g. 0.8) are returned as the search result.

2.1.3 Numerical clustering algorithms

Historically, the earliest search results clustering algorithms were adaptations of the well-known methods such as K-Means or Agglomerative Hierarchical Clustering (AHC) to the new task. These algorithms require that some **similarity measure** be defined between objects being grouped. In [Hearst and Pedersen, 96] the standard Vector Space Model and the cosine distance are used for this purpose.

K-Means K-Means is an iterative approach whereby clusters are built around k central points[9] called **centroids**. The process starts with k random centroids, and based on the similarity measure, each object is assigned to its nearest centroid. Then, iteratively, based on the content of each group, the respective centroids are updated and objects re-assigned to their now closest centroids. The algorithm stops when there have been no object assignment changes in the last iteration or when a certain number of iterations has been reached.

Agglomerative Hierarchical Clustering The idea of the Agglomerative Hierarchical Clustering (AHC) is to merge two objects, an object and a cluster or two clusters that are most similar into a new group. In this way, the relationships between all objects will be captured in a tree-like structure called **dendrogram**. Based on the dendrogram, a hierarchical groupings of objects can be derived. In [Wroblewski, 03] several strategies of pruning the dendrogram are proposed that increase the effectiveness of AHC in the search results clustering task.

Strong and weak points of numerical algorithms Among the advantages of the classic clustering algorithms is that they are thoroughly researched and fairly easy to implement. The main drawback of these algorithms, however, is that they lack mechanisms of *explaining* the meaning of particular clusters to the end users. Moreover, due to the fact that the property making particular snippets belong to the same cluster is synthetic by its very nature, these algorithms will often create groups that have excellent mathematical parameters but are at the same time completely meaningless from a human user's point of view.

2.1.4 Suffix Tree Clustering

The Suffix Tree Clustering (STC) is a purpose-built search results clustering algorithm, which groups the input texts according to the identical **phrases**[10] these texts share [Zamir, 99]. The rationale behind such approach is that phrases, compared to sets of single keywords, have greater descriptive power. The main reason for that is that phrases retain the relationships of proximity and order between words.

The Suffix Tree Clustering algorithm works in two main phases: **base cluster** discovery phase and base cluster merging phase. In the first phase a **generalised suffix tree** of all texts' sentences is built using words as basic elements. After all sentences have been processed, nodes of the tree contain information about the documents containing particular phrases. Using

[9] k is a parameter of the algorithm and must be determined before clustering starts

[10] For the purposes of this project, we will refer to *any* sequence of words, no matter its grammatical correctness or function, as a phrase

that information documents that share the same phrase are grouped into base clusters of which only those are retained whose score exceeds a predefined threshold. In the second phase of the algorithm, a graph representing relationships between the discovered base clusters is built using information on their similarity. Base clusters belonging to coherent subgraphs of that graph are merged into final clusters.

Strong and weak points of STC

A clear advantage of Suffix Tree Clustering is that it uses phrases to provide concise and meaningful descriptions of groups. However, as noted in [Stefanowski and Weiss, 03], STC's thresholds play a significant role in the process of cluster formation, and they turn out particularly difficult to tune. Also, STC's phrase pruning heuristic tends to remove longer high-quality phrases, leaving only the shorter and less informative ones. Finally, as pointed out in [Zhang and Dong, 04], if a document does not include any of the extracted phrases or just some parts of them, it will not be included in the results although it may still be relevant.

2.1.5 Semantic Hierarchical Online Clustering

The Semantic Online Hierarchical Clustering (SHOC) [Zhang and Dong, 04] is a web search results clustering algorithm that was originally designed to process queries in Chinese. Although it is based on a variation of the Vector Space Model called Latent Semantic Indexing (LSI) and uses phrases in the process of clustering, it is much different from the its predecessors.

To overcome the STC's low quality phrases problem, in SHOC Zhang and Dong introduce two novel concepts: **complete phrases** and a **continuous cluster definition**. Informally, the complete phrases technique tries to maximise the length of the discovered sequences of words, and thus avoids extracting short ones that may turn out difficult to understand[11]. In [Zhang and Dong, 04] an algorithm is proposed that uses a data structure called **suffix array** to identify complete phrases and their frequencies in O(n) time, n being the total length of all processed documents. The continuous cluster definition allows documents to belong to different clusters with different intensity. In a natural way, such a definition addresses the requirement of overlapping clusters. Additionally, it provides a method of ordering documents *within* clusters.

The SHOC algorithm works in three main phases: complete phrase discovery phase, base cluster discovery phase and cluster merging phase. In the first phase, suffix arrays are used to discover complete phrases and their frequencies in the input collection. In the second phase, using Singular Value Decomposition a set of orthogonal base clusters is obtained. Finally, in the last phase, base clusters are merged into a hierarchical structure.

Weak points of SHOC

One of the drawbacks of SHOC is that Zhang and Dong provide only vague comments on the values of thresholds of their algorithm and the method which is used to label the resulting clusters. Additionally, a test implementation of SHOC, which we prepared during the course of our previous research, shows that in many cases the Singular Value Decomposition produces unintuitive, sometimes even close to "random", continuous clusters. The reason for this lies probably the fact that the SVD is performed on document snippets rather than the full texts as it was in its original applications.

SHOC was a source of inspiration for our recent research [Osinski and Weiss, 04] where we showed that Singular Value Decomposition can indeed provide a base for a successful search results clustering algorithm. The novelty of our approach is that we use the SVD to discover a number of short and concise cluster *labels*, to which we then assign content in order to create the actual groups. We call this method a description-comes-first approach and devote to it the next section.

[11] compare: "Senator Hillary" and "Senator Hillary Rodham Clinton"

2.2 The description-comes-first approach

This section describes the search results clustering method on which we will be concentrating during the course of this project. The main idea behind the description-comes-first approach [Osinski and Weiss, 04] is that the process of clustering is *reversed*: we first find meaningful cluster labels and only then assign snippets to them to create proper groups.

Cluster label quality

The main goal of Lingo is to ensure good quality of cluster labels. It is the labels that provide the users with an overview of the topics covered in the results and help them to identify the specific group of documents they were looking for. Therefore, the quality of the search results clustering process as a whole crucially depends on the readability of group descriptions. In our opinion, to help the user to efficiently handle search results, group labels should have the following five characteristics:

a. **Accurate —** A label must accurately describe the contents of its cluster. This property is extremely important as some users will choose to examine more closely only those groups whose labels suggest relevant content. In this way, groups that contain relevant documents but lack accurate description are likely to be disregarded.

b. **Concise —** Search results clustering aims to speed up the process of finding relevant search results. Therefore, group labels must be concise and grammatically well-formed, so that the users can identify groups they are interested in at first glance.

c. **Unambiguous —** In order to be useful for the users, cluster labels must be narrowly focused. In some contexts, groups labelled, for example, "Home Page" or "Latest News" might turn out too general.

d. **Diverse —** Ideally, the set of group labels as a whole would cover all subjects present in the input collection of snippets. This means that not only major topics should have their respective labels, but also the ones that by some users may be considered as outliers.

e. **Distinct —** Among the whole set, cluster labels must differ significantly from each other. This will ensure a high level of dissimilarity between respective clusters.

Differences between Lingo and other algorithms

The main difference between Lingo and other search results clustering algorithms is in the way they try to explain the property that makes snippets in one cluster similar to each other. In previous approaches documents were assigned to groups according to some abstract mathematical properties[12], which would sometimes lead these algorithms down a blind alley of *knowing* that certain documents should be clustered together and at the same time being unable to *explain* the relationship between them in a human-readable fashion. In the description-comes-first approach this problem is avoided by *first* finding readable descriptions and only then trying to create appropriate clusters.

Lingo performs clustering in two major phases: **cluster label induction phase** and **cluster content assignment phase**. In the following sections we describe how dimensionality reduction techniques can be used in the former and the Vector Space Model in the latter phase. We also formulate the description-comes-first approach in pseudo-code and provide an illustrative example.

[12] such as minimised cosine distance from the cluster's centroid

2.2.1 Cluster label induction

During the cluster label induction phase, the term-document matrix A of all input snippets, and the snippets' frequent phrases[13] are used to discover labels for not yet existing clusters. The key component in this phase is a dimensionality reduction algorithm, which is used to produce a low-dimensional basis for the column space of A. Each vector of that basis gives rise to a single cluster label.

Why dimensionality reduction helps to find labels

In linear algebra, basis vectors of a linear space can be perceived as building blocks that create vectors over that space. For example, in [Lee and Seung, 99] a dimensionality reduction technique called Non-negative Matrix Factorisation was shown to be able to produce a part-based representation of human face images. Following this intuition, we believe that in the search results clustering setting each of the building blocks should carry some broader idea referred to in the input collection of snippets. Linear bases produced by particular dimensionality reduction methods can have specific properties, discussed in Section 2.3, that can additionally increase the readability or diversity of cluster labels

For obvious reasons, however, base vectors in their original numerical form are useless as human-readable cluster descriptors. To deal with this problem let us notice that basis vectors are vectors in the original term space of the A matrix. Moreover, frequent phrases or even single words appearing in the input snippets can also be expressed exactly in the same way. Thus, the well-known similarity measures, such as the cosine similarity, can be used to determine which frequent phrase or single word best approximates the verbal meaning of a base vector. The value of the similarity measure can then be taken as the score of the cluster label, which in turn can be used during the process of score calculation for the group as a whole.

2.2.2 Cluster content assignment

In the cluster content assignment phase, snippets are allocated to previously discovered labels in order to create the actual groups. The simplest method here will adopt the Vector Space Model and use cluster labels as queries against the input snippets. In other words, each group will be assigned snippets that the Vector Space Model returned in response to a "query" being its label. In a natural way, this assignment scheme has the ability to create overlapping clusters. Additionally, within clusters snippets can be ordered according to their similarity score obtained from the VSM.

One drawback of the above approach is that the Vector Space Model disregards information about word order and proximity. On the other hand, replacing VSM with exact phrase matching would result in the content assignment being too restrictive and creating very small clusters. An appropriate solution might be using proximity search algorithms, such as these presented in [Sadakane and Imai, 01].

The final stage of the cluster content assignment is calculating cluster score, which would usually take into account its size and the score of its label. Optionally, cluster merging step can also be performed.

2.2.3 Pseudo-code and illustrative example

To summarise our discussion of the description-comes-first approach, in Figure 2.4 we present our algorithm in a form of pseudo-code. Please note that the initial preprocessing phase has also been included.

Figure 2.4 *Lingo pseudo-code*

```
/** Phase 1: Preprocessing */
for each document
{
  do text filtering;
```

[13] sequences of words that appear in the input snippets more than once

```
  identify the document's language;
  apply stemming;
  mark stop words;
}

/** Phase 2: Cluster label induction */
use suffix arrays to discover frequent terms and phrases;
build a term-document matrix A for all snippets;
find a low-dimensional base for the column space of A;
for each base vector
{
  find best-matching frequent phrase;
  add the phrase to the set of cluster labels;
}
prune similar cluster labels;

/** Phase 3: Cluster content assignment */
for each cluster label
{
  use the label to query the input snippets;
  put returned snippets to a cluster described by the label;
}

/** Phase 4: Final cluster formation */
calculate cluster scores;
```

The following example of the description-comes-first approach is based on an implementation [Osinski and Weiss, 04] in which Singular Value Decomposition is used as a dimensionality reduction technique and the cluster content assignment is based on the Vector Space Model.

In Figure 2.5 a collection of d=7 document titles is presented, in which t=5 terms and p=2 phrases appear more than once. In the term-document matrix A each snippet is represented as a column vector and row vectors denote documents' words. Thus, in our example the first row represents the word "Information", second — "Singular", and so on. Similarly, column one represents "Large Scale Singular Value Computations", column two — "Software for the Sparse Singular Value Decomposition" etc.

Figure 2.5 *Lingo example: input data*

The t=5 **terms**:
T1: Information
T2: Singular
T3: Value
T4: Computations
T5: Retrieval

The p=2 **phrases**:
P1: Singular Value
P2: Information Retrieval

The d=7 **documents**:
D1: Large Scale Singular Value Computations
D2: Software Library for the Sparse Singular Value Decomposition
D3: Introduction to Modern Information Retrieval
D4: Using Linear Algebra for Intelligent Information Retrieval
D5: Matrix Computations
D6: Singular Value Analysis of Cryptograms
D7: Automatic Information Organization

The 5x7 **term-document** matrix after column length normalization (tf-idf weighting):

$$A = \begin{pmatrix} 0.00 & 0.00 & 0.56 & 0.56 & 0.00 & 0.00 & 1.00 \\ 0.49 & 0.71 & 0.00 & 0.00 & 0.00 & 0.71 & 0.00 \\ 0.49 & 0.71 & 0.00 & 0.00 & 0.00 & 0.71 & 0.00 \\ 0.72 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.83 & 0.83 & 0.00 & 0.00 & 0.00 \end{pmatrix}$$

The SVD decomposition of A gives five base vectors, which are column vectors of the U matrix in figure Figure 2.6. It can be easily observed that column one corresponds to the topic of "Singular Value Computations" and column two to "Information Retrieval". Let us further assume that k=2 has been chosen as the number of desired clusters. Thus, in further processing U_k — a truncated version of U denoted by a vertical line in the figure — will be used.

Figure 2.6
Lingo example: basis vectors

$$U = \left(\begin{array}{ccc|cc} 0.00 & 0.75 & 0.00 & -0.66 & 0.00 \\ 0.65 & 0.00 & -0.28 & 0.00 & -0.71 \\ 0.65 & 0.00 & -0.28 & 0.00 & 0.71 \\ 0.39 & 0.00 & 0.92 & 0.00 & 0.00 \\ 0.00 & 0.66 & 0.00 & 0.75 & 0.00 \end{array}\right)$$

Bearing in mind our earlier observation that base vectors, frequent phrases and single words can all be expressed as vectors in the original term space of A, let us create the $t \times (p+t)$ matrix P, whose columns will represent frequent phrases and frequent single words appearing in the input snippets. In our example matrix P, shown in Figure 2.7, column one corresponds to the phrase "Singular Value", column two — to "Information Retrieval", column three — to the word "Information" etc. As with the term-document matrix A, P has been created using the *tf-idf* weighting and has Euclidean length-normalised columns.

Figure 2.7
Lingo example: phrase and single word matrix

$$P = \begin{pmatrix} 0.00 & 0.56 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.71 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.71 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0.00 & 0.83 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \end{pmatrix}$$

Assuming that column vectors of both U and P are length-normalized, which is the case in our example, the problem of calculating the cosine distance between every base-vector—phrase-or-term pair comes down to a simple matrix multiplication $M=U_k^T P$. Rows of the M matrix represent base vectors, its columns—phrases and single words, and individual values are the cosine similarities in question. Thus, in a single row, the maximum component will indicate the phrase or single word that best approximates the corresponding base vector. In our example, see Figure 2.8, the first base vector is related to "Singular Value", while the other one to "Information Retrieval", and these phrases will be taken as candidate cluster label E1 and E2, respectively.

Figure 2.8
Lingo example: phrase matching matrix

$$M = U_k^T P = \begin{pmatrix} \mathbf{0.92} & 0.00 & 0.00 & 0.65 & 0.65 & 0.39 & 0.00 \\ 0.00 & \mathbf{0.97} & 0.75 & 0.00 & 0.00 & 0.00 & 0.66 \end{pmatrix}$$

The e=2 **candidate cluster labels**:
E1: Singular Value (score: 0.92)
E2: Information Retrieval (score: 0.97)

To allocate snippets to the two candidate labels, let us define matrix Q in which each label is represented as a column vector exactly in the same way as in the P matrix. Further, let $C=Q^T A$, where A is the original term-document matrix. In this way, element c_{ij} of the C matrix indicates the strength of membership of the jth document in the ith group.

A snippet will be added to a cluster if the corresponding element of the C matrix exceeds the Assignment Threshold. From our observations, values of the threshold falling between 0.15 and 0.30 yield clusters of best quality. Finally, as it is possible that some snippets may match neither of the cluster labels, a special group labelled e.g. "Other topics" can be created in which such snippets should be placed.

The Q and C matrices for our example collection of snippets as well as the final clusters are shown in Figure 2.9. Cluster scores have been calculated as a product of cluster label score and the number of documents contained in the cluster, and then normalised.

Figure 2.9
Lingo example: cluster content assignment

$$Q = \begin{pmatrix} 0.00 & 0.56 \\ 0.71 & 0.00 \\ 0.71 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.83 \end{pmatrix} \qquad C = Q^T A = \begin{pmatrix} 0.69 & 1.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0.00 & 0.00 & 1.00 & 1.00 & 0.00 & 0.00 & 0.56 \end{pmatrix}$$

Information Retrieval [1.0]
Introduction to Modern Information Retrieval [1.00]
Using Linear Algebra for Intelligent Information Retrieval [1.00]
Automatic Information Organization [0.56]

Singular Value [0.95]
Software Library for the Sparse Singular Value Decomposition [1.00]
Singular Value Analysis of Cryptograms [1.00]
Large Scale Singular Value Computations [0.69]

Other Topics
Matrix Computations

2.3 Dimensionality reduction techniques

In this section we review the three matrix factorisation techniques, Singular Value Decomposition (SVD), Non-negative Matrix Factorisation (NMF) and Local Non-negative Matrix Factorisation (LNMF), which can be used as dimensionality reduction methods in the description-comes-first clustering and therefore are the main focus of this project. Matrix factorisations have proved efficient in various computing tasks, such as information retrieval or image processing, for two main reasons. First of all, by reducing the number of dimensions in which image or text data is represented, frequently by two or three orders of magnitude, substantial savings can be achieved in disk storage cost. More importantly, many algorithms cannot deal with high-dimensional "raw" data either due to an intolerable performance loss or simply due to prohibitive computational costs.

Applications of matrix factorisation

Different dimensionality reduction techniques have been used in various areas of computing. In [Turk and Pentland, 91] Principal Component Analysis (PCA) was used as a base of a face recognition algorithm. Further work in this area resulted in methods using NMF [Lee and Seung, 99] and—very recently—LNMF [Feng et al., 02]. Singular Value Decomposition is the key mathematical concept underlying the Latent Semantic Indexing [Berry et al., 95], which proved fairly efficient in document retrieval tasks. Text clustering algorithms using SVD [Zhang and Dong, 04] or NMF [Xu et al, 03] have also been proposed.

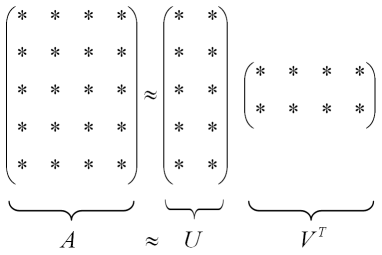
To introduce the general concept of dimensionality reduction and its relation to matrix factorisation, let us denote a set of d t-dimensional[14] data vectors[15] as columns of a $t \times d$ matrix A. It is usually the case that the rank r_A of such a matrix, which is equal to the size of the basis of the linear space it spans, is equal or near to $\min(t, d)$. The aim of a dimensionality reduction technique is to find A', which is a good approximation of A and has rank k, where k is significantly smaller than r_A. For this reason, the A' matrix is often referred to as the k-rank approximation of A.

The task of factorisation, or decomposition, of matrix A is to break it into a product of two[16] matrices U and V so that $A \approx UV^T = A'$, the sizes of the U and V matrices being $t \times k$ and $d \times k$, respectively. Figure 2.10 illustrates this situation for $t=5$, $d=4$ and $k=2$. Intuitively, columns of the U matrix can be thought of as base vectors of the new low-dimensionality linear space,

[14] In the related literature the numbers of rows and columns are usually denoted by m and n, respectively. In this thesis, however, we have decided to adopt a convention that directly relates to a term-document-matrix having t rows and d columns.
[15] In image processing these vectors can represent e.g. bitmaps, and in text processing, they will correspond to documents encoded using the Vector Space Model.
[16] In case of the Singular Value Decomposition the number is three

and rows of V as corresponding coefficients that enable to approximately reconstruct the original data.

Figure 2.10 Matrix factorisation results



An important problem of dimensionality reduction is how to measure the quality of low-rank approximation. A commonly employed approach is to calculate the Frobenius norm of the difference between the A and A' matrices:

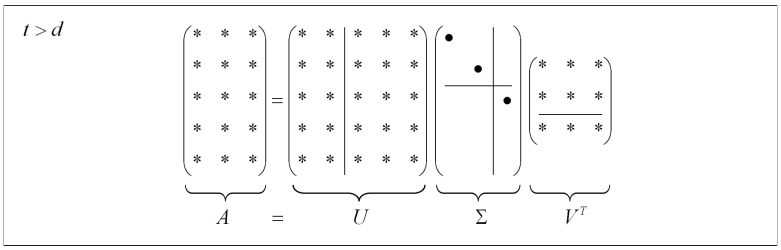
$$\|A - A'\|_F = \sqrt{\sum_{i=1}^{t}\sum_{j=1}^{d}(a_{ij} - a'_{ij})^2}$$

The lowest possible value of the above quality measure is zero and it indicates that a strict equality $A=UV^T$ has been achieved.

2.3.1 Singular Value Decomposition

Mathematical background Singular Value Decomposition breaks a $t \times d$ matrix A into three matrices U, Σ and V such that $A=U\Sigma V^T$. In figure Figure 2.11 the relative sizes of the three matrices are shown when $t>d$.

Figure 2.11 Singular Value Decomposition results



U is a $t \times t$ orthogonal matrix whose column vectors are called the left singular vectors of A, V is a $d \times d$ orthogonal matrix whose column vectors are termed the right singular vectors of A, and Σ is a $t \times d$ diagonal matrix having the singular values of A ordered decreasingly ($\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_{min(t,d)}$) along its diagonal. The rank r_A of matrix A is equal to the number of its non-zero singular values. The first r_A columns of U form an orthogonal basis for the column space of A.

A k-rank approximation of A can be obtained by multiplying the first k columns of the U matrix by the first k singular values from the Σ matrix and the first k rows of the V^T matrix as shown in Figure 2.11. Noteworthy is the fact that for given k, SVD produces the optimal k-rank approximation with respect to the Frobenius norm.

Algorithms A wide variety of implementations of SVD are available on the Internet free of charge. Lapack[17] provides a broad range of linear algebra algorithms including SVD written in Fortran77. There also exist implementations of SVD in the Java language, Colt Open Source Libraries for High Performance Scientific and Technical Computing in Java[18] being the best benchmarked and most up-to-date package. References to other Java libraries are available on the Java Numerics homepage[19].

[17] *http://www.netlib.org/lapack/*
[18] *http://hoschek.home.cern.ch/hoschek/colt/*
[19] *http://math.nist.gov/javanumerics/*

Applicability to search results clustering

During our previous research we prepared an implementation of the description-comes-first clustering algorithm in which SVD was used as the dimensionality reduction technique. Below we summarise features of SVD that we found advantageous to our approach.

a. **Orthogonal basis —** SVD produces an orthogonal basis for the low-dimensional linear space. Intuitively, orthogonal vectors can be perceived as perpendicular[20] or having nothing in common. For this reason the orthogonality of base vectors should lead to a high level of diversity among the candidate cluster labels.

b. **Ordered base vectors —** SVD orders the base vectors in the U matrix according to their significance to the process of reconstruction of the original matrix A. This information can be used during the evaluation of candidate cluster labels.

c. **Seeding not required —** In contrast to NMF and LNMF, the SVD algorithm does not require prior initialisation of the resulting matrices. In this way, with no random seeding, for the same set of input snippets always the same set of candidate cluster labels will be generated, which may not be the case with randomly-initialised NMF or LNMF.

d. **Support for calculating k —** Apart from the base vector matrix U and the coefficient matrix V, in the Σ matrix SVD provides a set of decreasingly ordered singular values of A. As shown in [Zhang and Dong, 04] these values can be used to estimate the optimal number of candidate cluster labels.

As pointed out in the introduction, Singular Value Decomposition proved fairly efficient as a basis for the description-comes-first approach. However, we muse be aware of some other characteristics of SVD described below that have a negative impact on the quality of candidate cluster labels.

a. **Negative values in base vectors —** To achieve the orthogonality and optimality of the k-rank approximation some components of the SVD-derived base vectors may have to be negative. This makes such components hard to interpret in terms of their verbal meaning. Moreover, although in practice the cosine distance measure seems to work well in the SVD-based cluster label induction phase (see Section 2.2.1), interpretation of the similarity between phrases and base vectors would be more straightforward if the latter contained only non-negative values.

b. **Not a real part-based decomposition —** The fact that the U matrix contains a number of negative values makes it difficult to think of the base vectors as purely additive parts that can be combined together to re-create the original data. It is more natural for a non-negative input matrix A to be represented by a set of non-negative parts, even if they are not strictly orthogonal.

2.3.2 Non-negative Matrix Factorisation

Mathematical background

The Non-negative Matrix Factorisation (NMF) was introduced in [Lee and Seung, 99] as means of finding part-based representation of human face images. More formally, given k as the desired size of the basis, NMF decomposes a $t \times d$ non-negative matrix A into two non-negative matrices U and V such that $A \approx UV^T$, the sizes of U and V being $t \times k$ and $d \times k$, respectively. As described in the introductory part of this section, the U matrix can be regarded as a set of k base vectors and V as a coefficient matrix. An important property of NMF is that by imposing the non-negativity constraints it allows only additive, and not subtractive, combinations of base vectors.

Algorithms

In [Lee and Seung, 99] an approach to computing NMF has been suggested that is similar to the Expectation Maximisation (EM) algorithms. The basic idea behind this approach is to randomly initialise the U and V matrices and then iteratively update their content based on

[20] orthogonal vectors are indeed perpendicular in a two-dimensional space

some objective function. In [Lee and Seung, 01] **updating rules** for two different objective functions — Euclidean Distance and Kullback-Leibler Divergence — have been proposed and proved to monotonically decrease the value of these functions. Below we present the two functions along with their corresponding updating formulae.

The **Euclidean Distance** algorithm aims to calculate the NMF by minimising the Euclidean distance between each column of A and its approximation $A=UV^T$:

Figure 2.12 Euclidean Distance objective function

$$\Theta_{NMF_E} = \sum_{i=1}^{t}\sum_{j=1}^{d}\left(A_{ij} - \sum_{l=1}^{k} U_{il}V_{jl}\right)^2$$

In [Lee and Seung, 01] the following[21] two-step updating rules have been proposed and proved correct:

Figure 2.13 Euclidean Distance updating rules

$$V_{ij} \leftarrow V_{ij}\frac{[A^TU]_{ij}}{[VU^TU]_{ij}} \qquad U_{ij} \leftarrow U_{ij}\frac{[AV]_{ij}}{[UV^TV]_{ij}}$$

In the above figure the $[\bullet]_{ij}$ notation indicates that the corresponding multiplications and divisions are performed element by element.

In the **Divergence Algorithm** the divergence, or entropy, measure is used as the objective function:

Figure 2.14 Divergence objective function

$$\Theta_{NMF_D} = \sum_{i=1}^{t}\sum_{j=1}^{d}\left(A_{ij}\log\frac{A_{ij}}{\sum_{l=1}^{k}U_{il}V_{jl}} - A_{ij} + [UV^T]_{ij}\right)$$

The following three-step updating rules have been proved to monotonically decrease the values of the above criterion:

Figure 2.15 Divergence updating rules

$$V_{ij} \leftarrow V_{ij}\left[\left(\frac{[A]_{ij}}{[UV^T]_{ij}}\right)^T U\right]_{ij} \qquad U_{ij} \leftarrow U_{ij}\left[\frac{[A]_{ij}}{[UV^T]_{ij}}V\right]_{ij} \qquad U_{ij} \leftarrow \frac{U_{ij}}{\sum_{l=1}^{k}U_{lj}}$$

As empirically established by the authors in [Wild, 03], the Euclidean Distance algorithm tends to produce slightly better approximations, while the Divergence algorithm converges faster. They also claim that one update in the Euclidean Distance algorithm involves a smaller number of floating point operations than a corresponding update in the Divergence algorithm. The implementation of these two algorithms we have prepared, however, shows that it is not the case. The performance of the Euclidean Distance crucially depends on the matrix multiplication strategy employed to compute the denominators of the updating rules[22] shown in Figure 2.13. With the right strategy chosen, the Euclidean Distance algorithm significantly outperforms the Divergence algorithm.

Applicability to search results clustering

Being able to generate a low-dimensional basis for the original term-document matrix of snippets, NMF can potentially be used as part of the description-comes-first algorithm. Below we summarise possible advantages of NMF in this setting.

---

21 We have adjusted the notation of the original formulae to match our conventions

22 More precisely, the VU^TU expression in Figure 2.13 can be computed in two ways: $(VU^T)U$ or $V(U^TU)$. As it can be easily shown, the former requires $2kdt$ multiplications, whereas the latter only $k^2(d+t)$. In view of k being much smaller than $\min(d, t)$, the latter matrix multiplication order is significantly faster. Exactly the same reasoning applies to the UV^TV expression.

a. **Non-negative base vectors —** The non-negativity constraint the NMF imposes on the base vectors leads to a truly part-based representation of the input snippets. This enables us to interpret the verbal meaning of such base vectors in an intuitive way, i.e. the greater value of a component in the vector, the more significant the corresponding term is in explaining its meaning. This also makes the interpretation of the cosine similarity between phrases and base vectors less ambiguous.

b. **Iterative algorithm —** Both NMF algorithms we presented have iterative nature, which enables the practical implementations to balance the quality of the results and the computation time. This is especially important in on-line search results clustering systems, which can choose to lower the number of iterations per query to serve more users during a high-load period.

Before NMF is used as part of a fully-fledged search results clustering system, several issues must be addressed, which are highlighted below.

a. **The choice of k —** In contrast to SVD, for Non-negative Matrix Factorisation the desired size of the low-dimensionality space must be known in advance – NMF does not provide any explicit clues as to it. Therefore, the clustering algorithm must estimate k based on other criteria e.g. the number of snippets.

b. **No ordering among base vectors —** Unlike SVD, NMF does not order base vectors according to their significance. Thus, only the level of similarity between frequent phrases and these vectors can be used in the label scoring formula.

c. **Initialisation —** Both NMF algorithms presented above use random initialisation of the U and V matrices. This will cause a clustering algorithm based on NMF to produce different set of groups every time it is run, which may be undesirable in certain applications or for evaluation purposes. In [Wild, 03] alternative methods of seeding NMF are discussed.

d. **Lack of orthogonality —** The non-negativity of NMF-derived basis is achieved at the cost of the base vectors not being orthogonal, which may cause some of the NMF-induced cluster labels to be more similar to each other than desired. To alleviate this problem pruning of candidate cluster labels may be needed.

2.3.3 Local Non-negative Matrix Factorisation

Mathematical background Local Non-negative Matrix Factorisation is a variation of NMF introduced in [Feng et al., 02]. Similarly to NMF it seeks to decompose a $t \times d$ non-negative matrix A into two non-negative matrices U and V such that $A \approx UV^T$, where k is the desired size of the low-dimensionality space and the sizes of U and V are $t \times k$ and $d \times k$, respectively. Unlike NMF, this technique imposes three additional constraints on the U and V matrices, which aim to expose the local features of the examples defined in the A matrix:

a. **Maximum Sparsity in V —** The coefficient matrix V should contain as many zero components as possible. This ensures that the number of base vectors required to represent the original matrix A is minimised.

b. **Maximum Expressiveness of U —** Retain only those components of B that carry most information about the original matrix A.

c. **Maximum Orthogonality of U —** Base vectors should be as orthogonal to each other as possible. This will reduce the redundancy in the basis as a whole.

Algorithms Similarly to NMF, the algorithm for calculating LNMF also uses the iterative updating technique, the difference between them being only in the forms of objective function and updating

formulae. In [Feng et al., 02] the following divergence function has been proposed that incorporates the LNMF's three additional constraints:

Figure 2.16 LNMF objective function

$$\Theta_{LNMF} = \sum_{i=1}^{t}\sum_{j=1}^{d}\left(A_{ij}\log\frac{A_{ij}}{\sum_{l=1}^{k} U_{il}V_{jl}} - A_{ij} + [UV^T]_{ij} \right) + \alpha\sum_{i=1}^{k}\sum_{j=1}^{k} W_{ij} + \beta\sum_{i=1}^{k} H_{ii}$$

In the above formula $a, \beta > 0$ are some constants, which will be eliminated during derivation of the updating rules, and $W=U^TU$ and $H=V^TV$. The following updating rules have been proposed that lead to a monotonic decrease of the above objective function:

Figure 2.17 LNMF updating rules

$$V_{ij} \leftarrow \sqrt{V_{ij}\left[\left(\frac{[A]_{ij}}{[UV^T]_{ij}}\right)^T U\right]_{ij}} \qquad U_{ij} \leftarrow U_{ij}\left[\frac{[A]_{ij}}{[UV^T]_{ij}}V\right]_{ij} \qquad U_{ij} \leftarrow \frac{U_{ij}}{\sum_{l=1}^{k} U_{lj}}$$

As shown in [Wild, 03], compared to NMF the basis computed by LNMF is much more sparse (i.e. contains more zero elements) and much more orthogonal. This, however, is achieved at the cost of higher approximation error and much slower convergence.

Applicability to search results clustering

With the additional constraints it imposes on the base vectors, LNMF can have the following advantages as part of the description-comes-first clustering algorithm:

a. **Non-negative base vectors and iterative algorithm —** Being a variation of NMF, Local Non-negative Matrix Factorisation inherits all its advantages, including the non-negativity of base vectors and the iterative nature of the NMF algorithm.

b. **Highlighting local features —** The fact that LNMF promotes sparseness of the base vectors should result in less ambiguous matching between these vectors and frequent phrases.

c. **Better orthogonality of base vectors —** Orthogonality among base vectors is desirable as it guarantees high diversity of candidate cluster labels. This in turn is the key to generating well-separated clusters in the description-comes-first approach.

Below we summarise the key problems that must be resolved in order for the LNMF to be efficient in the description-comes-first clustering algorithm.

a. **Choice of *k*, initialisation and base vectors ordering —** As LNMF is only a slight variation of NMF, it does not remove its three major deficiencies, see Section 2.3.2 for more details.

b. **Slower convergence —** As shown in [Wild, 03] LNMF converges much slower than NMF. This means that more time will be needed for the cluster induction phase to complete.

c. **Lower approximation quality —** Special properties of LNMF's base vectors are achieved at the cost of lowered overall quality of approximation. We believe, however, that for the description-comes-first approach more important is the localised character of base vectors than perfect accuracy of approximation.

2.3.4 Concept Decomposition

Concept Decomposition [Dhillon and Modha, 01] is a dimensionality reduction technique based on the Spherical K-Means clustering algorithm. For a $t \times d$ matrix A and given k Concept Decomposition generates a $t \times k$ matrix U and a $d \times k$ matrix V such that $A \approx UV^T$.

The K-Means algorithm

K-Means is an iterative clustering algorithm in which clusters are built around k central points called **centroids**. The algorithm starts with a set of centroids corresponding to randomly selected columns of A and to each centroid assigns columns of A that are closest to that centroid with respect to the cosine similarity measure. Then, repeatedly, for each group of columns, a new centroid is calculated as a mean of all column vectors in that group and column assignments to their now closest centroids are changed accordingly. The algorithm stops when no column reassignments are needed or when certain number of iteration has been completed. Spherical K-Means is a variation of the base K-Means algorithm in which all centroids are normalised to have unit Euclidean length.

In Concept Decomposition each column of the U matrix directly corresponds to one centroid obtained from the K-Means algorithm. Various methods have been proposed to derive the coefficient matrix V. In the simplest case, row d of matrix V contains a non-zero element in column c only if the K-Means algorithm assigned column vector d of matrix A to centroid c. Clearly, in this scheme, columns of A are represented by their closest centroids, which may have negative impact on approximation accuracy. In [Dhillon and Modha, 01] and [Wild, 03] more advanced techniques of calculating the V matrix are proposed.

Applicability to search results clustering

In our experiments Concept Decomposition will be used only as a base line for comparisons with other dimensionality reductions presented in this section. Intuitively, because K-Means is based around averaged centroids of groups of documents, it should be able to successfully detect major themes in the input snippets. However, it may prove less efficient in identifying topics represented by relatively small groups of documents.

3 Implementation

In this chapter we provide details on the software components developed during the course of this project. We start with remarks on our implementation of matrix factorisations in Java. Then we proceed to a brief description of the Carrot^2 framework, within which our clustering algorithms operate. We conclude this chapter with a list of text processing components and applications this project contributes to the Carrot^2 framework.

3.1 Matrix factorisation in Java

One of the major tasks of our project was implementation of three matrix factorisation algorithms, two variants of Non-negative Matrix Factorisation and Local Non-negative Matrix Factorisation, in Java. We have decided to first develop working prototypes in MatLab, which would then be used to verify the correctness of the corresponding Java programs. We also looked into how the computation time of the initial versions of the routines could be further improved.

3.1.1 MatLab prototypes

The implementations we prepared were based on multiplicative updating rules presented in Section 2.3. Figure 3.1 presents the MatLab code that computes NMF with Euclidean Distance minimisation[23].

Figure 3.1 Euclidean Distance NMF algorithm MatLab code

```
%
% Non-negative Matrix Factorisation
% Euclidean Distance Algorithm
%
% A - input matrix
% U - base vectors matrix
% V - coefficient matrix
% C - approximation quality for subsequent iterations
%
function [U, V, C] = nmf_ed(A)
    [t, d]  = size(A);
    k       = 2;            % the desired number of base vectors
    maxiter = 50;           % the number of iterations
    eps     = 1e-9;         % machine epsilon

    U = rand(t, k);         % initialise U randomly
    V = rand(d, k);         % initialise V randomly

    for iter = 1:maxiter
        U = U.*((A*V+eps)./(U*V'*V+eps));    % update U
        V = V.*((A'*U+eps)./(V*U'*U+eps));   % update V
        C(1, iter) = norm((A-U*V'), 'fro');  % approximation quality
    end
```

If matrix A is very sparse (i.e. contains many zero elements), it may happen that the $0/0$ operation will be attempted during the element-by-element division while updating U or V. In [Wild, 03] this problem is solved by adding machine epsilon both to the numerator and the

[23] MatLab codes for all factorisation algorithms are included in the electronic submission of this dissertation.

denominator of the expression. As the machine epsilon is very small, it only matters when a $0/0$ division would occur, making the expression equal to 1 in that case.

3.1.2 Java implementations

Our Java translation of the above code relies on the matrix computaion API provided as part of the Colt framework[24]. In Figure 3.2 we present a fragment[25] of the `NonnegativeMatrixFactorizationED` class, which corresponds to the MatLab code from Figure 3.1.

Figure 3.2 Euclidean Distance NMF algorithm Java code

```
// Machine epsilon
double eps = 1e-9;

// Seed U and V with initial values
U = doubleFactory2D.make(A.rows(), k);
V = doubleFactory2D.make(A.columns(), k);
seedingStrategy.seed(A, U, V);

// Temporary matrices
DoubleMatrix2D T = doubleFactory2D.make(k, k);
DoubleMatrix2D UT1 = doubleFactory2D.make(A.rows(), k);
DoubleMatrix2D UT2 = doubleFactory2D.make(A.rows(), k);
DoubleMatrix2D VT1 = doubleFactory2D.make(A.columns(), k);
DoubleMatrix2D VT2 = doubleFactory2D.make(A.columns(), k);

for (int i = 0; i < maxIterations; i++)
{
    // Update V
    U.zMult(U, T, 1, 0, true, false);   // T <- U'U
    A.zMult(U, VT1, 1, 0, true, false); // VT1 <- A'U
    V.zMult(T, VT2);                    // VT2 <- VT
    VT1.assign(plusEps);
    VT2.assign(plusEps);
    VT1.assign(VT2, Functions.div);     // VT1 <- VT1 ./ VT2
    V.assign(VT1, Functions.mult);      // V <- V .* VT1

    // Update U
    V.zMult(V, T, 1, 0, true, false);   // T <- V'V
    A.zMult(V, UT1);                    // UT1 <- AV
    U.zMult(T, UT2);                    // UT2 <- UT
    UT1.assign(plusEps);
    UT2.assign(plusEps);
    UT1.assign(UT2, Functions.div);     // UT1 <- UT1 ./ UT2
    U.assign(UT1, Functions.mult);      // U <- U .* UT1

    iterationsCompleted++;
}
```

The temporary matrices `T`, `UT1`, `UT2`, `VT1` and `VT2` have been introduced to avoid continuous allocation/freeing of memory during the updating process.

3.1.3 Performance improvements

Due to their iterative nature, matrix factorisation algorithms are fairly costly in terms of computation time. As shown in Table 3.1, this problem is especially acute in Java implementations. For this reason we have decided to seek for ways of improving the performance of the Java factorisation routines.

Architecture-specific matrix routines

ATLAS ATLAS[26] (Automatically Tuned Linear Algebra Software) is a freely available architecture-specific implementation of the BLAS[27] (Basic Linear Algebra Subprograms) package. It provides highly-optimised versions of vector-vector, matrix-vector and matrix-matrix multi-

[24] *http://hoschek.home.cern.ch/hoschek/colt/*
[25] Full source code of all factorisation algorithms is included in the electronic submission of this dissertation.
[26] *http://math-atlas.sourceforge.net/*
[27] *http://www.netlib.org/blas/*

plication routines, which use various CPU-specific extensions such as SSE and SSE2 on Intel or AltiVec on PowerPC.

In order to be able to use these optimised matrix routines in our Java programs we needed to create a bridge between the architecture-independent Java code and the machine-dependent ATLAS. To this end, using the Native Numerical Interface[28] (NNI), we extended the Colt framework with an implementation of its `DenseDoubleMatrix2D` class that is backed by the much faster machine-specific ATLAS routines. According to our simple experiment[29], as shown in Table 3.1, this can result in almost five-fold improvement in execution time.

Table 3.1 Java vs. native matrix routines (NMF-ED)

Matrix size	K	Iterations	Time [ms]		Speedup
			Pure Java	Native	
280×100	25	10	320	141	2.26
420×150	37	10	1031	341	3.02
560×200	50	10	2553	802	3.18
840×300	75	10	8161	2133	3.82
1120×400	100	10	18777	4507	4.16
1400×500	125	10	37464	8212	4.56

One potential disadvantage of this solution is the loss of portability: ATLAS libraries must be compiled separately for each target platform. To address this problem, our Java interface to ATLAS falls back to pure Java platform-independent code when ATLAS is unavailable on the particular machine.

Stop criterion

A common problem in construction of iterative numerical algorithms is the decision as to when the algorithm has converged and can stop the computations. A popular approach here is to monitor the value of some approximation quality measure that the algorithm aims to minimise/maximise and stop the computations when no substantial improvement has been made with respect to that measure in the last iteration.

For the the problem of matrix factorisation, a commonly used approximation quality metric is the Frobenius norm of the difference between the original matrix A and its approximation $A'=UV^T$, described in Section 2.3. As constructing the A' matrix involves a fairly expensive matrix multiplication, evaluation of the stop criterion may significantly increase to the total computational expense of one iteration of the algorithm. Eliminating that costly step is therefore another way of speeding up the factorisation algorithm as a whole.

To avoid continuous recalculation of the approximation quality measure, we have decided that the total number of iterations required for a factorisation algorithm to converge will be estimated in advance using a simple linear model:

$$number\text{-}of\text{-}iterations = a \times d + b \times k + c$$

where d is the number of rows of the A matrix and k is the desired number of base vectors. For each factorisation algorithm, we estimated the parameters of the model, a, b and c, under a number of assumptions correct for the search results clustering application, e.g. that the number of non-zero elements in the A matrix is not larger than 2%.

One possible problem with this approach is that for some particular input matrices the estimated number of iterations may be not enough for the algorithm to converge, which would result in a less accurate approximation. However, we feel that for an on-line search results clustering algorithm, more important is the speed of processing than the perfect accuracy of the underlying matrix factorisation.

[28] *http://www.math.uib.no/~bjornoh/mtj/nni/*

[29] The experiment was performed on a Pentium M (Centrino) 1.3GHz, 512MB RAM running Sun JDK 1.4.2_04.

3.2 Search results clustering in Carrot²

All software components developed during the course of our project, are designed to operate within a text processing framework called Carrot². This section we devote to the framework itself as well as to the text processing components our project contributed to it.

3.2.1 The framework

Carrot² is a freely available[30] Open Source framework for experiments with processing and visualisation of search results. Being a 100% pure Java application, Carrot's architecture is based upon on a set of components that can cooperate with each other either in a distributed fashion by exchanging XML data or locally within a single Java application.

Components available in Carrot²

Three general types of components are available in Carrot²: input, filter and output components. The task of an **input component** is to deliver data for further processing. An example input component could, for example, fetch the results generated by some web search engine in response to the Carrot's user query. **Filter components** transform their input data in some way. An example transformation could be tokenization or stemming of the input data. Clearly, in Carrot², our search results clustering algorithms act as filter components. Finally, the role of an **output component** is to present its input data to the end user. Carrot² applications can flexibly arrange individual components into processing chains beginning with an input component, followed by any number of filter components and terminated with an output component.

Why use Carrot²?

The main advantage of using Carrot² in our project is that having ready-to-use tokenization and stemming components, we will not be distracted from the main focus of this project, which is developing and evaluating different variants of a search results clustering algorithm. Additionally, Carrot² already contains two[31] clustering algorithms similar to Lingo, Suffix Tree Clustering (see Section 2.1.4) and Tollerance Rough Set Clustering (TRC) [Lang, 04]. Thus, not only will we be able to compare different variants of the description-comes-first approach with each other, but also we will have the opportunity to compete against STC and TRC

3.2.2 New components

During the course of our project we contributed the following components to the Carrot² framework:

a. **Lingo Clustering Filter Component** — This component implements the description-comes-first clustering method described in Section 2.2. One of the parameters the Lingo component can take is the algorithm that is to be used to perform dimensionality reduction. In this way we were able to easily compare variants of the description-comes-first approach using different matrix factorisation algorithms. Refer to Section 5.2 for the detailed results.

   Another parameter of the Lingo filter component is the clustering quality level, which can assume three values: low (1), medium (2) and high (3). The higher the quality level, the more iterations the factorisation algorithms perform and the more terms are selected to be included in the term-document matrix. For the influence of this parameter on the clustering results refer to Section 5.3.

b. **ODP Data Input Component** — The task of this component is to fetch all snippets belonging to the ODP categories specified in the query and pass them down

[30] *http://carrot.cs.put.poznan.pl*

[31] In fact, there are two more search results clustering algorithms in Carrot². These, however, generate hierarchical groups and hence are difficult to directly compare with Lingo.

the processing chain. As the ODP data is provided as one extremely large XML file (nearly 2GB of text), we had to implement a simple indexing engine that would split the file into manageable pieces and thus allow quick access to individual ODP categories.

c. **Cluster Metrics Output Component** — The purpose of the cluster metrics output component is to calculate the cluster quality metrics described in Section 4.4.

d. **Matrix Computation Library** — This independent Carrot^2 component provides a number of matrix factorisation algorithms as well as some utility routines. If machine-specific ATLAS libraries are present, they will be used to speed up the computation. Otherwise, the slower machine-independent Java code is used.

Clustering benchmark application

To perform the experiments described in Chapter 5 we implemented a simple command-line Carrot^2 application for benchmarking search results clustering algorithms. For given set of ODP input component queries the application performs clustering using selected algorithms available in the framework and saves the individual and aggregated results as XML files. XSLT stylesheets for transforming the results into HTML and Excel files are also provided. In Figure 3.3 we show the application's HTML output for an example ODP data set.

Figure 3.3 *Example HTML output from the clustering benchmark application*

Source code availability

Complete source code of the Carrot^2 framework including components developed as part of this project can be downloaded from SourceForge.net CVS repository at *http://www.sourceforge.net/projects/carrot2*. Please refer to the `readme.txt` file in the main checkout directory for building instructions.

4 Evaluation

In this chapter we give details on the evaluation methodology we have decided to use for the purposes of this project. We start with outlining some general problems involved in assessment of clustering results. Then, we explain how data gathered within the Open Directory Project (ODP) can be used to implement the merge-then-cluster assessment scheme. Finally, we define three cluster quality measures, Topic Coverage, Snippet Coverage and Cluster Contamination to be calculated in that scheme.

4.1 Problems

The main goal of search results clustering is to help human users to better understand search results and to perform their searches more effectively. Therefore, algorithms should aim to maximise the usefulness of clusters they produce. Unfortunately, the term *usefulness for human users* involves very subjective judgments resulting from different preferences different people may have. This makes it particularly difficult, if at all possible, to automatically evaluate the real-world performance of search results clustering. Below we list a number problems arising in this area.

a. **Subjective judgements —** The ultimate performance test for a search results clustering algorithm would be to verify whether it can help human users to perform their searching tasks more efficiently. However, as shown in [Macskassy et al., 98], humans themselves have problems with creating consistent document clusters. Therefore, it will be extremely difficult to construct an automatic evaluation procedure that takes into account the subjective nature of perception of clustered search results.

b. **Many "correct" solutions —** As clustering is an example of an unsupervised learning technique, in a natural way there will be more than one "correct" solution for one data set. For example, for an input collection containing documents about two films, "The Lord of the Rings" and "Blade Runner", the most obvious groups would correspond to the two titles. However, clusters such as "Film Review" or "Soundtracks" that present different aspects of the same information are equally justified.

c. **Evaluation of cluster labels —** In search results clustering, group labels play a very important role. Unfortunately, the assessment of correspondence between the content of a cluster and its label is difficult to express in terms of a formal procedure.

d. **No standard collections for search results clustering —** To the best of our knowledge, there are no standard test collections for the problem of search results clustering. Such a collection would be based around small groups (5-100) of document excerpts (snippets), possibly containing some noise resulting from the process of snippet extraction, and accompanied by a reference cluster structure.

4.2 Methods

Several methods of evaluation of clustering results have been proposed, three of which we summarise below.

4.2.1 Standard IR metrics

In the standard IR approach, the quality of clustering is evaluated in terms of the two standard Information Retrieval metrics: precision and recall. One variant of such a method is where the contents of the top-scoring cluster is assumed to be the set of retrieved documents for some query [Zamir, 99]. Provided that the set of relevant documents is known a priori or had been identified in advance, the precision and recall can be easily calculated.

One drawback of this method is that no standard document collections for the problem of search results clustering have been defined so far. Also, the results in this approach can be strongly influenced by the ambiguity of human relevance judgments.

4.2.2 User evaluation

This evaluation method relies on the real users' opinions about the quality of clustering. All the necessary data can be collected from specially prepared questionnaires or from web server logs.

While this approach is potentially capable of verifying whether the clustering algorithm satisfies its users' needs, it still has some drawbacks. First of all, the results may be influenced by the level of the users' experience with search engines. Secondly, this type of evaluation cannot be performed automatically or on-demand, which makes it unsuitable for experiments with large amounts of data, e.g. for tuning of algorithms' parameters.

4.2.3 Merge-then-cluster approach

The merge-then-cluster method relies on a special collection of texts created by merging smaller document groups each of which relates to some well-defined topic. Given such a mixture of documents an effective clustering algorithm should be able to identify all the original topics.

Reliability of this approach largely depends on the way the correspondence between the automatically generated clusters and the original topics is measured. The similarity between two sets of clusters can be expressed as a single numerical value using e.g. mutual-information measures [Dom, 01]. One drawback of such measures is that the smallest difference between the automatically generated clusters and the reference groups will be treated as the algorithm's mistake, even if the algorithm made a different but equally justified choice (e.g. splitting a large original group into sub-groups). The alternative cluster quality measures we propose in this chapter aim to alleviate this problem.

What should be emphasised about the merge-then-cluster approach is that it can be performed on-demand and fully automatically, which enables large-scale experiments. It is mainly for this reason that we have decided to use the merge-then-cluster based on the Open Directory Project data to evaluate our work.

4.3 Open Directory Project

Open Directory Project[32] is a large and fast-growing human-edited hierarchical directory of the Web. At the time of writing, ODP contained web page references arranged in over 630,000 thematic categories. For several reasons ODP is a good source of data for the purposes of search results clustering evaluation:

a. **Document summaries** — Each web page reference stored in ODP is accompanied by a short summary written by a human. The summaries are usually between 10 and 50 words in length, which perfectly corresponds with the characteristics of document snippets returned by search engines.

b. **Varied topics** — ODP contains a vast variety of topics ranging from arts and sciences to computers, health and recreation. This makes it possible to construct mixtures of well-separated document references, such as Relational Databases + The Lord of the Rings Movies + Fishing.

c. **Varied topic sizes** — In addition to the large spectrum of themes, ODP provides topics which contain from as little as 1 up to as many as 1000 references. This enables creating mixtures of different characteristics, e.g. containing outlier topics, for example MySQL Database + Postgres Database + Orthopedic Equipment.

d. **Deep topic hierarchies** — ODP contains topic hierarchies with nesting level as deep as 13, which can be used to generate mixtures of very closely related topics.

e. **Multilingual content** — Although the majority of ODP's references are in English, it also contains a substantial amount of data written in other languages, such as German, Spanish, Polish or Japanese. This enables us to test how clustering algorithms would deal with multilingual input.

f. **Free availability** — All data catalogued in ODP can be downloaded free of charge from the ODP web site as an XML file.

Using document references drawn from ODP the following properties of clustering algorithms can be tested:

a. **Topic separation** — Tests the algorithm's ability to separate document references belonging to different topics. The following parameters determine the properties of a particular snippet mixture:

    i. **Affinity level** — The extent to which topics in the test set relate to each other — the higher the affinity level, the more related topics. Affinity level can be expressed numerically as the nesting level of the most specific ODP topic that is the parent of all snippets in the mixture. Affinity level equal to 1 means that the test set contains snippets from different ODP top categories, e.g. Computers, Health, Science and Recreation. Affinity level of e.g. 4, on the other hand, means that mixed are documents belonging to some subcategories of e.g. the Computers/Software/Databases/MySQL topic.

    ii. **Topic count** — The number of topics from which the snippets will be drawn.

    iii. **Topic size balance** — Numbers of documents in the topics included in the test set can be either balanced (i.e. equal or almost equal) or

[32] http://www.dmoz.org

unbalanced. Unbalanced mixtures of snippets may turn out more difficult to handle for some clustering algorithms.

b. **Outlier detection** — Tests whether the algorithm can highlight a topic that is clearly different from the rest of the test set. Two parameters can be defined for this type of snippet mixture:

   i. **Outlier size** — The number of snippets in the outlier topic expressed as a percentage of the average size of other topics.

   ii. **Outlier count** — The number of outlier categories present in the test set.

c. **Performance** — Tests the clustering speed for different numbers of input snippets, e.g. 50, 100, …, 500.

4.4 Evaluation methodology

To evaluate our work we have decided to adopt the merge-then-cluster method based on the Open Directory Project data. To this end, we will define three cluster measures: Cluster Contamination, Topic Coverage and Snippet Coverage.

4.4.1 Cluster Contamination measure

According to the **Cluster Contamination** measure a cluster is **pure** if it contains documents belonging to only one original partition. Noteworthy is the fact that a cluster that consists of only a subset of some original partition is still pure. The contamination measure of pure clusters is 0. If a cluster contains documents from more than one original partition, its contamination measure falls within the 0..1 range. Finally, in the worst case, a cluster consisting of an equally distributed mixture of snippets representing different partitions will be called **contaminated** and have the measure equal to 1.

More formally, we define Cluster Contamination (CC) in the following way [Osinski and Weiss, 04]. Let C be the original partitioning of documents (e.g. taken from ODP), and let K denote the set of automatically generated clusters. Further, let $H \equiv [h(c, k)]$ denote a two-dimensional contingency matrix, where $h(c, k)$ is the number of documents from the original partition c assigned to cluster k. For a perfect clustering where $C \equiv K$, H is a square matrix with exactly one non-zero element in each row and column.

We can now define the number of pairs of objects found in the same cluster k but not in the same partition:

$$a_{10}(k) = \sum_{c} \sum_{c'<c} h(c,k) \times h(c',k)$$

We also need to calculate a_{max}, which represents the worst-case scenario of document distribution in a cluster, whereby the same number of documents is taken from each partition and combined into a single cluster:

$$a_{\max}(k) = \sum_{c} \sum_{c'<c} \hat{h}(c,k) \times \hat{h}(c',k)$$

where for i=0... $|C| - 1$:

$$\hat{h}(c_i,k)=\begin{cases}\left\lfloor\frac{m}{|C|}\right\rfloor+1 & \text{if } i<(m \bmod |C|)\\ \left\lfloor\frac{m}{|C|}\right\rfloor & \text{otherwise}\end{cases}$$

$$m=\sum_c h(c,k)$$

Finally, the contamination measure of cluster k containing more than m>1 documents is defined as the ratio between a_{10} and a_{max}:

$$CC(k)=\frac{a_{10}(k)}{a_{\max}(k)}$$

It can be easily verified that the above formula has the desired properties, i.e. it is equal to 0 for clusters containing documents originating from one partition, and is equal to 1 for an equally distributed mixture of documents from different partitions.

Compared to other methods the advantage of the Cluster Contamination measure is that it does not penalise algorithms unnecessarily, e.g. for splitting big topics into a number of smaller clusters. However, this measure is designed to be calculated for individual groups, and therefore it must be accompanied by other measures which take into account the set of clusters as a whole.

4.4.2 Topic Coverage

A simple example of a situation where the Cluster Contamination measure alone fails is when for a large number of original partitions the clustering algorithm generates only one cluster containing documents from only one original partition. Clearly, from the point of view of the Cluster Contamination measure, such clustering is perfect because the generated group is pure. However, the algorithm is not penalised for omitting all other documents from the original partitioning. For this reason we have decided to introduce the **Topic Coverage** measure.

The Topic Coverage (TC) measure aims to show the algorithm's ability to cover all topics present in the input documents:

$$TC=\frac{s}{\sqrt{p\times|C|}}$$

where $|C|$ denotes the number of partitions in the original data, s is the number of partitions from C represented in K, and p is the position on the list of clusters at which the full partition coverage is achieved (i.e. every partition represented in K has at least one corresponding cluster). If there are partitions in C that have no representation in K, p is equal to $|K|$.

Topic Coverage equal to 1 means that all original partitions from C have at least one corresponding cluster in K and that the first $|C|$ top positions on the cluster list correspond to different partitions in $|C|$. Topic Coverage equal to 0 means that none of the clusters corresponds to any of the original partitions.

Clearly, Topic Coverage promotes algorithms that can find balance between different topics in the input documents, and can put exactly one cluster representing each topic in one of the top positions on the cluster list. In our opinion, such behaviour is perfectly reasonable, as it helps the users to find the documents of interest more quickly, even if they come from a small outlier topic.

4.4.3 Snippet Coverage

As clustering algorithms may omit some input snippets or put them in a group of "Other topics", it is important to define the **Snippet Coverage** (SC) measure, which is the percentage of snippets that have been assigned to at least one cluster:

$$SC = \frac{a}{t}$$

where a is the number of snippets assigned to clusters in K, and t is the total number of snippets in C. Snippet Coverage values can range from 0, when no snippets are assigned to meaningful clusters, up to 1, when all input snippets are assigned to clusters in K.

4.4.4 Additional measures

The following measures will be used in our project to complement the three base metrics:

a. **Time efficiency —** As search results clustering is meant to be performed on-line, the overall running time of algorithms as a function of the number of input snippets needs to be measured.

b. **Cluster label quality —** Evaluating the quality of cluster labels in an automatic way is very difficult (it should involve e.g. an analysis of their grammatical correctness), and therefore we have decided that this step will be based on manual investigation of good and bad cluster labels for selected data sets.

5 Results and discussion

This chapter reports on the results of our experimental evaluation of the description-comes-first approach to search results clustering. We start with a comparison of a number of variants of Lingo based on different dimensionality reduction techniques. Then we investigate the influence of the clustering quality parameter and matrix factorisation seeding strategy on the clusters produced by Lingo. Finally, we compare Lingo with two other search results clustering algorithms which do not employ dimensionality reduction techniques at all.

During the experiments we tested several properties of our algorithms, described in Section 4.3, such as the ability to separate different topics or to highlight outliers. For each of these tests we calculate cluster metrics introduced in Section 4.4. We also "manually" analyse the quality of cluster labels and compare the computational efficiency of all the algorithms.

5.1 Experimental setup

Topic separation tests were performed using 63 ODP topic mixtures with affinity level varying from 1 to 6, the number of topics varying from 2 to 8, balanced and unbalanced topic sizes. In Table 5.1 we present an example topic separation data set consisting of 6 size-balanced ODP topics with affinity level equal to 2.

Table 5.1 Example data set: topic separation

ODP CatId	Topic path	Document count
8421	Top/Recreation/Autos/Makes_and_Models/Porsche/944	22
40632	Top/Recreation/Boating/Power_Boating/Hovercraft	22
178528	Top/Recreation/Food/Drink/Cider	22
140175	Top/Recreation/Outdoors/Landsailing	22
59800	Top/Recreation/Pets/Reptiles_and_Amphibians/Snakes	22
350389	Top/Recreation/Travel/Specialty_Travel/Spas/Europe	22

Outlier detection tests were performed using 14 topic mixtures containing one and two outliers of size ranging from 10% to 100% of the average topic size in the data set. In Table 5.2 we present an example outlier detection data set containing one outlier of size 40%.

Table 5.2 Example data set: outlier detection

ODP CatId	Topic path	Document count
429194	Top/Computers/Internet/Abuse/Spam/Tracking	28
397702	Top/Computers/Internet/Protocols/SNMP/RFCs	28
791675	Top/Computers/Internet/Searching/Search_Engines/Google/Web_APIs	28
5347	Top/Computers/Internet/Chat/IRC/Channels/DALnet	29
783404	Top/Science/Chemistry/Elements/Zinc	11

Computational efficiency comparisons were carried out using 50 data sets consisting of 50 to 500 ODP snippets each. To eliminate the influence of the execution environment (JVM's just-in-time compilation, garbage collection, etc.), each batch was processed by each algorithm ten times.

Table 5.3 summarises the assumptions and conditions for our experiments, including values of the algorithms' parameters and characteristics of the execution environment. Unless noted, values presented in Table 5.3 hold for all experiments described in this chapter.

Table 5.3 *Experimental setup*

Lingo parameters	
Preprocessing — text preprocessing steps performed before the actual clustering takes place	Language Identification, Stemming, Letter Case Normalisation
k — The number of base vectors generated by the matrix factorisation algorithm.	15
Seeding Strategy — initialisation method of the factorization's *U* and *V* matrices	Random
Clustering Quality — sets the balance between clustering quality and computation time.	3 (highest)
Assignment Threshold — snippets whose cosine similarity to a cluster's label exceeds the Assignment Threshold will be placed in that cluster.	0.2
Suffix Tree Clustering parameters	
Preprocessing — text preprocessing steps performed before the actual clustering takes place	Stemming
Min Base Cluster Score — minimum allowed score of a base cluster	2
Merge Threshold	0.6
Maximum Clusters — the maximum number of clusters the algorithm is allowed to generate	15
Tolerance Rough Set Clustering parameters	
Preprocessing — text preprocessing steps performed before the actual clustering takes place	Stemming
Maximum Clusters — the maximum number of clusters the algorithm is allowed to generate	15
Membership Threshold	0.3
Co-occurrence Threshold	5
Execution environment	
Operating System	Windows XP Home
Java Virtual Machine	Sun JDK 1.4.2_04
Hardware configuration	Pentium M (Centrino) 1.3GHz, 512MB RAM

5.2 Dimensionality reduction techniques

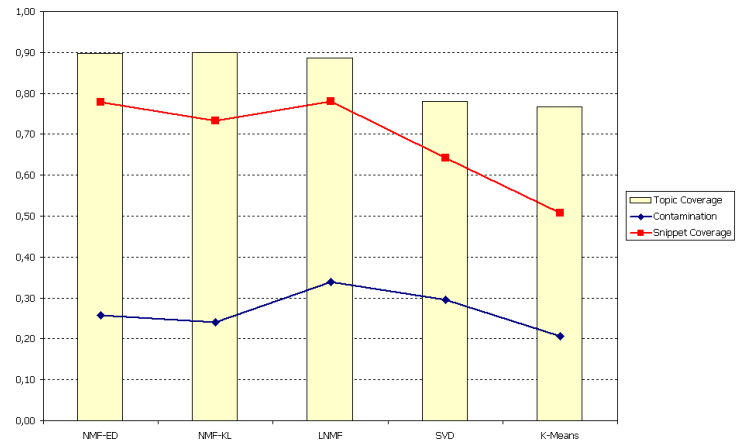
In this section we compare the performance of four dimensionality reduction techniques, Non-negative Matrix Factorisation with Euclidean Distance algorithm (NMF–ED), Non-negative Matrix Factorisation with Divergence algorithm (NMF–KL), Local Non-negative Matrix Factorisation (LNMF) and Singular Value Decomposition (SVD), as the main component of the description-comes-first clustering algorithm. Concept Decomposition (CD), a dimensionality reduction technique based on the K-Means clustering algorithm, is used as a

baseline. In the following subsections we report on the algorithms' performance with respect to topic separation, outlier detection, cluster label quality and computational efficiency.

5.2.1 Topic separation

Figure 5.1 presents average topic coverage, snippet coverage and cluster contamination for variants of Lingo employing different matrix factorisation algorithms.

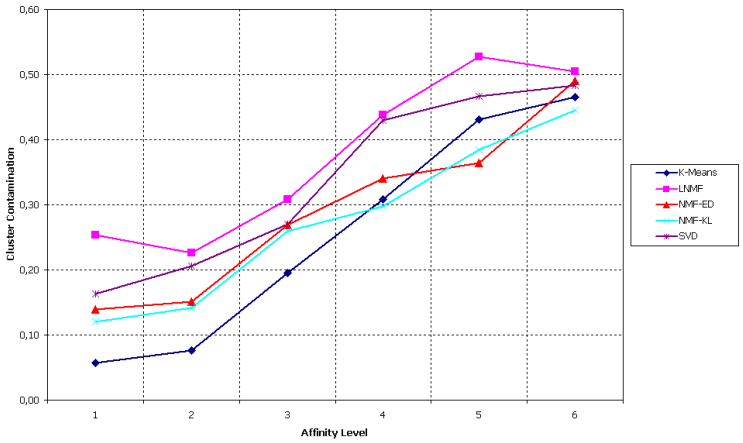
Figure 5.1 *Dimensionality reductions: aggregated cluster measures for topic separation*



Analysis Clearly, the NMF-like dimensionality reduction techniques provide significantly[33] better average topic and snippet coverage, the difference between the NMF-like algorithms themselves being statistically insignificant. Interesting is the much higher value of cluster contamination in case of the LNMF algorithm compared to the other NMF-like factorisations. We explain this phenomenon in Section 5.2.3 where we analyse cluster labels generated by all the algorithms.

In Figure 5.2 we present cluster contamination as a function of different input data affinity levels across the five matrix factorisation techniques.

Figure 5.2 *Dimensionality reductions: contamination as a function of affinity level*



Analysis As expected, the more similar topics in the mixture, the higher cluster contamination. Again, LNMF produced least pure clusters, while the K-Means method provided least contaminated groups. The latter, however, was achieved at the cost of much lower topic coverage (compare Figure 5.1).

[33] Due to the fact that our data does not follow Gaussian distribution, differences marked hereafter as statistically significant have been tested using the Mann-Whitney non-parametric two-group comparison test ([PAST, 04]) at the significance level of 0.001.

Figure 5.3 shows the dependency between input data affinity level and topic coverage across different variants of the description-comes-first clustering algorithm.

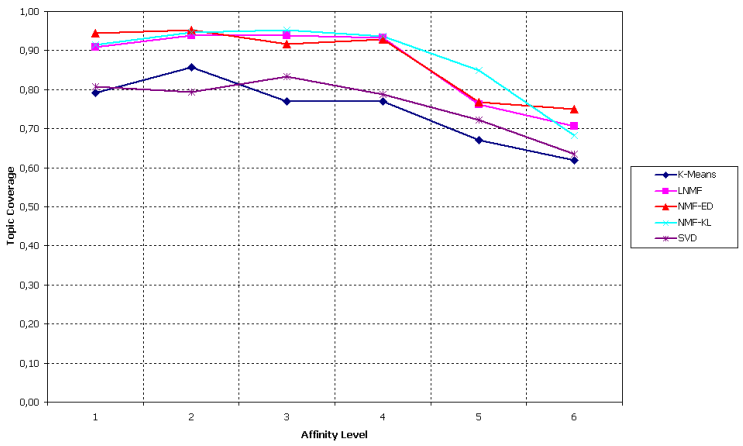


Figure 5.3 *Dimensionality reductions: topic coverage as a function of affinity level*

Analysis Again, thematically close snippet mixtures proved more difficult to cluster — for affinity levels 5 and 6 the algorithms did not manage to include all input topics in the results.

In Figure 5.4 and Figure 5.5 we show cluster contamination and topic coverage across different algorithms for balanced and unbalanced topic sizes.

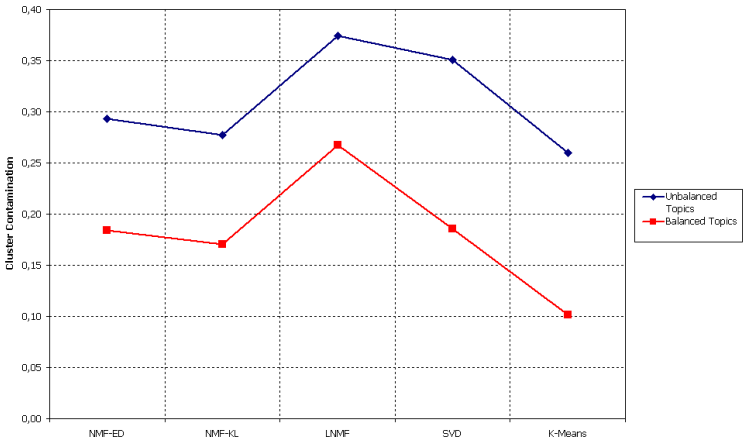


Figure 5.4 *Dimensionality reductions: contamination as a function of topic size balance*

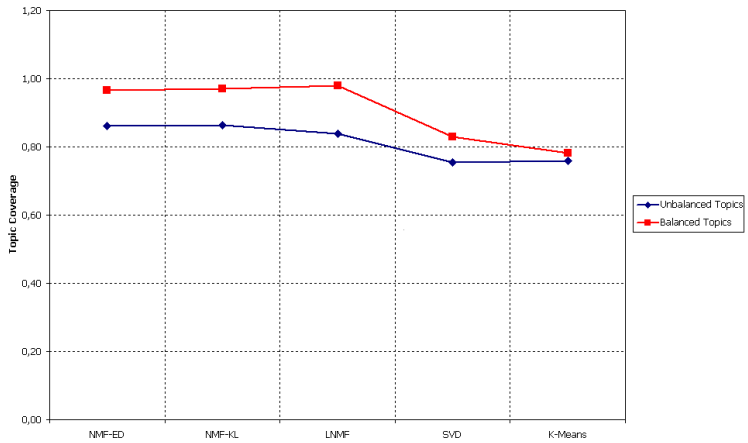


Figure 5.5 *Dimensionality reductions: topic coverage as a function of topic size balance*

Analysis Size-balanced input data sets turned out to be slightly easier to cluster compared to the unbalanced ones. Figure 5.4 confirms that LNMF produces significantly more contaminated clusters in comparison to the other NMF-like methods.

5.2.2 Outlier detection

Table 5.4 summarises the number of outliers detected by different algorithms for different outlier sizes and counts.

Table 5.4 *Dimensionality reductions: detected outliers*

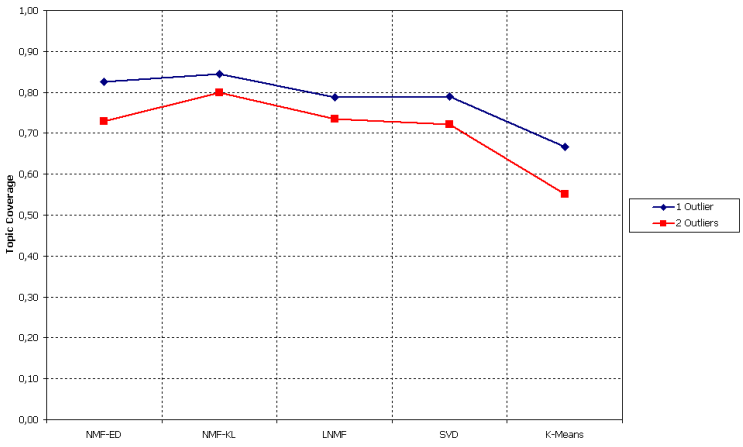
Outlier size	Detected outliers									
	NMF-ED		NMF-KL		LNMF		SVD		K-Means	
	1	2	1	2	1	2	1	2	1	2
100	1	2	1	2	1	2	1	1	0	0
50	1	2	1	1	1	1	1	1	0	0
40	1	2	1	2	1	2	0	0	0	0
30	1	1	1	1	1	1	0	1	0	0
20	1	2	1	2	1	2	1	1	0	0
15	1	1	0	1	0	1	1	2	0	0
10	1	0	1	0	1	0	1	1	0	0

Analysis Interestingly, the base line K-Means-based factorisation did not manage to reveal any of the outliers, neither in the one-outlier data set nor in the two-outlier one. This may be because K-Means tends to locate its centroids in most dense areas of the input snippet space, which is usually not where the outliers lie. All NMF-like methods performed equally well, slightly better than SVD. However, SVD was the only algorithm do discover one of the two smallest 10% outliers.

Figure 5.6 shows average topic coverage across different algorithms for outlier detection queries containing one and two outliers.

Analysis Figure 5.6 confirms the intuition that data sets containing two outliers should be more difficult to cluster than snippet mixtures with only one outlier topic. Additionally, the figure clearly shows that the K-Means-based dimensionality reduction is inferior to the rest of the algorithms in outlier detection tasks.

Figure 5.6 *Dimensionality reductions: aggregated topic coverage for outlier detection*



5.2.3 Cluster label quality

In Figure 5.7 we provide the labels of clusters produced by our algorithms for a data set containing four size-balanced ODP topics: Assembler Programming, Oncology, Collecting Stamps and Earthquakes.

Figure 5.7 *Dimensionality reductions: cluster labels*

NMF-ED	NMF-KL	LNMF	K-Means	SVD
Earthquake Prediction (21)	Earthquake Prediction (21)	Stamp Collecting (23)	Earthquake Prediction (21)	Web Sites (11)
Oncology Conference (19)	Stamp News (22)	Earthquake Prediction (21)	Programming Site (10)	Stamp Collecting (23)
Stamp Collecting (23)	Oncology Conference (19)	Oncology Conference (19)	Stamp Collecting (23)	Cancer Care (16)
Cancer Care (16)	Cancer Care (16)	Cancer Care (16)	Assembly (11)	Assembler (7)
Web Sites (11)	Assembly (11)	Assembly (11)	(Other Topics) (61)	Seismic Cataloges (5)
Assembly (11)	University (10)	Resource Site (8)		Oncology Conference (19)
Assembler Programming (8)	Resource Site (8)	New Approach (9)		Collecting (7)
University (10)	s Philatelic (5)	University (10)		Information (6)
New York (9)	Stamps (3)	Assembly Language Programming (6)		Stamps (3)
Geological Survey (3)	Exhibiting (2)	Assembler (7)		(Other Topics) (45)
Technology (2)	(Singletons) (1)	s Philatelic (5)		
(Other Topics) (23)	(Other Topics) (29)	Engineering (5)		
		World (4)		
		Geological Survey (3)		
		Network (3)		
		(Other Topics) (20)		

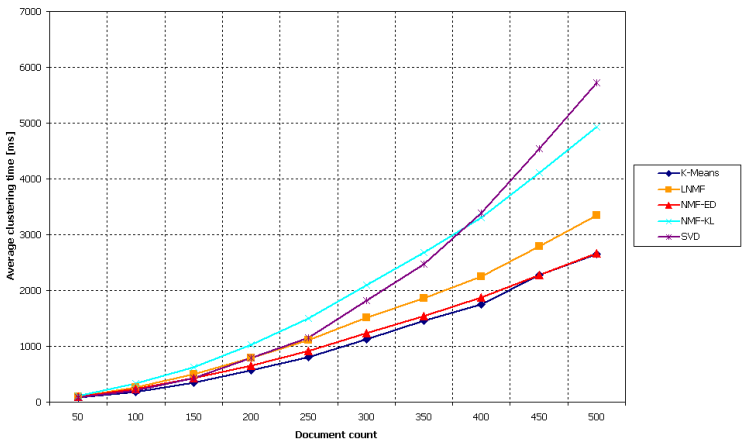
Analysis The general observation here is that the majority of cluster labels, especially those placed in top positions on the cluster lists, are well-formed readable noun phrases (e.g. "Earthquake Prediction", "Oncology Conference", "Stamp Collecting", "Assembly Language Programming"). One interesting phenomenon is that two very similar labels appeared in the NMF–ED results: "Assembly" and "Assembler Programming". The reason for this is that the English stemmer we used did not recognise the words *assembly* and *assembler* as having the same stem. Another interesting observation is the "s Philatelic" cluster label generated by the NMF–KL and LNMF algorithms. This clearly indicates a problem with the tokenization algorithm that split the possessive form of a noun (e.g. "collector's") into two tokens (e.g. "collector" and "'s").

A more careful analysis of the cluster labels created by the LNMF version of Lingo can reveal why this algorithm produces significantly more contaminated clusters (compare Figure 5.1). The key observation here is that LNMF aims to generate highly sparse and localised base vectors, i.e. having as few non-zero elements as possible (see Section 2.3.3). This results in a high number of one-word general candidate labels, such as "University", "Engineering", "World" or "Network", which in turn contribute to the high cluster contamination.

5.2.4 Computational efficiency

Figure 5.8 shows clustering time as a function of the number of input snippets for variants of Lingo employing different dimensionality reduction techniques.

Figure 5.8 *Dimensionality reductions: computational efficiency*



Analysis The most apparent observation here is that the SVD-based variant of Lingo is the least scalable. The reason for this is that in the present implementation we compute the full SVD of the term-document matrix (i.e. all min(t, d) singular vectors) and only k base vectors in case of

NMFs, which makes the latter scale much better. Thus, one of the future improvements of the former algorithm could be replacing the current SVD with its truncated variant (TSVD), which computes only the first k singular vectors.

Of the three NMF-like variants of Lingo the NMF–ED is most computationally efficient. The reason for this is that the Euclidean Distance does not need to recreate the k-rank approximation of the A matrix in every iteration, which is a fairly costly operation.

5.3 Quality level settings

In this section we show the influence of the clustering quality level setting on the results produced by the variant of Lingo which uses the NMF–ED algorithm.

5.3.1 Topic separation

Figure 5.9 presents average cluster contamination, topic and snippet coverage, for different quality level settings.

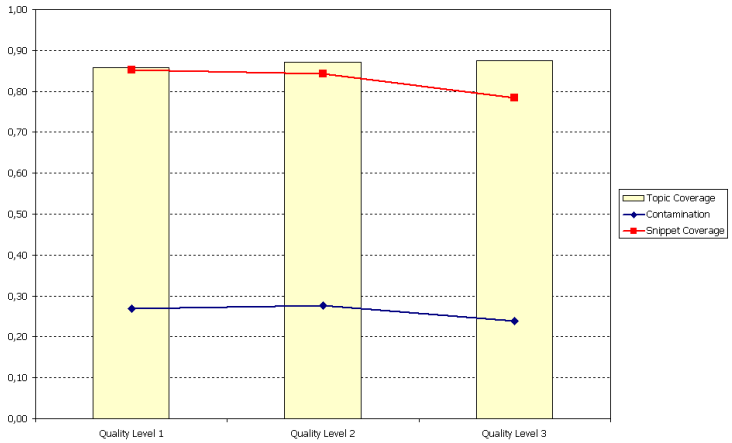


Figure 5.9 Quality level settings: aggregated cluster measures for topic separation

Analysis Interestingly, there is no statistically significant difference between topic coverage achieved at quality levels 1, 2 and 3. A closer analysis of cluster labels, see Section 5.3.3, reveals the reason for this: at quality level 1 Lingo generates worse-quality more general labels, but the correspondence between the clusters and the original topics is still maintained. It is for the same reason that for quality level 3 the average snippet coverage is significantly smaller. In this case more precise cluster labels cause more snippets to end up in the Other Topics group.

5.3.2 Outlier detection

In Table 5.5 we present the numbers of detected outliers for different quality level settings and different outlier counts.

Table 5.5 Quality level settings: detected outliers

Outlier size	Detected outliers					
	Quality level 1		Quality level 2		Quality level 3	
	1	2	1	2	1	2
100	1	1	1	2	1	2
50	1	1	1	2	1	2
40	1	2	1	2	1	2
30	1	2	1	2	1	1

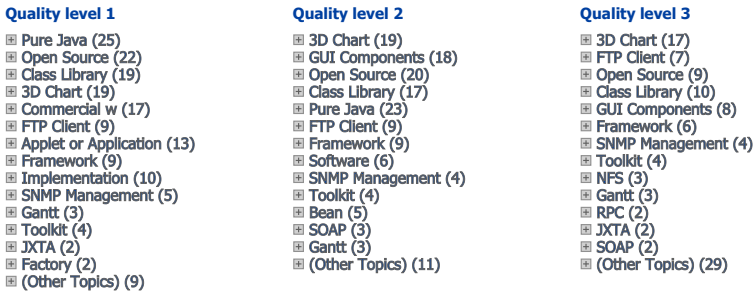
Outlier size	Detected outliers					
	Quality level 1		Quality level 2		Quality level 3	
	1	2	1	2	1	2
20	1	0	1	2	1	2
15	1	1	1	1	1	1
10	0	0	0	1	1	0

Analysis In the outlier detection task Lingo at quality level 1 performs slightly worse than at quality levels 2 and 3. The reason for this is that due to a higher term frequency cut-off at quality level 1, some of the words needed to detect the smallest outliers may not be present in the term-document matrix.

5.3.3 Cluster label quality

In Figure 5.10 we provide cluster labels generated by Lingo NMF–ED at quality levels 1, 2 and 3 for a data set containing three topics: Java Networking Class Libraries, Java GUI Class Libraries and Java Charts Class Libraries.

Figure 5.10 *Quality level settings: cluster labels*

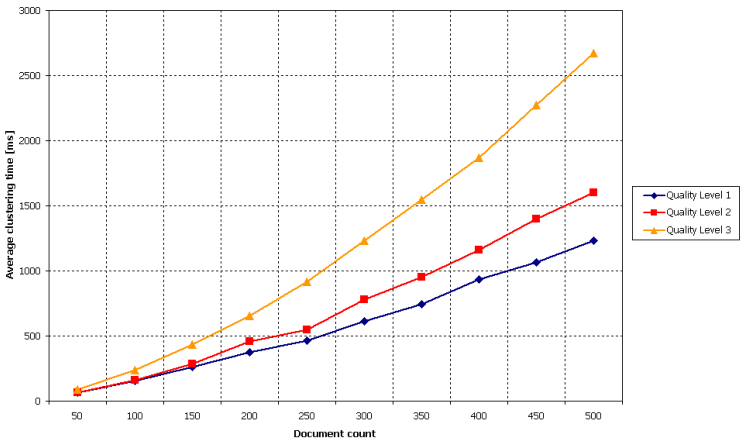


Analysis The most noticeable difference between quality levels 1 and 3 is that the former generates more general labels, such as "Pure Java", "Commercial", "Toolkit" and "Factory", while the latter more specific ones, e.g. "NFS" or "SOAP". As explained in the previous section, the reason for this is that at quality level 1 the term frequency cut-off is higher, which may eliminate some of the more specific terms from the term-document matrix.

5.3.4 Computational efficiency

In Figure 5.11 we present clustering time as a function of the number of input snippets for Lingo with different quality level settings.

Figure 5.11 *Quality level settings: computational efficiency*



Analysis Figure 5.11 shows the cost to be paid for higher clustering quality: clustering 500 snippets at quality level 3 takes more than two times as much time as at quality level 1. The latter setting can be particularly useful in high-load environments where it may be more desirable to serve more queries per second than to deliver only slightly better results at a much slower speed.

5.4 Factorisation seeding strategies

In this section we show the influence of the choice of matrix seeding strategy on the results produced by the variant of Lingo based on the NMF–ED matrix factorisation algorithm.

5.4.1 Topic separation

In Figure 5.12 aggregated cluster metrics are shown for two variants of Lingo using random and K-Means matrix factorisation seeding.

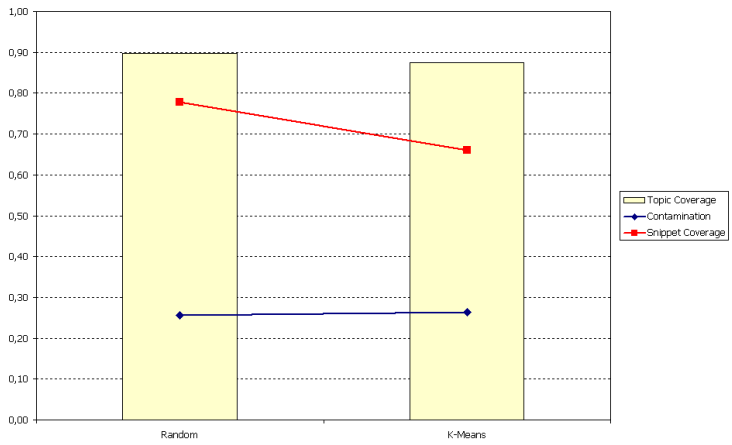


Figure 5.12 *Factorisation seeding strategies: cluster measures for topic separation*

Analysis While there is no statistically significant difference between the two variants of Lingo with respect to topic coverage, the version with K-Means-based factorisation seeding achieves significantly lower snippet coverage. We explain this phenomenon in Section 5.4.3, where we analyse cluster labels produced by the two algorithms.

5.4.2 Outlier detection

In Table 5.6 we provide the numbers of outliers the two algorithm variants were able to discover for input data containing one and two outliers.

Table 5.6 *Factorisation seeding strategies: detected outliers*

Outlier size	Detected outliers			
	Random		K-Means	
	1	**2**	**1**	**2**
100	1	2	1	0
50	1	2	0	1
40	1	2	1	1
30	1	1	0	0
20	1	2	1	0
15	1	1	1	0
10	1	0	0	0

Analysis Clearly, the randomly-initialised NMF performs better at the outlier detection task than its K-Means-initialised counterpart. To explain this result, let us refer back to Table 5.4, which shows that the K-Means-based matrix factorisation did not detect any outliers in any of the 14 test data sets. In the present case, where K-Means is used only to *initialise* the NMF, it still seems to affect the outlier detection performance to some extent.

5.4.3 Cluster label quality

Figure 5.13 shows cluster labels generated by the randomly- and K-Means-initialised variants of NMF–ED for a data set containing three ODP topics: Drugs and Medications, Pharmacology and Suicides.

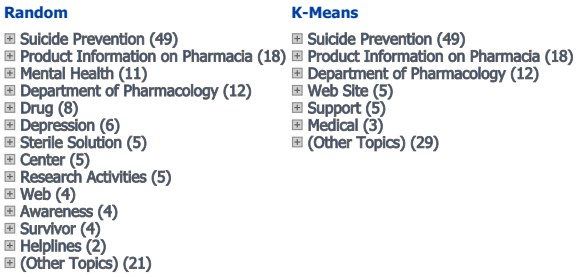


Figure 5.13 *Factorisation seeding strategies: cluster labels*

Analysis Although the K-Means-initialised NMF–ED generates much fewer labels, they still represent the major topics of the data set fairly well. However, due to the small number of clusers, the K-Means-initialised variant performs worse in outlier detection and achieves lower snippet coverage.

5.4.4 Computational efficiency

Figure 5.14 shows clustering time as a function of the number of input snippets for Lingo using different matrix seeding strategies.

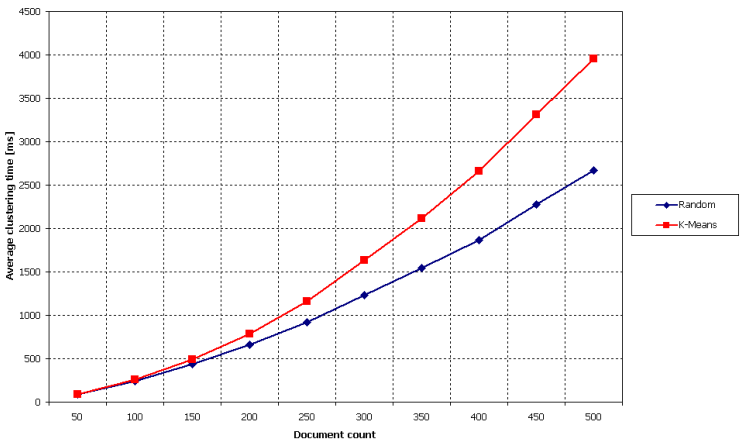


Figure 5.14 *Factorisation seeding strategies: computational efficiency*

Analysis Because of the additional K-Means clustering step at the beginning of the algorithm, the variant of Lingo based on the K-Means-initialised NMF–ED requires more time to execute and scales slightly worse.

5.5 Lingo vs. STC and TRC

5.5.1 Topic separation

Figure 5.15 shows aggregated cluster measures for three different approaches to search results clustering, Lingo (NMF–ED, Quality level 2), Suffix Tree Clustering and Tolerance Rough Set Clustering.

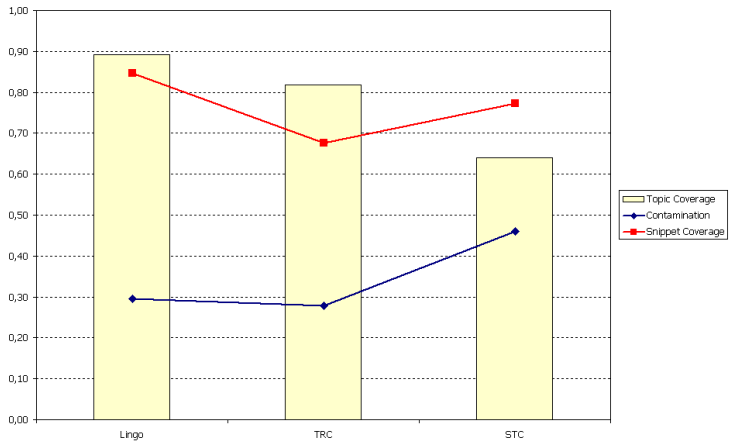


Figure 5.15 *Lingo vs. STC and TRC: aggregated cluster measures for topic separation*

Analysis Compared to TRC and STC Lingo achieves significantly better topic and snippet coverage. TRC produces slightly purer clusters, but the difference is not statistically significant. The above results prove that the description-comes-first approach to search results clustering is a viable alternative to the existing algorithms.

5.5.2 Outlier detection

In Table 5.7 we provide the number of outliers detected by each algorithm for input data containing one and two outlier topics.

Table 5.7 *Lingo vs. STC and TRC: detected outliers*

Outlier size	Detected outliers					
	Lingo		TRC		STC	
	1	**2**	**1**	**2**	**1**	**2**
100	1	2	1	1	1	0
50	1	2	0	1	0	0
40	1	2	0	2	0	0
30	1	2	0	1	0	0
20	1	2	0	0	0	0
15	1	1	0	0	0	0
10	0	1	0	0	0	0

Analysis Clearly, Lingo proves superior to the other two algorithms in the outlier detection tests for both one- and two-outlier data sets. This demonstrates the NMF's ability to discover not only the collection's major topics but also the not so well represented themes.

5.5.3 Cluster label quality

In Figure 5.16 we show cluster labels generated by Lingo, STC and TRC for a data set containing six size-balanced topics: Book Previews, Search Engines, Fitness, Do-It-Yourself, Graph Theory and Independent Filmmaking.

Figure 5.16 *Lingo vs STC and TRC: cluster labels*

Lingo NMF-ED

- Search Engines (18)
- Regular Graphs (13)
- DIY Audio (14)
- Independent Film (14)
- Book Reviews (11)
- Software Sites (19)
- Senior Health (11)
- Fitness Association (10)
- Vacuum Tube (7)
- Sample Chapters (5)
- Current and Past Projects (6)
- Color Theorem (4)
- National Institute on Aging (5)
- (Other Topics) (57)

Suffix Tree Clustering (STC)

- search, software (26)
- includes (28)
- information (20)
- site (18)
- book (16)
- resource (14)
- article (11)
- film (11)
- projects (10)
- offered (10)
- free (10)
- online (10)
- seniors (9)
- tube (9)
- audio (8)

Tollerance Rough Set (TRC)

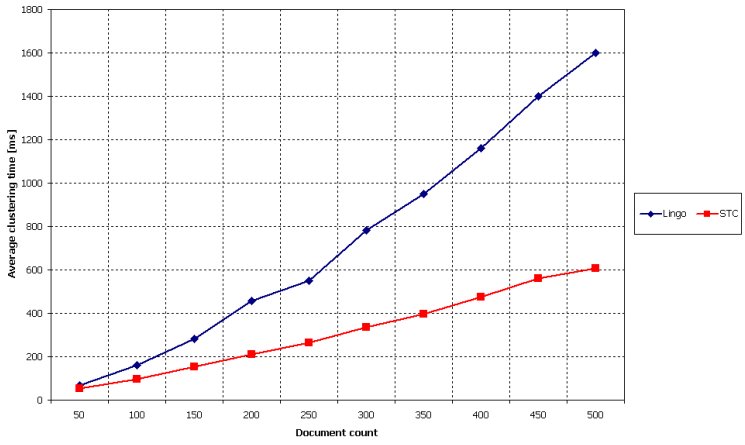
- Search (30)
- Software Search (21)
- Tube (17)
- Graph (11)
- Books (16)
- Senior (11)
- Downloadable Software Directories (3)
- Notes (1)
- Film (19)
- Other (65)

Analysis Compared to STC and TRC Lingo seems to produce labels that are slightly more specific and probably easier to interpret, compare: "Search Engines" (Lingo) vs. "Search" (TRC), "Vacuum Tube" (Lingo) vs. "Tube" (STC and TRC) or "Independent Film" (Lingo) vs. "Film" (STC and TRC). Also, for this particular data set Lingo managed to avoid generating too general or meaningless labels such as "free", "online", "site", "includes", "information" (STC) or "Notes" (TRC).

5.5.4 Computational efficiency

Figure 5.17 shows clustering time as a function of the number of input snippets for Lingo and STC[34].

Figure 5.17 *Lingo vs. STC: computational efficiency*



In Figure 5.18 we present the average clustering performance, measured in snippets per second[35], for the three compared algorithms.

[34] For the sake of clarity, due to significantly longer clustering times, TRC has been omitted in this chart

[35] The snippets per second measure is applicable only to algorithms that scale linearly with input data set size, which is not the case with TRC; we provide this value for reference only.

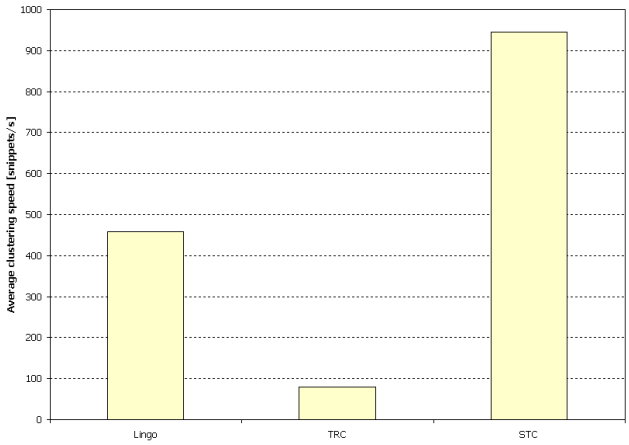


Figure 5.18 *Lingo vs. STC and TRC: aggregated computational efficiency*

Analysis Figure 5.17 shows that, similarly to STC, NMF-based Lingo scales approximately linearly with the size of the input data set. This is mainly due to the fact that the desired number of clusters was constant, irrespective of the number of input snippets. STC could cluster almost 1000 snippets per second, but this was achieved at the cost of much lower quality of the groups.

5.6 Summary

Below we summarise the most important observations we made during the whole evaluation process:

a. The NMF-like dimensionality reduction algorithms significantly outperform both SVD and K-Means factorisation with respect to topic and snippet coverage, while maintaining almost the same level of cluster contamination. The reason for this may be that, in contrast to SVD, NMF produces non-negative base vectors which can be better matched with the frequent phrases found in the input snippets.

b. Due to high sparsity of base vectors Local Non-negative Matrix Factorisation generates cluster labels that are shorter and more general compared to the other NMF methods. For this reason, contrary to our initial expectations, LNMF performs much worse with respect to average cluster contamination, and thus in the present form is not the best choice for the dimensionality reduction algorithm for Lingo.

c. Of all NMF factorisations NMF with Euclidean Distance algorithm is most computationally efficient, while still maintaining better or equal performance in topic separation and outlier detection.

d. The description-comes-first approach to search results clustering significantly outperforms both STC and TRC in topic separation and outlier detection tests.

The most general conclusion to be drawn from the results we presented in this chapter is that the best performance of the description-comes-first approach, both in terms of cluster quality and computation time, can be achieved when NMF Euclidean Distance is used to perform the dimensionality reduction step of the algorithm.

6 Conclusions and future work

In our previous work [Osinski and Weiss, 04] we have shown that it is possible to construct an efficient search results clustering algorithm based on the description-comes-first principle and Singular Value Decomposition. The aim of this project was to generalise that approach to different dimensionality reduction techniques and choose the one that would perform best in terms of clustering quality and computational efficiency.

To achieve the general goal, we implemented four dimensionality reduction techniques (two variants of NMF, LNMF and K-Means-based Concept Decomposition) and integrated them with our description-comes-first search results clustering algorithm called Lingo. Evaluation of all variants of Lingo was based on documents references drawn from the Open Directory Project database and a set of cluster quality measures we proposed specifically for the search results clustering task.

Our experiments have revealed that as part of the description-comes-first search results clustering algorithm Non-negative Matrix Factorisations perform better than SVD and the K-Means-based Concept Decomposition. Among the NMFs themselves, best results were achieved by NMF with Euclidean Distance minimisation. Finally, compared to other search results clustering algorithms we have tested, which are not based on dimensionality reduction techniques at all, Lingo performs significantly better.

All software components developed during the course of this project are part of the Open Source Carrot^2 Framework and can be downloaded[36] and modified without restrictions. We hope that this will encourage other researches to reuse and extend our algorithm. The prospect of Lingo being used as part of a major search engine at some point makes our work particularly rewarding. Overall, we fell that all aims of this project have been fully achieved.

6.1 Scientific contributions

Below we list the major scientific contributions of our project:

a. **Matrix factorisation routines in Java** — We have implemented a number of matrix factorisation algorithms in the Java programming language, including two variants of Non-negative Matrix Factorisation, Local Non-negative Matrix Factorisation and K-Means-based Concept Decomposition. Additionally, we developed a bridge between highly tuned machine-specific matrix routines provided by ATLAS, and our Java code, which can result in more than fourfold improvement in computational efficiency.

b. **ODP-based evaluation of search results clustering** — We have developed a set of Carrot^2 components that enable evaluation of search results clustering based on document references provided by the Open Directory Project. Also, we have proposed a number of cluster quality measures which, in our opinion, reasonably approximate human users' expectations about grouped search results.

[36] *http://sourceforge.net/projects/carrot2*

c. **Search results clustering benchmarking application —** To perform comprehensive tests of different variants of Lingo, STC and TRC we implemented a command-line Java application that automatically exercises each of these algorithms using data sets drawn from the Open Directory Project and saves detailed and aggregated results as XML files. We have also prepared simple XSLT stylesheets for transforming the results into HTML pages and Excel files.

6.2 Future work

In this work we have shown that the description-comes-first approach to search results clustering performs fairly well compared to other existing algorithms. Nonetheless, there are still problems, both with the algorithm and the evaluation scheme, that we did not manage to address in this project. Below we provide a list of research directions we feel are worth following.

a. **Factorisation algorithms —** All factorisation algorithms we implemented during the course of this project operate on dense matrices. In the search results clustering application, however, only about 2% of elements in the term-document matrix are non-zero. For this reason, developing dimensionality reduction routines taking advantage of the high sparsity of input matrices may yield substantial improvements in computation time and memory footprint.

   We also feel that new fast-converging algorithms for computing NMF [Liu and Yi, 02] as well as new dimensionality reduction techniques [Xu and Gong, 04] are worth experimenting with.

b. **Improvements to Lingo —** At the present stage the most important component of Lingo that is missing is a reliable method of estimating the desired number of clusters which is independent of the underlying dimensionality reduction technique. As suggested in [Li et al., 04], eigenvalue analysis of the term-document matrix may turn out useful in this area.

   Due to the limited time allocated to this project, we did not manage to look into the alternative methods of assigning snippets to clusters. One approach that may improve Lingo's clusters' purity is proximity search [Sadakane and Imai, 01].

c. **ODP-based evaluation —** At present, data sets for our ODP-based evaluation of search clustering results need to be selected manually. However, having precisely defined selection criteria, such as the desired number of topics, affinity level and topic sizes, data sets can be created fully automatically. This would dramatically increase the amount of test data and hence the reliability of the evaluation process.

Bibliography

[Berry et al., 95] Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien. *Using Linear Algebra for Intelligent Information Retrieval*. SIAM Rev., 37, pp. 537-595.

[Dhillon and Modha, 01] Inderjit S. Dhillon and Dharmendra S. Modha. *Concept Decompositions for Large Sparse Text Data Using Clustering*. Machine Learning, Vol. 42, No. 1, pp. 143-175, 2001..

[Dom, 01] Byron E. Dom. *An Information-Theoretic External Cluster-Validity Measure*. IBM Research Report RJ 10219, 2001..

[Feng et al., 02] Tao Feng, Stan Z. Li, Heung-Yeung Shum, and HongJiang Zhang. *Local Non-Negative Matrix Factorization as a Visual Representation*. Proceedings of the 2nd International Conference on Development and Learning, 2002.

[Grefenstette, 95] Gregory Grefenstette. *Comparing two language identification schemes*. 3rd International Conference on Statistical Analysis of Textual Data, Rome, 1995.

[Hearst, 98] M. A. Hearst. *The use of categories and clusters in information access interfaces*. In T. Strzalkowski (ed.), Natural Language Information Retrieval, Kluwer Academic Publishers, 1998.

[Hearst and Pedersen, 96] M. A. Hearst and J. O. Pedersen. *Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results*. Proceedings of the Nineteenth Annual International ACM SIGIR Conference, Zurich, June 1996.

[Jansen et al., 00] Bernard J. Jansen, Amanda Spink, and Tefko Saracevic. *Real life, real users, and real needs: a study and analysis of user queries on the web*. Information Processing and Management, Vol. 36, No. 2, pp. 207-227, 2000.

[Lang, 04] Nao Chi Lang. *A tolerance rough set approach to clustering web search results*. Master thesis, Faculty of Mathematics, Informatics and Mechanics, Warsaw University, 2004, *http://www.cs.put.poznan.pl/dweiss/carrot-bin/chi-lang-ngo-2004.pdf*.

[Lee and Seung, 99] Daniel D. Seung and H. Sebastian Seung. *Learning the parts of objects by non-negative matrix factorization*. Nature, Vol. 401, 1999.

[Lee and Seung, 01] Daniel D. Seung and H. Sebastian Seung. *Algorithms for non-negative matrix factorization*. Adv. Neural Info. Proc. Syst. 13, pp. 556-562, 2001.

[Li et al., 04] Tao Li, Sheng Ma, and Mitsunori Ogihara. *Document clustering via adaptive subspace iteration*. Proceedings of the 27th annual international ACM SIGIR conference on Research and development in informaion retrieval, Sheffield, UK, 2004, pp. 218-225.

[Liu and Yi, 02] By Wenguo Liu and Jianliang Yi. *Existing and New Algorithms for Non-negative Matrix Factorization*. *http://cs.utexas.edu/users/liuwg/383CProject/CS_383C_Project.htm*.

[Liu et al., 03] Tao Liu, Shengping Liu, Zheng Chen, and Wei-Ying Ma. *An Evaluation on Feature Selection for Text Clustering*. Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003..

[Macskassy et al., 98] Sofus A. Macskassy, Arunava Banerjee, Brian D. Davison, and Haym Hirsh. *Human performance on clustering Web pages: A preliminary study*. In Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98), 1998.

[Osinski and Weiss, 04] Stanislaw Osinski and Dawid Weiss. *Lingo: A Concept-Driven Algorithm for Clustering Search Results*. Accepted for IEEE Intelligent Systems, 2004.

[PAST, 04] O. Hammer, D. A. T. Harper, and P. D. Ryan. *PAST: Paleontological Statistics Software Package for Education and Data Analysis*. Palaeontologia Electronica 4(1), pp. 9, *http://palaeo-electronica.org/2001_1/past/issue1_01.htm*.

[Pedersen et al., 91] Jan Pedersen, Doug Cutting, and John Turkey. *Snippet Search: a Single Phrase Approach to Text Access*. In Proceedings of the 1991 Joint Statistical Meetings, American Statistical Association, 1991, *http://citeseer.ist.psu.edu/pedersen91snippet.html* .

[Porter, 80] M. F. Porter. *An algorithm for suffix stripping*. Program, 14 (3), pp. 130-137, 1980.

[Riboni, 02] Daniele Riboni. *Feature Selection for Web Page Classification*. *http://citeseer.nj.nec.com/554644.html*.

[Salton, 89] Gerald Salton. *Automatic Text Processing*. Addison-Wesley. 0-201-12227-8.

[Sadakane and Imai, 01] Kunihiko Sadakane and Hiroshi Imai. *Fast Algorithms for k-word Proximity Search*. Transactions on Communications/Electronics/Information and Systems, 2001.

[Selberg, 99] Eric W. Selberg. *Towards Comprehensive Web Search*. Doctoral Dissertation, University of Washington, 1999.

[Stefanowski and Weiss, 03] Jerzy Stefanowski and Dawid Weiss. *Web search results clustering in Polish: experimental evaluation of Carrot*. Advances in Soft Computing, Intelligent Information Processing and Web Mining, Proceedings of the International IIS: IIPWM´03 Conference, Zakopane, Poland, vol. 579 (XIV), 2003, pp. 209-22.

[Turk and Pentland, 91] M Turn and A. Pentland. *Eigenfaces for recognition*. In Journal of Cognitive Neuroscience, 3(1), 71–83.

[Wild, 03] Stefan Wild. *Seeding Non-Negative Matrix Factorizations with the Spherical K-Means Clustering*. MSc Dissertation, University of Colorado, 2003.

[Wroblewski, 03] Michał Wróblewski. *Hierarchical Web documents clustering algorithm based on the Vector Space Model*. Master Thesis, Poznan University of Technology, 2003.

[Xu et al, 03] Wei Xu, Xin Liu, and Yihong Gong. *Document Clustering Based On Non-negative Matrix Factorization*. Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, Toronto, Canada, 2003, pp. 267-273.

[Xu and Gong, 04] Wei Xu and Yihong Gong. *Document clustering by concept factorization*. Proceedings of the 27th annual international ACM SIGIR conference on Research and development in informaion retrieval, Sheffield, UK, 2004, pp. 202-209.

[Yang and Pedersen, 97] Yiming Yang and Jan O. Pedersen. *A Comparative Study on Feature Selection in Text Categorization*. Proceedings of ICML-97, 14th International Conference on Machine Learning. Morgan Kaufmann Publishers, San Francisco, US, 1997, pp. 412-420.

[Zamir and Etzioni, 98] Oren Zamir and Oren Etzioni. *Document Clustering: A Feasibility Demonstration*. Proceedings of the 19th International ACM SIGIR Conference on Research and Development of Information Retrieval, 1998, pp. 46-54.

[Zamir, 99] Oren E. Zamir. *Clustering Web Documents: A Phrase-Based Method for Grouping Search Engine Results*. Doctoral Dissertation, University of Washington, 1999.

[Zhang and Dong, 04] Dell Zhang and Yisheng Dong. *Semantic, Hierarchical, Online Clustering of Web Search Results*. Proceedings of the 6th Asia Pacific Web Conference (APWEB), Hangzhou, China, April 2004.

[Zhang, 02] Dell Zhang. *Towards Web Information Clustering*. PhD Dissertation, Southeast University, Nanjing, China, 2002.