Tag Cloud Control by Latent Semantic Analysis

Master's Thesis

submitted by

Angelina Velinska

IMT

supervised by

Prof. Dr. rer. nat. habil. Ralf Möller
Dipl. Ing. Sylvia Melzer
Software Systems Institute

Prof. Dr. Karl-Heinz Zimmermann
Institute of Computer Technology
Hamburg University of Technology

Dr. Michael Fritsch
CoreMedia AG
Hamburg

Hamburg University of Technology
December, 2010

Declaration

I declare that:
this work has been prepared by myself,
all literal or content based quotations are clearly pointed out,
and no other sources or aids than the declared ones have been used.

Angelina Velinska

Hamburg
December, 2010

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Information Retrieval (IR) systems become more important not only due to their use in Internet and digital libraries, but also because the majority of companies organize their activities and depend on digital documents, and information. Finding the right information brings value to the business, and failing to do so usually leads to losses.

Certain factors influence the performance of search applications. On one side is the constantly growing problem of information overload. On the other, it is the peculiarities of humans users. IR systems need to adapt to their users, and to their information need.

- **Human behavior**. Users searching for information usually submit queries composed of a few keywords only, as shown in studies [1]. The search application performs exact matching between the submitted keywords, and the document set it searches through, and often returns a long list of results. When searching with short queries, the focus of a user is unclear, and the missing information contributes to long result lists containing many unrelevant hits. Users without domain expertise are not familiar with the appropriate terminology thus not submitting the right query terms with respect to relevance or specialization. Another issue is the ambiguity of words, when words have more than one meaning. As a consequence, search results do not fit the information needs of users. When relevant documents contain words that are semantically relevant to the queries but not the same (synonyms), they will not be judged relevant.

- **Manual processing of results**. When a large number of matching documents is returned as a search result, only some of the documents can be read, due to human limitations in information processing, and time constraints (users want to find information quickly). Human users need to narrow down the search iteratively by reformulating the query, since it is unclear in which context their queried words are used, and which words could be useful to focus the search. This reformulation is known to be a frustrating and time consuming process.

- **Information need**. Search applications that implement the keyword search paradigm (or full-text search) are broadly accepted by users; however, the challenge for the next years is the development of search solutions that reflect users' context ("what the user meant" versus "what the user entered as a query"). In other words, solutions that are able to: *a*) organize search results better than in the form of long lists, *b*) adapt to a users personal skills and experience concerning the underlying document collection, and *c*) adapt to the retrieval task a user is concerned with; in other words, to adapt to the users' information need.

The factors listed above have lead to the development of techniques that assist users to effectively navigate and organize search results, with the goal to find the documents best matching their needs. *Document clustering* is a process of forming groups (clusters) of similar objects from a given set of inputs. When applied to search results, clustering organizes the results into a number of easily browsable thematic groups. It contributes to solving the problem of finding a meaningful ordering in a large amount of returned results. *Latent Semantic Analysis* is another method from IR that handles the problem of words having more than one meaning, or words ambiguity. It also filters out noise well, and reduced the number of unrelevant hits. Both techniques have been implemented in this work.

After documents have been clustered into categories according to their topics, they have to be presented to the users. In particular, the categories have to be labeled with characteristic terms for browsing. Which words from the cluster to choose as labels is a difficult problem to solve. This work presents algorithms for cluster labeling. It evaluates an interesting new algorithm, called Weighted Centroid Covering (WCC), proposed in [2] and [3]. Further, it proposes an improvement of WCC by using external semantic knowledge for definition of the cluster labels, provided by a domain-specific ontology, which was developed as a part of this work.

1.2 Information Retrieval systems

As a part of this work, we have implemented techniques from the field of IR. Therefore, what follows is an overview of the text analysis processes, common to most IR systems. Then, in the context of these processes, we will summarize the contributions presented in this thesis.

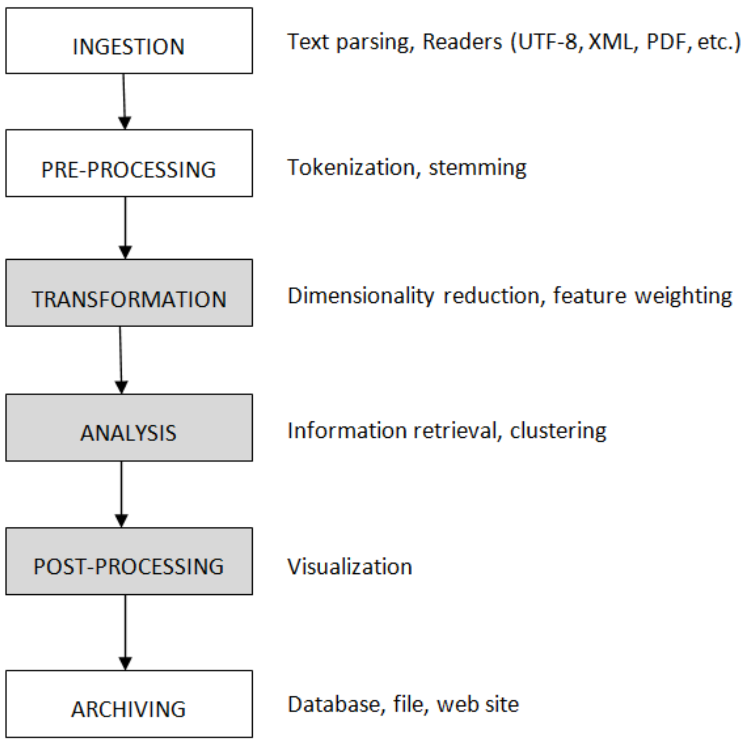

Figure 1.1: Workflow in IR systems

Most IR systems share common workflow, and follow common data processing stages. Some of the processes involved in text analysis are: lexical analysis to study word frequency distributions of words, retrieval, tagging/annotation, and visualization among the others. The workflow in IR systems usually starts with **document ingestion**, where texts from the document corpus[1], which will be analyzed, are parsed or loaded into memory. After that comes the **pre-processing** phase, responsible for text extraction, tokenization, token-filtering, text folding, etc. In this phase, documents are often represented as vectors, whose components quantify properties like how often words occur in the document. In the **transformation** phase is where a dictionary, matrix or other form of in-

[1]Throughout this work we use *document corpus* as a synonym to a document collection, in which IR tasks are performed.

dex is created from the extracted texts. In this phase all extracted terms[2] are assigned weights by applying a weight function. In Chapter 2.6 are given more details about the most common weight functions used in IR systems. Phase **analysis** can include retrieval process by querying the corpus, clustering, etc. **Visualization** phase is where concepts, query results, or summarizations extracted from the text collection are presented to users. And in the final **archiving** phase, results from the text analysis process can be saved.

The workflow in fig. 1.1 doesn't show any iterations or cycles between phases for simplicity. Iterations and repetition of certain phases of analysis, however, are common, e.g. between transformation and post-processing phases, or analysis and post-processing.

The focus of the current work is on transformation, analysis and post-processing phases of IR workflow.

1.3 Goal and scope of work

This works contributes with the following:

1. It offers an overview of the current cluster labeling algorithms, and makes an evaluation of WCC, proposed for unsupervised topic labeling of clusters in [2] and [3].

2. It proposes an improvement in WCC algorithm, performing topic identification based on external knowledge. A light-weight ontology has been developed for this purpose, in order to be used as a reference for external semantic knowledge during cluster labeling.

3. A software application for executing an IR process has been developed. It implements Latent Semantic Analysis (LSA) for information retrieval(analysis phase from fig. 1.1, and WCC for visualization of the main concepts contained in a document set in the form of a tag cloud.

[2]A term in this context denotes a word after pre-processing of the texts. A term is a single unit, e.g. a word, a phrase, a number, or an abbreviation.

1.4 Outline

This chapter motivates the presented research work, and summarizes its contributions. It also offers a general overview to the phases of text analysis, common to most IR systems. Chapter 2 gives theoretical foundations for preprocessing and transformation phases of text analysis, and reviews a specific technique for information retrieval, called Latent Semantic Analysis. Chapter 3 contains the contribution related to evaluation of WCC algorithm, and a proposal for its improvement, by using a light-weight domain ontology. It is the analytical phase of the IR process. Chapter 4 refers to the post-processing phase of text analysis, giving the visualization means for presenting the main concepts retrieved from a document set. The software contribution, developed as a part of this work, is described in Chapter 5. Finally, in Chapter 6 the thesis concludes with evaluation of results and outlook.

Chapter 2

Latent Semantic Analysis

Summary. *The chapter gives a theoretical overview of LSA in the context of its use in this work.*

2.1 Information Retrieval process

IR systems aim to satisfy user's information needs, by providing access to large collections of documents. In a search application, the IR process retrieves a set of documents which match a given query. There are three basic processes which an IR system has to support: to represent the content of the documents, to represent the user's information need, and to compare the two representations, based on a chosen similarity measure (fig. 2.1). Therefore, the first stage of constructing an IR system is to extract information about the documents content, and implement a similarity measure, based on which documents and queries can be compared. Representing the documents is usually called the *indexing process*. The comparison of the query against the document representations based on a chosen measure is called the *matching process*.

2.2 Document representation

In order to find documents which are similar to a given query, both documents and query have to be comparable, or have the same representation in the IR system. Various models have been proposed for internal representation of documents and queries. The *Boolean*, *Vector space* and *Probabilistic models* are popular IR models that find wide implementation. The Boolean model represents both documents and

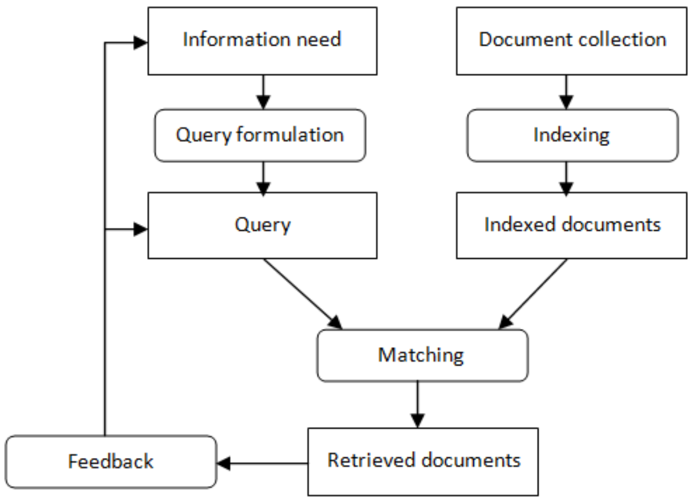

Figure 2.1: Information retrieval process, adapted from [4]. The information need of the user is formulated as query, which is transformed in the chosen model of the IR system. Documents from the collection are also represented according to the chosen model. Based on the implemented similarity measure, matching documents are identified. Finally, the retrieved results are presented to the user. If the user is not satisfied with the search results, the search query can be reformulated during feedback.

queries as a set of terms, and compares them based on boolean functions (AND, OR, NOT, etc.). The probabilistic model uses probabilistic inference for document retrieval [5]. Similarities are computed as probabilities that a document is relevant for a given query. And finally, the Vector Space Model (VSM), introduced first by Salton [6], represents both documents and queries as vectors in a multidimensional space, whose dimensions are the terms used to build an index to represent the documents (section 2.2.1 provides more details on VSM). Boolean model is easy to implement; however, when querying, users need to be familiar with boolean operators, which is a drawback to this model. Concerning the probabilistic model, one requires prior knowledge for its implementation, as it usually includes tuning of independent probabilistic parameters. VSM and the probabilistic model both have the advantage that they rank the relevant results according to a chosen weight function, but the former is easier to implement.

2.2.1 Vector Space Model

During indexing (fig. 2.1), documents are presented as data structures in memory. In VSM a document is a vector, whose elements represent properties like term frequencies, or frequency of word occurrence within the document. Before documents can be represented as vectors, they have to be tokenized, or converted from a stream of characters to a stream of words. Thus parsed, words will be used in building an index of the document collection. During tokenization one can apply filtering, i.e. removing HTML tags or other markup from text, as well as stop-words and punctuation marks removal. Stop words are such words that don't convey information specific to the text corpus, but occur frequently, such as: $a, an, and, any, some, that, this, to.$

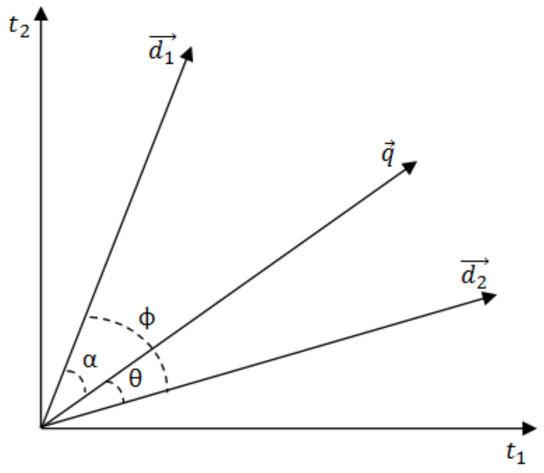

Figure 2.2: The Vector Space Model. Documents $\vec{d_1}$ and $\vec{d_2}$, and a query vector $\vec{q}$ are represented in a two-dimensional vector space, formed by terms t_1 and t_2.

A distinction has to be made between words or terms, and tokens. A term is the class which is used as a unit during parsing, and a token is each occurrence of this class. For example, in the sentence:

> *CoreMedia CMS is shipped with an installation program for interactive graphical installation and configuration of the software.*

the term *installation* is represented by two tokens.

There is no universal way to parse a text, and the parsing decisions to address depend on the application in which the text collection will be used.

After tokenization, documents are represented as vectors, where each term from the document is an element in the vector. Thus, all document vectors and terms form a multi-dimensional vector space, where terms are the dimensions, and documents - the corresponding vectors. A representation of the vector space is given in fig. 2.2, where two document vectors $\vec{d_1}$ and $\vec{d_2}$, and a query vector $\vec{q}$ are represented in a two-dimensional space.

2.2.2 Weight functions

Vectors in VSM have as elements the occurrence frequencies of words in documents. However, some documents are shorter than others, therefore one has to apply a normalization function in order to avoid representing words from longer documents as "more important" than words from shorter documents, as they would occur more often. Such normalization functions are called weight functions, and they are applied after the vector space has been constructed.

Weight functions are generally separated into two categories - local and global. They are often implemented as a pair together, because local functions measure the importance of a given word in a single document, while global functions give the importance of a word in the whole document collection. The most commonly used function pair is *term frequency* and *inverse document frequency.*

Term frequency - inverse document frequency

The simplest local weight is the term frequency $tf_{t,d}$. It assigns a weight to each term equal to the number of occurrences of the term t in a given document d. However, not all words carry the same meaning in text (therefore we remove the stop words during preprocessing, as mentioned in 2.2.1). Words that are common to all documents in the collection don't reveal much information, as compared to words which occur only in several documents. The latter are more likely to contain key information about the meaning of these documents. This is reflected by the weight function *inverse document frequency* (eq. 2.1)

$$idf_t = 1 + log\frac{N}{df_t} \tag{2.1}$$

where N is the total number of documents in the collection, and t is a specific term we are weighting. Using idf_t is a way to scale down the importance of commonly used words in text. When one combines both tf and idf, a composite weight is produced for each term in each document.

The *tf-idf* weighting function assings to a term t in a document d a weight given by

$$(tf - idf)_{t,d} = tf_{t,d} \times idf_t \tag{2.2}$$

.

As defined by Manning et al. [7], the weight assigned to term t in document d by using a combination of local and global weight function is

1. highest when t occurs many times within a small number of documents;
2. lower when the term occurs fewer times in a document, or occurs in many documents;
3. lowest when the term occurs in virtually all documents.

Log - entropy

Another popular pair of weight functions, frequently used in text analysis, is the *log-entropy* pair. The local function *log* takes the logarithm of the raw term frequency, thus normalizing the effect when large differences in term frequencies occur. In eq. 2.3 $L(t, d)$ denotes the local function that assigns a weight to term t in document d.

$$L(t, d) = \log(tf(t, d) + 1) \tag{2.3}$$

The global function *entropy* $H(d, t)$ calculates the conditional distribution that term t appeared in document d (eq. 2.4).

$$G(t) = H(d, t) = 1 + \frac{\Sigma_j p(t, d)}{\log n} \tag{2.4}$$

In eq. 2.4, $p(t, d)$ is defined by:

$$p(t, d) = \frac{tf_{t,d}}{gf_t} \tag{2.5}$$

where gf_t is the total number of times that term t occurred in the whole document collection (in all documents). The entropy measure takes into account the distribution of terms over documents.

2.2.3 Similarity measures

Once the vector space has been built, one can find the documents which are most similar to a given query. During *query formulation*(fig. 2.1), the queries are tokenized and represented as vectors, as already described in section 2.2.1. Therefore, the similarities between documents and queries can be measured based on the angles between their respective vectors (in fig. 2.2.1, these are α and θ). Using the angles between vector representations, one can define a similarity measure which is necessary for the matching process in IR systems. The standard way to computing the similarity between two documents d_1 and d_2 is to compute the *cosine similarity* of their vector representations $\overrightarrow{d_1}$ and $\overrightarrow{d_2}$ (eq. 2.6).

$$sim(d1, d2) = \frac{\overrightarrow{V}(d_1).\overrightarrow{V}(d_2)}{\left|\overrightarrow{V}(d_1)\right| . \left|\overrightarrow{V}(d_2)\right|}, \tag{2.6}$$

where the numerator represents the *dot product*[1] of the vectors, while the denominator is the product of their *Euclidean lengths*[2]. The measure from eq. 2.6 is the cosine of the angle ϕ between the two vectors $\overrightarrow{d_1}$ and $\overrightarrow{d_2}$.

Once we represent a collection of N documents as a collection of vectors, it is easy to obtain a natural view of the collection as a *term-document matrix*: this is a $m \times n$ matrix whose rows represent the m terms in the document collection, and each of whose n columns corresponds to a document. And this specific matrix grouping all documents and terms from the collection is used in *Latent Semantic Analysis*, a technique which we will discuss next.

2.3 Latent Semantic Analysis

LSA was first introduced ([8], [9]) as a technique for improving information retrieval. Vector Space Model matches the words from a user's query with the words in documents. Such IR models that depend on lexical matching have to deal with two problems: *synonymy* and *polysemy.* Due to the many meanings which the same word can have,

[1]The dot product $\overrightarrow{x}.\overrightarrow{y}$ of two vectors is defined as $\sum_{i=1}^{M} x_i y_i$.

[2]Let $\overrightarrow{d}$ is the document vector for d, with M components $\overrightarrow{d_1}...\overrightarrow{d_M}$. The Euclidean length of d is defined to be $\sqrt{\sum_{i=1}^{M} \overrightarrow{d_i^2}}$

also called polysemy, irrelevant information is retrieved when searching. And as there are different ways to describe the same concept, or synonymy, important information can be missed. LSA has been proposed to address these fundamental retrieval problems, having as a key idea dimension reduction technique, which maps documents and terms into a lower dimensional semantic space. LSA models the relationships among documents based on their constituent words, and the relationships between words based on their occurrence in documents. By using fewer dimensions than there are unique words, LSA induces similarities among words including ones that have never occurred together [10]. The basic steps in using LSA are: document representation (the same as in VSM), Singular Value Decomposition (SVD) with dimensionality reduction, and querying. Next, we give the theoretical basis for LSA, as it has been implemented as a part of this work.

We mentioned above that the first step of LSA implementation is document representation. As it is similar to the one in VSM, refer to section 2.2. After tokenizing all documents in the collection, and computing the corresponding term weights, one has to construct a term-document matrix (eq. 2.7). Having as rows the terms, and as columns the documents, its elements are the occurrences of each term in a particular document, where a_{ij} denotes the frequency with which term i occurs in document j. The size of the matrix is **m x n**, where **m** is the number of terms, and **n** is the number of documents in the text collection. Since every term doesn't appear in each document, the matrix is usually sparse.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \tag{2.7}$$

Local and global weightings are applied to increase or decrease the importance of terms within documents. After presenting the most popular weight functions in sections 2.2.2, we can write

$$a_{ij} = L(i,j) \times G(i), \tag{2.8}$$

where $L(i,j)$ is the local weighting of the term i in document j, and $G(i)$ is the global weighting for term i. The choice of weight function impacts LSA performance. In section 2.6 we give reasons for the specific implementation decisions made in this work concerning LSA.

2.4 Singular Value Decomposition

After its generation, the term-document matrix is decomposed into three matrices (eq. 2.9) by applying SVD. It is a unique decomposition of a matrix into the product of three matrices - U, V and Σ, where U and V are orthonormal matrices[3], and Σ is a diagonal matrix[4] having singular values[5] on its diagonal.

$$A = U\Sigma V^T \tag{2.9}$$

After the initial matrix A is decomposed, all but the highest k values of its product diagonal matrix Σ are set to 0. When A is recomputed again following eq. 2.9, the resulting matrix A_k represents the semantic space of the text collection. A classical vusial example can be used to presenting SVD (as given in [8]) into three product matrices. On can notice here how the dimensionality reduction of matrix Σ affects all three product matrices.

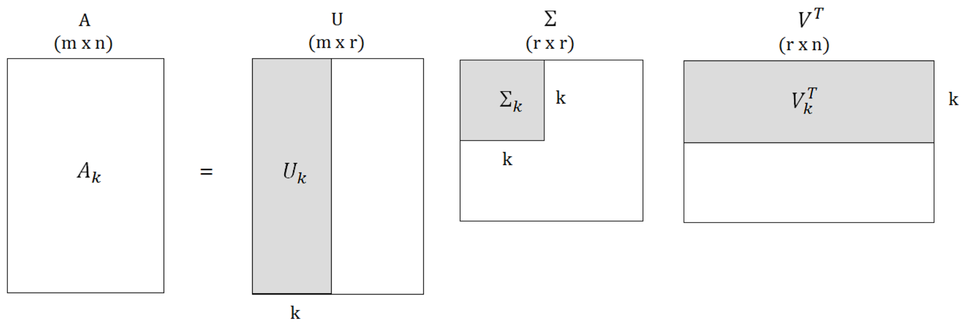

Figure 2.3: Diagram of truncated SVD

A_k - best rank-k approximation of A	m - number of terms
U - term vectors	n - number of documents
Σ - singular values	k - number of factors
V^T - document vectors	r - rank of A

[3]An orthonormal matrix is a matrix, whose columns, treated as vectors, are also orthonormal. A matrix is orthonormal, if its transpose is equal to its inverse. For more information on matrices and matrix operations, refer to [11]

[4]A diagonal matrix is a square matrix, in which the entries outside the main diagonal are all 0.

[5]For a square matrix A, the square roots of the eigenvalues of A^TA, where A^T is the conjugate transpose, are called *singular values*. Given a matrix A, a non-zero vector x is defined to be an *eigenvector* of the matrix, if it satisfies the *eigenvalue equation*: $Ax = \lambda x$ for some scalar λ. The scalar λ is called an *eigenvalue* of A corresponding to the eigenvector x.[11]

In fig. 2.3, U and V contain the term and document vectors respectively, and Σ is constructed by the singular values of A. An important property of SVD is that the singular values placed on the diagonal of Σ are in decreasing order. Hence, if all but the first k singular values are set to 0, the semantic meaning in the resulting space is preserved to some approximation k, while noise or variability in word usage, is filtered out. Noise in this case are the terms with lowest weights which carry little meaning. By using fewer dimensions k, LSA induces similarities among terms including ones that have never occurred together. Terms which occur in similar documents, for example, will be near each other in the k-dimensional vector space even if they never co-occur in the same document. This means that some documents that don't have any common words with a given query may be near it in resulting k-space.

A factor to be considered when computing SVD is the run-time complexity of the algorithm. For decomposition of very large matrices, it is $O(n^2k^3)$, where n is the number of terms in the text corpus, and k is the number of dimensions in semantic space after dimensionality reduction. Note that k is typically a small number between 50 and 350.

A more detailed description of SVD can be found in [12] and [11].

2.5 Querying the semantic space

In this work we are using LSA for IR purpose. Therefore, the final step of applying the technique is to pose queries on the constructed semantic space. A query q is a set of words which must be represented as a document in the k-dimensional space, in order to be compared to other documents. The user's query can be represented by using eq. 2.10:

$$q = q^T U_k \Sigma_k^{-1} \tag{2.10}$$

where q is the set of words in the query, multiplied by the reduced term and singular values matrices. Using the transformation in eq. (2.10), the query is "mapped" onto the reduced k-space. After the mapping, the resulting query vector can be compared to the documents in the k-space, and the results ranked by their similarity or nearness to the query. A common similarity measure used to computer similarity is the cosine between the query and the document vector 2.6. From the resulting document set, the documents closest to the query above certain threshold are returned as search results.

2.6 Factors which influence LSA performance

The effective usage of LSA is a process of sophisticated tuning. Several factors can influence the performance of the technique. These factors are pre-processing of texts (removal of stop-words, filtering, stemming), frequency matrix transformations, choice of dimensionality k, choice of similarity measure.

Dumais et al. [13] and Nakov et al. [14] have carried out research on LSA performance depending on the choice of factors such as frequency matrix transformations, similarity measures, and choice of dimension reduction parameter k. They conclude that LSA performance based on the choice of these factors depends on the particular text corpus, as well as on the purpose of LSA application. However, in the case of matrix transform, log-entropy (section 2.2.2) performs better as compared to other matrix transform function combinations, including the popular term frequency - inverse document frequency ($tf - idf$) (section 2.2.2). Therefore, we implement the former in this work. It has been further stated ([13],[15]) that with respect to similarity measures used, LSA performs optimal when cosine similarity measure (eq. 2.6) is implemented to calculate the distance between vectors in the semantic space. We have therefore used it to measure the relevance between queries and documents. And finally, the dimensionality reduction parameter k is defined empirically based on the experimentation results presented in Chapter 5.

Chapter 3

Cluster labeling

A major problem in text analysis is to determine the topics in a text collection and identify the most important or significant relationships between them. Clustering and visualizations (tag clouds) are key analysis methods in order to address this problem. In this chapter, we discuss three algorithms for cluster labeling, which are relatively new, and in Chapter 4 we will introduce tag clouds as a visualization mean in IR systems.

3.1 Clustering

Clustering is an IR method that groups objects (documents) together based on their similarities in so called clusters. Such methods are often applied in the post processing phase of IR (fig. 1.1) for visualization of retrieved results. For example, similar search results can be grouped together during presentation of results in search engines. This should help users gain a quick overview of the retrieved results. It is important to choose suitable labels of the categories defined in this way, so that they are usable to human users. The process of cluster labeling creates labels for these clustered groups of objects.

3.1.1 Clustering algorithms

A large variety of clustering algorithms exists, and they are usually classified according to the characteristic of their output structure. Jain et al. [16] make a classification based on the following properties:

Flat vs. hierarchical clustering
Flat clustering creates a flat set of clusters without relations between

clusters due to some structure. *Hierarchical clustering* on the other hand creates a hierarchy of clusters, with parents and children, in a tree-like manner.

Hard vs. soft clustering
Another important distinction can be made between *hard* and *soft* (also called fuzzy) clustering algorithms. If each document is assigned to a single cluster only, we have hard clustering. Otherwise, clustering is soft, and it assigns a document as a distribution over all clusters. This means that each document can have a fractional membership in several clusters. LSA (Chapter 2) is a good example for a soft clustering algorithm. K-means, first introduced by Lloyd [17], is a popular example for hard clustering.

Monothetic vs. polythetic clustering
Based on how documents are assigned to different clusters, clustering can be divided into *monothetic* and *polythetic* [18]. Monothetic algorithms assign a document to a cluster based on a single feature, while polythetic algorithms classify documents by overall degree of similarity/difference calculated over many properties. This makes monothetic clustering well suited for generating hierarchies and browsing search results, since a single feature, common to all documents in the cluster, describe each cluster. Users understand easily clusters generated in this way. K-means is an example of polythetic document clustering algorithm, where each cluster is described by several words of phrases.

Partial vs. complete clustering
Complete clustering assigns each document to a cluster. *Partial clustering*, on the other hand, leaves some documents unassigned.

3.1.2 Clustering algorithm??

Which method to present? K-means or hierarchical agglomerative clustering?
We implement in this work ? clustering algorithm, therefore what follows is an overview of this IR method.

3.2 Cluster labeling

There exists no commonly accepted definition of cluster labeling in literature, as it is a relatively young field of research, as compared to clustering. Assume that a categorization of a document set is determined using an unsupervised approach (e.g. clustering). To present this categorization to a user, it is convenient to label the individual categories with characteristic terms. These terms, called *category labels*, should characterize the content of the associated category with respect to the remaining categories. This implies that cluster labeling should *summarize* a category's content and that it should *discriminate* a category from the other categories. This section states desired properties of category labels and reviews three algorithms which generate such labels. It further makes a proposition for improvement of WCC (section 3.2.2) topic identification method, and evaluates it using hierarchical clustering.

When performing classification by clustering, the cluster labels are usually manually created by humans. However, this is a very expensive approach. It is sensible to find and algorithm for automatically identifying topic labels or cluster labels. Therefore, we have chosen to present three new algorithms for cluster labeling, which offer improvements in different aspects of the classical methods used until now. We further implement the Weighted Centroid Covering algorithm by Stein and zu Eissen[2], and make an evaluation of its performance. We also propose to improve WCC using external semantic knowledge. Therefore, we have developed a domain ontology for CoreMedia Content Management System (CMS) domain, which is the use case of this work.

Some clustering algorithms attempt to find a set of labels first and then build (often overlapping) clusters around the labels, thereby avoiding the problem of labeling altogether (Zamir and Etzioni 1999, Kki 2005, Osi nski andWeiss 2005).

Carpineto et al. [19] claim that there is a difference between clustering of data sets, and clustering of search results, returned from search engines: ”in search results clustering description comes first”. Thus, in contrast to the classical categorization of clustering algorithms we previously outlined, they propose a classification based on how well the clustering algorithms can generate sensible, comprehensive and compact cluster labels, and divide algorithms in the following three categories:

Data-centric algorithms
Data-centric algorithms were the first ones to be implemented for cluster

labeling. Here, clustering is done with some general clustering algorithm, and terms which are close to the cluster centroids are nominated as cluster labels. Cluster centroids are a collection of independent terms from all documents not related to each other. In order to define the terms from cluster centroid, one uses a frequency measure, such as $tf - idf$ (eq. 2.2). We present WCC (see 3.2.2) as an example for a data-centric clustering algorithm.

Description-aware algorithms
While data-centric algorithms don't consider word order, and process documents as "a bag of words", description-aware algorithms process ordered sequence of words (phrases), instead of terms, as candidate labels, and assign labels during clustering process, not after it. Using monothetic term selection, one nominates frequent phrases containing a noun head as labels, which are interpretable to human users. Clustering and labeling are closely related, and labeling influences the clustering process. Therefore, it is not possible to combine the labeling with any clustering algorithm. Suffix Tree Clustering (STC) is an example of a description-aware algorithm, introduced by Zamir and Etzioni [20]. STC builds a tree from the most frequent phrases in documents by using the data structure *suffix tree* [20]. It group documents that have common phrases under the same nodes of the tree. Thus, all documents below a given node contain the phrase at the node.

Description-centric algorithms
These algorithms operate on the principle *"description comes first"*, and in this sense are the opposite of the data-centric algorithms. The goal here is to find meaningful cluster labels. If there is not suitable label found, the cluster is useless for the users, and therefore is discarded. Description-centric algorithms are mainly applied for clustering of search results (see [19]). We introduce in this chapter two examples of such algorithms - Descriptive k-means and LINGO (both presented in Section 3.2.2).

3.2.1 Formal framework for cluster labeling algorithms

When we use an unsupervised approach to categorize a collection of documents, such as clustering, we also need to label the categories defined, in order to present them to users. The category labels should characterize the given categories - labels should summarize category content, and should discriminate a category from other categories [2]. Until now, no uniformly agreed upon formal framework exists that defines the requirements for cluster labeling algorithms. There are publications which name desired properties for cluster labels ([21], [22]), but they all give informal descriptions. Stein and Meyer Zu Eissen [2] have given their definitions for desired properties of cluster labeling algorithms as a formal framework. We base our definition below on this source.

If we have an unstructured collection of documents D, a clustering algorithm creates a categorization for this collection in the form $C = \{c_1, c_2, ..., c_k\}$, where the sets c_i are subsets of D, and their union covers D: $\cup_{c_i \in C} c_i = D$. When applying a hierarchical clustering algorithm, this implies a cluster labeling hierarchy H_C over C. Then, H_C is a tree and its nodes are the categories in C, having one node as a root. Let $c_i, c_j \in C, c_i \neq c_j$ are two categories. If c_j is a child cluster of c_i, then c_i is closer to the root of the hierarchy H_C, and we write $c_i \succ c_j$.

For an existing categorization C, we therefore need a cluster labeling $Ł = \{l_1, l_2, ..., l_k\}$.

Let $T = \cup_{d \in D} T_d$ is the term set in the given document collection D. As defined by Stein and Meyer zu Eissen [2], a cluster labeling function $\tau : c \rightarrow Ł$ assigns a cluster label to each cluster $c \in C$, where $Ł_c \subset T$.

Thus, the following properties are desired for a cluster labeling function:

1. **Unique**
   Cluster labels should be unique. The same terms should not be assigned as cluster labels to more than one cluster, or no two labels should include the same terms from two different clusters.

$$\forall_{\substack{c_i, c_j \in C, \\ c_i \neq c_j}} : \tau(C_i) \cap \tau(C_j) = 0 \tag{3.1}$$

2. **Summarizing**
   If possible, the label of a cluster c should contain at least one term t from each document $d \in c$. Terms occurring in all documents, which are part of the cluster, represent it better than terms that occur only in few documents.

$$\forall_{c \in C}, \forall_{d \in c} : \tau c \cap T_d \neq 0 \tag{3.2}$$

   where T_d is the set of terms in document d.

3. **Discriminating**
   Apart from summarizing, terms in labels should be discriminating. They should contribute to discriminate the cluster from other clusters, i.e. the same terms should be present in a considerably smaller set of documents in the other clusters. In cluster label c exists a term t whose frequency of occurrence is relatively higher in the documents of the cluster as compared to other clusters.

   $$\forall_{\substack{c_i,c_j\in C\\ c_i\neq c_j}}\exists_{t\in T_{c_i}} : \frac{tf_{c_i}(t)}{|c_i|} \ll \frac{tf_{c_i}(t)}{|c_j|} \tag{3.3}$$

   Here, T_{c_i} is the term set in category c_i, and $tf_{c_i}(t)$ is the term frequency of term t in category c_i, or the sum of $tf(t)$ in all documents in cluster $c_i : tf_{c_i}(t) = \sum_{d\in c_i} tf_d(t)$.

4. **Expressive**
   Terms forming a label of cluster c should have highest frequency of occurrence in the documents from c:

   $$\forall_{c\in C}\forall_{d\in c}\forall_{t\in T_d} : tf_d(t) \leq tf_d(t'), t' \in \tau(c) \tag{3.4}$$

   Here, $tf_d(t)$ is the term frequency of occurrence of term t in document d.

5. **Contiguous**
   This property holds for consecutive terms, for example belonging to a phrase. Such cluster labels, containing consecutive terms, are more understandable to human users.

   $$\forall_{c\in C}\forall_{\substack{t,t'\in\tau(c),\\ c_i\neq c_j}}\forall_{d\in c}\exists_{t_i,t_{i+1}\in T_d} : t_i = t \wedge t_{i+1} = t' \tag{3.5}$$

6. **Hierarchically consistent**

   $$\forall_{\substack{c_i,c_j\in C,\\ c_i\neq c_j}} : c_i \succ c_j \Rightarrow P(t_i|t_j) = 1 \wedge P(t_j|t_i) < 1, \tag{3.6}$$

   where $t_i \in \tau(c_i)$ and $t_j \in \tau(c_j)$. This property is required only when using a hierarchical clustering algorithm (e.g. STC [20]).

7. **Irredundant**
   Irredundancy complements the property *unique*. Terms which are synonyms should be avoided in cluster labels.

   $$\forall_{c\in C}, \forall_{\substack{t,t'\in\tau c,\\ t\neq t'}} : \text{ t and t' are not synonyms} \tag{3.7}$$

The properties stated above describe ideal conditions and can only be approximated in the real world. One needs external knowledge (e.g. an ontology), in order to fulfill properties *hierarchical consistency* and *irredundancy*.

3.2.2 Algorithms for cluster labeling

We will present algorithms for cluster labeling, which are relatively new.

Weighted Centroid Covering

Weighted Centroid Covering was first introduced by Stein and Meyer zu Eissen [2]. It is a data-centric algorithm, in which labels are generated from sets of frequently occurring terms.

The algorithms consists of three stages. Having a clustering as input to the algorithm:

1. for all words in the input clusters, it saves all k most frequent occurrences of each word with its term frequency, and the cluster in which it occurs, to a data structure (say a vector).

2. it sorts the tuples (k, tf, word, cluster) in a descending order, based on the frequency tf;

3. it assigns l number of terms to each cluster as labels. Therefore, in the end each cluster has a label of l terms, assigned to it in a Round-Robin-like manner.

   A pseudo code of the algorithm is given as algorithm 3.1. Additionally, the major steps can be seen graphically in fig. **??**.

Descriptive k-means

Descriptive k-means - 2007

Lingo

Lingo - 2004

Algorithm 3.1 Weighted Centroid Covering for cluster labeling

```
Input: $C$ - clustering
  $l$ - number of terms per label
  $k$ - maximum occurrence of the same term in different labels
Output: $\tau$ - labeling function
  Ł $= 0$
  foreach $c$ in $C$ do
  $\tau(c) = 0$;
  end foreach
  foreach $t$ in $T$ do
  for $i = 1$ to $k$ do
    compute $c = k(t, i)$ from $C$;
    add tuple $\langle t, tf_c(t)\rangle$ to Ł;
  end for
  end foreach
  sort Ł according to descending term frequencies;
  for $labelcount = 1$ to $l$ do
    $assigned = 0$;
    $j = 1$;
    while $assigned < |C|$ and $j \leq |Ł|$ do
      let $t_j = \langle t, tf_c(t)\rangle$ be $j^{th}$ tuple of Ł;
      if $|\tau(c)| < labelcount$ then
        $\tau(c) = \tau(c) \cup \{t\}$;
        delete $t_j$ from Ł;
        $assigned = assiged + 1$;
      end if
      $j = j + 1$;
    end while
  end for
  foreach $c$ in $C$ do
  do sort $\tau(c)$
  end foreach
  return $\tau$;
```

3.3 Cluster labeling using external knowledge

Chapter 4

Tag Clouds

Tagging, which is one of the defining characteristics of Web 2.0 services, allows users to collectively classify and find information. Some websites include tag clouds as a way to visualize tags in a folksonomy.(4)

An empirical analysis of the complex dynamics of tagging systems, published in 2007,(5) has shown that consensus around stable distributions and shared vocabularies does emerge, even in the absence of a central controlled vocabulary.For content to be searchable, it should be categorised and grouped. This is possible only if the content is tagged like keywords in a journal article.

(4) Lamere, Paul (June 2008). "Social Tagging And Music Information Retrieval". Journal of New Music Research 37 (2): 101114. http://www.informaworld.com/sm
tent=a906001732.
(5) Harry Halpin, Valentin Robu, Hana Shepherd The Complex Dynamics of Collaborative Tagging, Proceedings of the 16th International Conference on the World Wide Web (WWW'07), Banff, Canada, pp. 211-220, ACM Press, 2007.

We have found so far no other application implementing LSA in order to present an overview of the main topics found in a collection of unstructured texts, based on user queries. The TagCloud Summarizer is in this sense new.

There exist, however, search engines, which utilize categorizing of search results, such as Yippy(former Clusty, Vivisimo). It utilizes search, classification, and Social web (Web 2.0).

4.1 existing implementations

http://cloud.yippy.com/ - visualizes topics based on search queries. Created at Vivisimo company, also creator of one of the most successful meta search engines, offering classification of search results, Clusty (Vivisimo).

SenseBot Summarizer summarizes search results in the form of a tag cloud.

Google on the other hand offers "Wonder wheel" option, in order to display search results

TagCloud Summarizer is a tool that users can use to instantly visualize a topic using the familiar tag cloud display. Users can create a cloud based on a query.

TagCloud Summarizer generates a cloud using the user's search results for the topic they enter. Using the Summarizer to generate the cloud also ensures that it is always up-to-date because topics/main concepts are generated in real-time, based on the user's query.

4.2 use

Use the TagCloud Summarizer for online web-pages, search systems, or personal web-sites.

Summary. *This chapter presents an overview of tag-clouds used as a method for representing text content.*

Tag Clouds are popular applications used for vaious purposes: as a navigation mechanism, as indicators of activity within social media experiences, for visualization in texts and textual data, for annotation of documents 4.1. The importance or weight of words in the tag cloud are shown with size of font and/or color. The tag clouds are hyperlinks leading to a collection of items associated with the tag.
A version of tag cloud is called text cloud. It is used as a visual display that conveys the broad themes that emerge from textual analysis. There are three types of tag clouds depeding on their purpose and use. The first type contains a tag represeting the frequency of each term. The second type is a global tag cloud whose tags has frequencies aggreggated over

all items and users. The third type of tag cloud contains categories, and its tags' size indicates the number of subcategories.

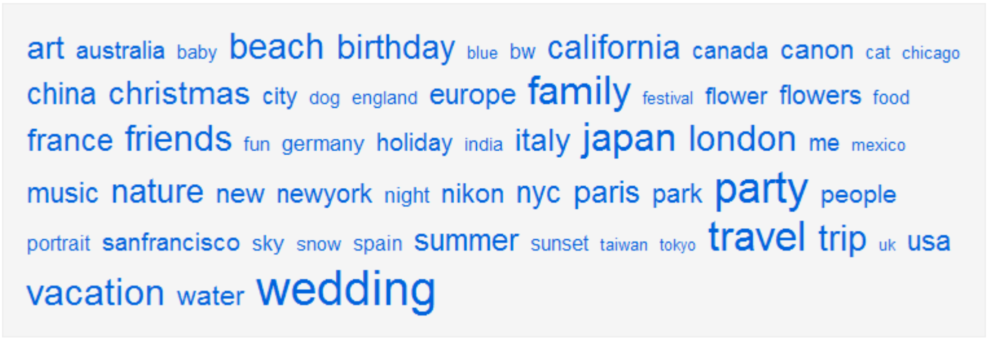
art australia baby beach birthday blue bw california canada canon cat chicago
china christmas city dog england europe family festival flower flowers food
france friends fun germany holiday india italy japan london me mexico
music nature new newyork night nikon nyc paris park party people
portrait sanfrancisco sky snow spain summer sunset taiwan tokyo travel trip uk usa
vacation water wedding

Figure 4.1: Tag Cloud

4.3 1

Related work
Opinion Crawl[1] - web sentiment analysis application. It generates a concept cloud from daily scanned blogs, web site articles.

SenseBot Search Results Summarizer is a plugin for Mozilla Firefox browser that generates a tag cloud of the main concepts returned as search results from Google.

Search Cloudlet[2] is another Firefox Addon that inserts a related tagcloud into Google interface. Working behind the scenes, Search Cloudlet injects a tag cloud of related words in to both Google and Yahoo search results pages. Then you can use the tag links to quickly and easily filter and refine your searches.
LinkSensor SenseBotSummarizer All three are based on SenseBot - a semantic search engine. Made available from Semantic Firefox Extensions [3]

[1]http://www.opinioncrawl.com/
[2]https://addons.mozilla.org/en-US/firefox/addon/9943/
[3]http://www.semanticengines.com/plugins.htm

4.4 Brainstorming

What is a tag cloud? Graphical representation of a collection of tags. Tag clouds visualize word frequency in a given text.

Tag clouds may be used as a topic summary.

There are three main types of tag cloud applications used in social software.

1. frequency of items / tags
2. number of items to which a tag has been applied
3. tags are categorization method for content items

The following tag clouds were evaluated in order to select the solution that is most applicable for Tag Cloud Summarizer project.

- Yippi Cloud Creator (former Clusty) [4]
- TagsTreeMaps[5]
- OpenCloud[6]

4.5 Tag clouds construction

According to [**?**]:
Erstellung von Schlagwortwolken Tagging beschreibt die Aktivitt, eine Ressource mit einem oder mehreren Schlsselworten oder Schlsselphrasen zu assoziieren. Ein Tag ist als Etikett oder Notiz zu verstehen, um zu einem spteren Zeitpunkt Dinge leichter wieder finden zu knnen. Tags dienen somit der Organisation von Ressourcen. Groer Beliebtheit erfreuen sich unter anderem kollaborative Tagging-Dienste wie Flickr (Bilder), del.icio.us (Internetseiten) oder Facebook (soziales Tagging). Beim kollaborativen Tagging annotieren Nutzer selbst die entsprechenden Ressourcen. Der auf diese Weise implizit entstehende sogenannte tagspace soll anschlieend effizient durchsuchbar sein. Jedoch fhrt das manuelle Annotieren von Ressourcen zu Problemen, wie Golder u. Huberman (2006) aufzeigen. Synonyme und Polyseme verursachen dabei Probleme, die

[4]`http://cloud.yippy.com/`
[5]`http://tagstreemaps.sourceforge.net/TagsTreeMaps.html`
[6]`http://opencloud.sourceforge.net/`

bereits in Kapitel 2.1.1 angesprochen wurden. So schliet eine Bildersuche mittels des Schlsselworts Auto keine Bilder mit ein, die nur mit dem Synonym Pkw assoziiert sind. Ein weiteres Problem ist, das dieselben Ressourcen von verschiedenen Nutzern unterschiedlich annotiert werden. Fgt ein Nutzer als Schlsselwort einem Bild Auto als Schlsselwort hinzu, so wird ein anderer durch Zuweisung von Volkswagen Golf Mk5 (A5/Typ 1K, 2003-2009) deutlich konkreter. Um Inkonsistenzen bei der Zuweisung von Tags zu vermeiden und dadurch die Retrievalqualitt von Suchmaschinen zu steigern, wird automatisches Tagging eingesetzt. Dieses dient vor allem beim Annotieren von neuen Weblog-Nachrichten dazu, Empfehlungen fr passende Schlsselworte einem Nutzer vorzuschlagen. Brooks u. Montanez (2006) annotieren automatisch neue Nachrichten in Weblogs. Sie verwenden dazu die drei hufigsten Terme einer Nachricht, gewichtet nach dem tf-idf - Schema. Nachteil ist, dass Schlsselworte somit auf Wort-Unigrammen und dem der Nachricht zugrundeliegenden Vokabular beschrnkt sind. Mishne (2006) ermitteln zu neuen Nachrichten in Weblogs zunchst hnliche Nachrichten, die von anderen Nutzern zuvor verfasst wurden. Anschlieend werden einem Nutzer die dort verwendeten Schlsselworte als Tags fr eine neue Nachricht vorgeschlagen. Lee u. Chun (2007) setzen dagegen ein berwachtes Lernverfahren ein ein neuronales Netz. Dieses wird auf einer Menge von Blog-Nachrichten trainiert, denen Tags zugewiesen sind. Anschlieend knnen Empfehlungen fr Tags fr neue Nachrichten gegeben werden. Grundlage zur Erfassung von Tags sind Verfahren der Schlsselwortbestimmung. Dabei werden hufige, durch tf-idf gewichteteWort-Bigramme in den Nachrichten ermittelt. Ob die ermittelten Bigramme auch allgemein hufig verwendet werden, wird anschlieend mittels WordNet evaluiert. Neben der automatischen Empfehlung von Schlsselworten wird ebenso versucht, aus dem tagspace eine Hierarchie abzuleiten (Brooks u. Montanez, 2006; Wu u. a., 2006). Solche Hierarchien dienen der Suchoptimierung. Da im Allgemeinen bei der Erstellung von Schlagwortwolken auf vorhandene Schlsselworte zurckgegriffen wird, finden sich bis auf die Erstellung von Taxonomien aus Schlagworten keine Gemeinsamkeiten mit Anstzen des Cluster-Labelings.

Chapter 5

Implementation and evaluation

Summary. *This chapter reports the implemented solution for the given thesis problem, presents experimental results, and discusses its advantages and disadvantages.*

It is a challenge by itself to come up with a sensible evaluation set for an IR implementation. ... Define here precision, recall, measures , how to measure LSA performance with diff. k, clusters, cluster labelling.

The document collection consists of guides and manuals about CoreMedia CMS 5.2.

5.1 Measures for evaluation

Evaluation in IR is based on measures. We outline several measures used with commonly used in IR systems to measure performance.

5.1.1 Evaluation of LSA

Presision:
Recall:

5.1.2 Evaluation of algorithms for cluster labeling

F-measure:

5.2 LSA implementation

LSA was applied to a collection of 11818 words in 4000 documents, all of which describe CoreMedia CMS 5.2. The algorithm took 63552 ms for preprocessing and indexing of the whole document collection.

For the implementation of LSA this work uses the open LSA library which is part of Semantic Spaces Project[23]. It is developed at the Natural Language Processing Group at the University of California at Berkley (UCLA)[1].

The real difficulty of LSA is to find out how many dimensions to remove - the problem of dimensionality.

TODO:test if inluding only terms that occur in more than one document improved LSA performance with respect to generating precise tag clouds.

5.2.1 Latest development in the field of LSA

LSAView is a tool for visual exploration of latent semantic modelling, developed at Sandia National Laboratories [24].

at the end- improvements of lsa with the basics explained.

5.3 Tag Cloud implementation

The implemented open source library used for tag cloud generation is called Opencloud[2], and is provided by Marco Cavallo.

5.4 Tools used

Airhead Research[3] project was used as a semantic spaces Package which provides a java-based implementation of LSA.

Apache Lucene[4] was used as an indexing and search library.

[1]`http://code.google.com/p/airhead-research/`
[2]`http://opencloud.mcavallo.org/`
[3]`http://code.google.com/p/airhead-research/`
[4]`http://lucene.apache.org/java/3_0_2/`

5.5 Advantages and drawbacks

why am i using lsa instead of lda for example?

1. PLSA - characteristics, advantages, disadvantages
2. LDA - characteristics, advantages, disadvantages

5.6 Experimental evaluation

Example for presentation and evaluation of results, IR system: [25]

- Data set
- Experimental setup
- Analysis of data matrices
- True/false positive rates
- Feature reduction

5.6.1 Document set

Document set		
Category Id	Category	Document #
1	session, connection	5
2	server	5
3	publication, workflow	5

The document set used for evaluation consists of 3 categories, and 15 documents - each category has 5 documents, and the documents have different length. Table 5.1 gives an overview of the constructed data sets. The document preprocessing involves parsing and stop word removal according to standard stop word lists.

Doc. Id	Cat. Id	Document
1	1	The session that is created while the connection is opened is also known as the connection session.
2	1	The sessions of the connected clients will be closed and no more content changes are possible.
3	1	The previous code fragment shows how a second session is created from an existing connection.
4	1	Multiple sessions show their greatest potential in trusted applications which receive help in restricting user views while ma
5	1	Having opened connection, all actions are executed on behalf of the single user whose credentials where provided when log
6	2	The state of the Master Live Server must be younger than the state of the Slave Live Server.
7	2	The rewrite module checks the availability of the requested file and, in the negative case, passes the request on to the Act
8	2	The directory layout of the Active Delivery Server has changed as well as the format of the configuration file.
9	2	The CoreMedia Content Server is a central component that manages the content repository and the user authorization.
10	2	The CoreMedia Content Management Server is the production system used to create and administrate content.
11	3	If the database does not allow to take online-backups ensure that all publications are finished and that the last publicatio
12	3	The third task in the workflow aims to check if the change set is empty. Then, no publication is necessary and the workflo
13	3	This element is used to define which information should be shown in the columns of the workflow list at the left side of th
14	3	Publication will be executed when finishing the task after all resources in the change set have been approved.
15	3	The CoreMedia Workflow installation comes with four predefined workflows which cover the publication of resources.

Table 5.1: Document set used for evaluation

5.6.2 Cluster labeling evaluation

To evaluate the quality of WCC, Popescul and Ungars method, and the standard keyword extraction algorithm RSP, we downloaded 150 documents from the digital library Citeseer, each of which containing author-defined keywords.
To evaluate the WCC algorithm, we prepared a document set with author-defined keywords. The evaluation idea is to cluster the documents, label the resulting clustering, compute f1-f4, and measure to which extent the identified topic labels appear in the keyword list of at least one document of the associated cluster. To measure the extent, standard precision and recall values can be chosen. Note, however, that the precision value is more important in our scenario since usually only a few words (about one to five) are presented to a user.

The clustering was done on the results from queries used for evaluation of LSA (??).
The clustering was done on the prepared data set 5.1, used for evaluation of the IR methods implemented in this work.

Figure 3.15 shows precision and recall curves for WCC depending on the number of extracted words per label. The values are averaged over 10 retries of the experiment, each time redrawing a new set of documents. The precision curve shows that one to five cluster labels can be identified at very high precision rates. (??check with my results?)

5.6.3 Results

5.7 LSA evaluation from IR book

Dumais (1993) and Dumais (1995) conducted experiments with LSI on TREC documents and tasks, using the commonly-used Lanczos algorithm to compute the SVD. At the time of their work in the early 1990s, the LSI computation on tens of thousands of documents took approximately a day on one machine. On these experiments, they achieved precision at or above that of the median TREC participant. On about 20system was the top scorer, and reportedly slightly better on average than standard vector spaces for LSI at about 350 dimensions. Here are some conclusions on LSI first suggested by their work, and subsequently verified by many other experiments.
The computational cost of the SVD is significant; at the time of this writing, we know of no successful experiment with over one million docu-

ments. This has been the biggest obstacle to the widespread adoption to LSI. One approach to this obstacle is to build the LSI representation on a randomly sampled subset of the documents in the collection, following which the remaining documents are folded in as detailed with Equation (18.21).
As we reduce k, recall tends to increase, as expected.
Most surprisingly, a value of k in the low hundreds can actually increase precision on some query benchmarks. This appears to suggest that for a suitable value of k, LSI addresses some of the challenges of synonymy.
LSIworks best in applicationswhere there is little overlap between queries and documents.
The experiments also documented some modes where LSI failed to match the effectiveness of more traditional indexes and score computations. Most notably (and perhaps obviously), LSI shares two basic drawbacks of vector space retrieval: there is no good way of expressing negations (find documents that contain german but not shepherd), and noway of enforcing Boolean conditions.
LSI can be viewed as soft clustering by interpreting SOFT CLUSTER-ING each dimension of the reduced space as a cluster and the value that a document has on that dimension as its fractional membership in that cluster.

Acronyms

CMS Content Management System.

IR Information Retrieval.

LSA Latent Semantic Analysis.

STC Suffix Tree Clustering.

SVD Singular Value Decomposition.

VSM Vector Space Model.

WCC Weighted Centroid Covering.

Appendix A

Ontology

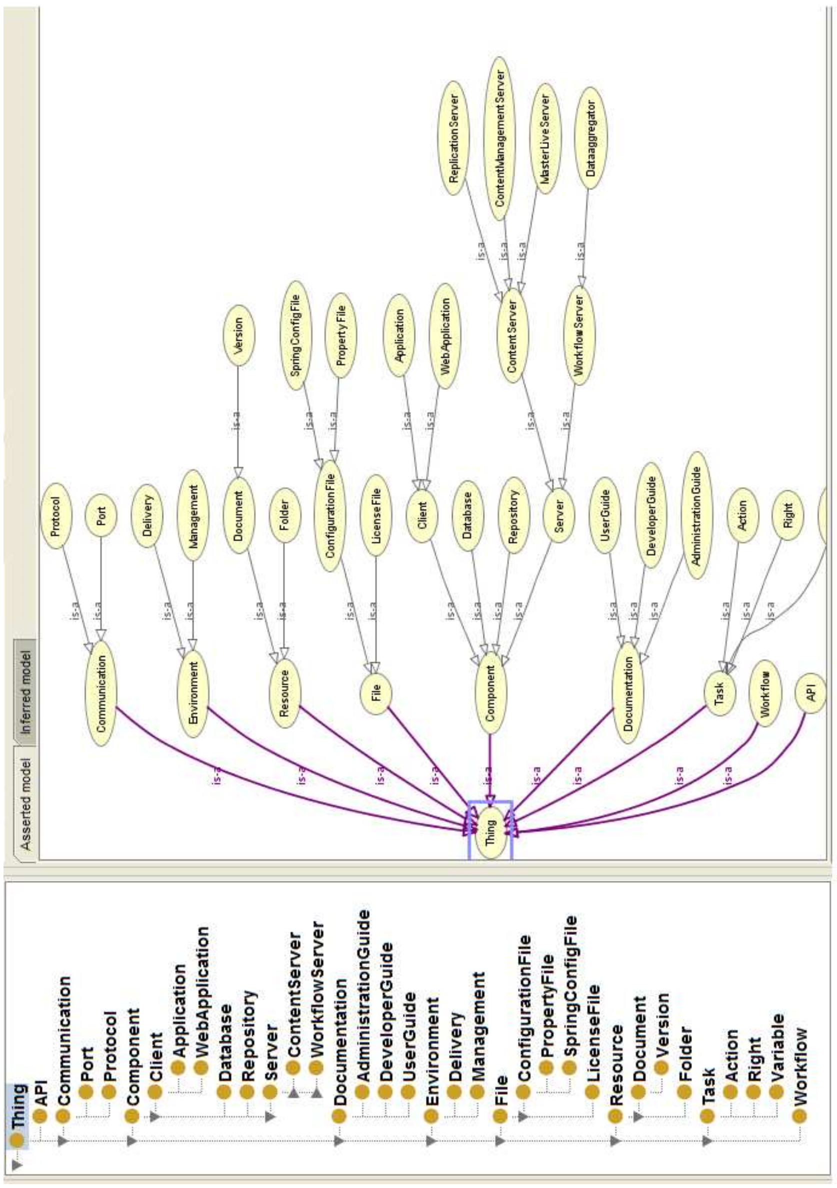

Figure A.1: Upper level ontology for CoreMedia CMS domain

Appendix B

Tag Cloud Summarizer

TODO: include source code here

Bibliography

[1] A. Spink, J. Bateman, and B. J. Jansen, "Users' searching behavior on the excite web search engine," in *WebNet*, 1998.

[2] B. Stein and S. M. Z. Eissen, "Topic identification: Framework and application," in *Proc of International Conference on Knowledge Management (I-KNOW)*, 2004.

[3] B. Stein and S. M. zu Eissen, "Topic identification," *KI*, vol. 21, no. 3, pp. 16–22, 2007.

[4] D. Hiemstra, "Information retrieval models," in *Information Retrieval: Searching in the 21st Century* (A. Göker and J. Davies, eds.), UK: Wiley, 2009.

[5] S. Robertson, "The probability ranking principle in IR," vol. 33, pp. 294–304, 1977.

[6] G. Salton, "Automatic text processing: The transformation, analysis, and retrieval of information by computer," AddisonWesley, 1989.

[7] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval.* Cambridge, UK: Cambridge University Press, 2008.

[8] S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman, "Using Latent Semantic Analysis to improve access to textual information," in *Sigchi Conference on Human Factors in Computing Systems*, pp. 281–285, ACM, 1988.

[9] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by Latent Semantic Analysis," *Journal of the American Society for Information Science*, vol. 41, pp. 391–407, 1990.

[10] S. Dumais, "LSA and Information Retrieval: Getting Back to Basics," pp. 293–321, 2007.

[11] G. H. Golub and C. F. Van Loan, *Matrix computations (3rd ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[12] M. W. Berry, S. Dumais, G. O'Brien, M. W. Berry, S. T. Dumais, and Gavin, "Using linear algebra for intelligent information retrieval," *SIAM Review*, vol. 37, pp. 573–595, 1995.

[13] S. T. Dumais, "Improving the retrieval of information from external sources," *Behavior Research Methods, Instruments, & Computers*, vol. 23, pp. 229–236, 1991.

[14] P. Nakov, A. Popova, and P. Mateev, "Weight functions impact on LSA performance," in *EuroConference RANLP'2001 (Recent Advances in NLP*, pp. 187–193, 2001.

[15] P. Nakov, "Getting better results with Latent Semantic Indexing," in *In Proceedings of the Students Prenetations at ESSLLI-2000*, pp. 156–166, 2000.

[16] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," 1999.

[17] S. Lloyd, "Least squares quantization in pcm," *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 129–137, 1982.

[18] K. Kummamuru, R. Lotlikar, S. Roy, K. Singal, and R. Krishnapuram, "A hierarchical monothetic document clustering algorithm for summarization and browsing search results," in *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pp. 658–665, ACM, 2004.

[19] C. Carpineto, S. Osinski, G. Romano, and D. Weiss, "A survey of web clustering engines," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–38, 2009.

[20] O. Zamir and O. Etzioni, "Grouper: A dynamic clustering interface to web search results," pp. 1361–1374, 1999.

[21] R. Krishnapuram and K. Kummamuru, "Automatic taxonomy generation: issues and possibilities," in *Proceedings of the 10th international fuzzy systems association World Congress conference on Fuzzy sets and systems*, IFSA'03, pp. 52–63, Springer-Verlag, 2003.

[22] D. Weiss, *Descriptive Clustering as a Method for Exploring Text Collections*. PhD thesis, Poznań University of Technology, Poznań, Poland, 2006.

[23] D. Jurgens and K. Stevens, “The S-Space package: an open source package for word space models,” in *ACL ’10: Proceedings of the ACL 2010 System Demonstrations*, (Morristown, NJ, USA), pp. 30–35, Association for Computational Linguistics, 2010.

[24] P. Crossno, D. Dunlavy, and T. Shead, “Lsaview: A tool for visual exploration of Latent Semantic Modeling,” in *IEEE Symposium on Visual Analytics Science and Technology*, 2009.

[25] G. Wilfried, J. Andreas, and R. Neumayer, *Spam filtering based on latent semantic indexing*, pp. 165–183. Springer, 2008.