Contents

Preface

This thesis is submitted in partial fullfillment of the requirements for the degree of Master of Science in Engineering at the Technical University of Denmark (DTU), Lyngby, Denmark.

Author is Sune Birch (s971739).

Thesis supervisor is Prof. Lars Kai Hansen and co-supervisor is Rasmus Elsborg Madsen, Dept. of Informatics and Mathematical Modelling (IMM), DTU.

Thesis work was conducted at Dept. of Informatics and Mathematical Modelling (IMM) from Mar. 2003 - Oct. 2003.

Resume

Intelligente computer-baserede systemer, som behandler tekstdokumenter er et aktivt forskningsområde. Søgemaskiner er højst sandsynligt det bedst kendte eksempel på, at anvendelsen af statistisk analyse af ustruktureret tekst kan være særdeles effektiv. Men der findes et væld af andre mulige anvendelser, hvor statistisk tekst modellering vil være velegnet. Et andet eksempel er matching af jobansøgere og job. Arbejdsgivere bruger mange resourcer på at finde den rette ansøger, og jobansøgere bruger tilsvarende mange resourcer i søgningen efter det rette job. Intelligente systemer, som automatisk og pålideligt kunne matche ansøgere og job ville spare mange kræfter. Dette eksamensprojekt er blevet inspireret af jobmatching-problemet og lignende matching-problemer. Målet er at finde frem til velegnede statistiske modeller for matching-problemer og at udføre en grundig analyse af disse modeller - både teoretisk og eksperimentielt.

Søgemaskiner hører til indenfor *Information Retrieval*. Vores studier af dette område viser, at der har været en del fokus på modeller, som beskæftiger sig med linkstrukturer. Med linkstrukturer menes dokumenter, som er indbyrdes forbundet med links. I forbindelse med matching-problemer kan det vise sig meget interessant, fordi et link minder meget om et match. Probalistic Latent Semantic Indexing (PLSI) er en model som kan udvides til, udover ord, også at repræsentere links. I dette eksamensprojekt bliver det vist, at PLSI er i stand til at "forstå" semantik i tekstdokumenter. Udvides PLSI dernæst med links, er modellen også til en vis grad i stand til at forudsige hvilke links et dokument burde have udfra dets tekstindhold. Dette virker bedst, hvis linkstrukturen er rimelig tæt. Intentionen er, at den insigt i statistisk tekstmodellering, som formidles i denne rapport vil vise sig nyttig under design af intelligente computersystemer for matching-problemer.

Keywords: Statistisk tekst modellering, Probalistic Latent Semantic Indexing, PLSI, PLINK, Supervised PLSI, Job matching.

Abstract

Automatic systems dealing with text is an active area of research. Search engines are probably the most successful and well-known area where statistical analysis of unstructured text documents have proven very useful. But there are a vast amount of potential applications, which are suitable for statistical text modelling. One such application is the matching of job applicants with job offers. Large amounts of human resources are used today on the search for future employees and - from the applicant's perspective - on the search for the right job. Successful automatic systems would be very welcome in this area. This master thesis has been motivated by the job matching problem and related matching problems. The objective is to find suitable models for matching problems and to perform a thorough analysis of these models from a theoretical and from an experimental perspective.

Search engines belong to the area of Information Retrieval. In Information Retrieval there has been a lot of focus on models dealing with link structures - that is, documents interconnected by links - and links are interesting, because a link is very similar to a match. The Probalistic Latent Semantic Indexing (PLSI) model is a model, which can be extended to incorporate link information. This thesis shows that PLSI is a model capable of capturing semantics in text documents. Furthermore when extended with link information it is capable of predicting links to some degree in environments where link information is not too sparse. It is the hope that the insight and the capabilities of the models presented here will turn out useful when building automatic systems for matching problems or related problems.

Keywords: Statistical text modelling, Probalistic Latent Semantic Indexing, PLSI, PLINK, Supervised PLSI, Job matching.

Chapter 1

Introduction

1.1 Statistical modelling and Matching problems

This work focuses on investigating statistical models for text documents. The reason for looking at these models is that currently there are a number of problems dealing with the understanding of text documents that we don't have very successful automatic systems for. On the other hand the human mind is very successful in the area of text understanding. That is, there is a large gap between the capabilities of computer systems and the capabilities of humans in this field. In this work we are not specifically aiming at modelling the human mind in order to build models that work like the mind. Rather we seek for models that can compete with the human mind in this area. A side effect might be to gain a better understanding of the human mind, but that is not our specific purpose.

My interest in statistical models for text documents is motivated by a problem, which seems to repeat itself in different areas. The problem can be characterized as a matching problem. Some examples of matching problems are job matching and dating profile matching. If we consider the job matching problem, then on the one hand we have a number of job offers and on the other hand we have a number of applicants. The job matching problem is to find the best match between job offers and applicants. Seen from the applicants perspective we want to find the job offers that provide the best match. And from the employer's perspective we want to find the applicants that provide the best match to the job offer. Matching dating profiles are concerned with a number of people described by profiles who are looking for other people that match their profile. What has to be fulfilled for a good match is not obvious, but it is not a trivial task, since e.g. some people might be looking for other people that do not necessarily resemble themselves.

Another area where good matching solutions can be applied is in Peer-to-peer (P2P) networks used for sharing all sorts of things such as music,

movies, software, electronic books and so on. P2P networks can have millions of users and searching through all users in order to find what one is looking for is not possible. The approach of a web search engine would be to crawl the items of all users and build a central index of all items, which can then be looked up at search time. But in P2P networks no central server exists so no central index exists (since Napster [24]) in which to look up the items of everyone at the time of the search. If we assume that other users with similar interests are more likely to have what the user is searching for then matching users with similar interests would be very helpful. The profile of users is the items that they possess, and this profile can be matched with other users.

Text modelling seems to be a good way of approaching the matching problem, because the objects that we have looked at so far, which need to be matched can be described as text "documents". Even the profile of peers in P2P networks can be described by a textual list of items. Therefore we shall be working under the assumption that statistical models for text documents will be useful for the matching problem.

The introductory chapter will give an overview on the application of statistical text modelling. We will be interested in the kind of problems that statistical text modelling has been applied to in order to draw on experience from problems similar to the matching problem. After that we will dig deeper into some specific statistical models in order to get a deeper understanding of how they are constructed, what the ideas behind them are and how well they work. We shall be analyzing the statistical models from a theoretical perspective as well as from an experimental perspective. The models will be implemented and used on experiments to get a deeper understanding of how they work.

1.2 Applications of statistical text modelling

1.2.1 Information Retrieval

The objective of this section is to give an introductory survey to the area of information retrieval. The survey will focus on practical applications. That is, we want to know what the tasks are that information retrieval tries to solve. Furthermore we want to know how successful the application of information retrieval is. What are the results of experiments in this field so far? And how well can we expect a state-of-the-art information retrieval system to perform?

A very good source of information on information retrieval is the web site of the TExt Retrieval Conference (TREC) [4]. The conference is held once a year and research groups from all over the world present their newest results in this field. But TREC is not like most conferences where you can pick any task that you want to solve and be accepted to the conference. In order to be able to compare the results of different research groups the tasks are well defined and the data sets that have to be used for a specific task are specified and provided by TREC. The tasks are divided into different areas of information retrieval called tracks. Thus each track has a couple of tasks and data sets, which all the participants have to work and present their results on. The methods of evaluation are also specified in advance, such that it is possible to compare the results. TREC is a good starting point for us in the information retrieval domain, because it can provide the inside into what tasks are being worked on and what the state-of-the-art results are. The tracks at the TREC conference are used in the following survey providing an overview on common information retrieval tasks.

Web searching

The web track at TREC deals with World Wide Web (WWW) search tasks. This is basically the playground for new search engine proposals. The document collection is, in principle, the entire WWW and the tasks deal with matching a given query from the user with the documents on the WWW. Two very important tasks exist for search engines:

Topic distillation Given a query find the best set of home pages describing the topic indicated by the query.

Named page finding Given a query find the particular page that the user is looking for.

The method of evaluation for the topic distillation task at TREC is precision score, which is defined as

$$\text{Precision} = \frac{N_{\text{relevant documents retrieved}}}{N_{\text{documents retrieved}}} \tag{1.1}$$

In the web track the precision score is computed for the first 10 documents retrieved by the searching algorithm. Thus it is judged manually how many out of the first 10 documents retrieved are relevant to the query. This score is called P@10 [6]. The reasoning behind this score is probably that search engine users tend not to look at more than the first 10 results, so the most relevant documents have to be among those 10.

For the TREC-2002 conference the best result obtained for the topic distillation task was a P@10 score of 0.2510 [6]. That is, around 25% of the 10 first documents were relevant to the query. The data set used for the web track was a crawl of the ".gov" domain retrieving 1.25 million web pages. For most topics the number of relevant documents was a number between 0 and 50, and the maximum number of relevant documents was under 200. The point is that out of 1.25 million documents to find those 0 - 50 relevant documents and return 2-3 out of 10 is not an easy task.

The objective of the named page finding task is to return the one particular page relevant to the query as one of the first results returned. If the pages returned are ranked such that the first page returned has rank 1, the second page returned has rank 2 and so on, then the measure of performance is the Mean Reciprocal Rank (MRR). The best result obtained for the TREC-2002 conference was an MRR score of 0.719 [6], which means that the one relevant page was returned as number $\frac{1}{0.719} = 1.39$ on average.

The statistical models participating in the web track differed in what features they extracted from the document collection, but the most common features used were document structure, anchor text and link structure. Document structure refers to the structure of web pages imposed by HTML. E.g. this facilitates the possibility of weighting different types of contents, such as body text, title, meta text and different fonts, in the document differently. For instance large font text might be considered more important than normal text. E.g. that approach is taken by the Google search engine [3]. Anchor text refers to the text, which is displayed by a page when linking to other pages. This is the text that one has to click on in a browser when surfing to a different page. In Google the anchor text is added to the document, which is linked to, because often the anchor text is a good description of the target page. Link structure of the document collection is also used in several models, which is interesting, because models capable of representing link information might be suitable models for matching problems. We will discuss that further in section 1.3.

Question Answering

The question answering track at TREC focuses on information retrieval as opposed to just document retrieval. The user seeks an answer to a specific question and the system returns a single answer as opposed to a ranked list of documents like in the web track. The tasks used in the TREC-2002

question answering task were the following [29]:

Find single answer to a question The system is supposed to return exactly one answer to each one of 500 test questions. One example of a test question is "What river in the US is known as the Big Muddy?". Furthermore the answers are ranked according to the confidence that the system has in its answers such that the answer for which the system is most confident is ranked first and so on.

List task The list task required systems to assemble an answer from information located in multiple documents. One example is "Name 22 cities that have a subway system". In this case the system is required to return 22 answers.

A lot of issues come into play when moving from returning documents to returning actual answers. For instance it is more difficult to judge whether an answer is correct, because there are multiple ways to put an answer as opposed to a particular document. In TREC-2002 the requirements for returning exact answers became more strict than at previous TREC conferences. E.g. redundant information not answering the particular question was not accepted in a correct answer requiring the system to understand the answer content to a higher degree than earlier. But basically a correct answer is a personal judgement, and different people might judge it differently. The TREC conference had humans assessing the exactness of the answers.

Two main approaches were used when building models for the question answering task. One was a data-driven approach and one was a logic representation approach. The system achieving the best results for both tasks mentioned here was a logic representation system, but the runner up used a data-driven approach [29]. Statistical models use a data-driven approach. On the other hand Logic representation systems e.g. use first order logic to represent knowledge and to make inferences when trying to answer the given question. Logic representation systems are related to the field of expert systems. These are out of the scope of this work, since we are concerned with statistical models. But it is worth noting that in the question answering task a subtle difference in the question such as using the word "not" changes the meaning dramatically. In a logic representation approach it might be easier to distinguish between such cases than in a statistical oriented system.

Typically all systems classified an incoming question according to predefined question types as a first step, and further processing was then question type specific. E.g. the classes of questions used in [26] corresponded to the types of answers expected, which were *abbreviation, entity, description, human, location* and *numeric*.

A number of performance measures were used for the single answer to question task. The "confidence weighted score" for instance, rewards the systems' abilities to give good rankings of the answers according to confidence. Here we shall only report the rate of correct answers score. The best

system achieved answering 83% out of the 500 questions correctly. The level of difficulty of the questions is of course a crucial factor for the success of the participating systems. For instance the TREC-2002 question answering track contained no definition questions such as "Who is Duke Ellington?", which are among the most difficult questions but also the answers are among the most difficult ones to evaluate, and that's why they were left out [29]. Still the 83% success rate gives us an idea about the performance of the state-of-the-art question answering systems.

For the list task the measure of performance was the average accuracy. E.g. out of the 22 answers returned to the question "Name 22 cities that have a subway system", how many answers were correct? The average accuracy is then taken over a total of 25 test questions. The best result obtained according to [29] was an average accuracy of 65%. That is, the best system found 65% of the list items on average.

Filtering

Filtering is an information retrieval task in which the information need of the user is stable. The system knows the user profile and monitors a stream of information filtering out the information relevant to a user. That is, for every piece of information the system has to make a binary decision whether the information is relevant to the user.

TREC has a filtering track consisting of three tasks [25]. They are concerned with documents as the type of information. The tasks are

Adaptive filtering The task is designed such that the user arrives with a few relevant documents for each of a number of topics. The system builds a preliminary user profile based on those documents. New documents arrive sequentially and the system must decide for every document whether or not to retrieve it. If it decides to do so, it receives relevance feedback from the user on whether or not the document was relevant to the user. Based on the relevance feedback the system can adapt the user profile and thereby enhance performance on future documents. In the TREC task the relevance feedback is simulated by releasing a pre-existing relevance judgement for the document.

Batch filtering and Routing In these tasks a larger training set is made available to the system including relevance judgements. The system is then evaluated on a test set for which it has to retrieve relevant documents. The difference between batch filtering and routing is that the results returned by the routing system are ranked.

For the adaptive filtering task a number of measures of performance were used. Here we shall focus only on one called **F-beta**, because the interpretation of the results are the easiest. F-beta is a trade-off between precision and recall, where precision was defined in equation 1.1 and recall is defined as

$$\text{Recall} = \frac{N_{\text{relevant documents retrieved}}}{N_{\text{relevant documents}}} \tag{1.2}$$

The TREC-2002 weighted precision higher than recall, such that the measure becomes

$$\frac{1.25 \cdot N_{\text{relevant documents retrieved}}}{N_{\text{retrieved documents}} + 0.25 \cdot N_{\text{relevant documents}}} \tag{1.3}$$

As an example, retrieving 5 relevant documents out of 10 relevant documents in total and 10 documents retrieved gives an F-beta score of 50%. The F-beta score is calculated as a mean over the 50 topics used in the task in TREC-2002. The best results for the adaptive filtering and batch filtering tasks can be seen in the following table:

Task	Adaptive Filtering	Batch Filtering
Best Mean F-beta score	0.45	0.55

The routing task used a different measure of performance, and the results are not reported here, because the task is very similar to the batch filtering task. Summarizing the results we can conclude that a state-of-the-art filtering system can achieve about 50% of "precision". Where precision is loosely defined to mean the F-beta score.

A lot of different models are used for the filtering task [25]. One feature, which seems to be common to systems is the use of classifiers. The system has to make a binary decision on whether to retrieve a document or not, which seems to require a binary classifier. A lot of classifiers exist, though, and a lot of different classifiers were used for the filtering task. Support Vector Machines (SVM) seem to be a popular choice of model for the systems participating in the filtering task. For an introduction to SVM see e.g. [22]

Other Information retrieval tasks

Other Information retrieval tasks besides web searching, question answering and filtering of course exist. An overview of the remaining tracks at the TREC conference can provide some insight.

Information Retrieval from multimedia sources such as video is also considered at TREC. The **Video track** is a new track at TREC. In 2002 it featured three tasks: Shot boundary determination, high-level feature extraction (e.g. indoor/outdoor) and searching for a shot with a given content.

In domains where huge amount of data exists finding the right information is particularly difficult and automatic Information Retrieval is very welcome. Such a domain is genomics - huge databases exist with articles describing new functions found concerning specific gene sequences. The **Genome track** at TREC considers retrieving information from these data

bases - e.g. finding articles about a gene that describes aspects of its function.

The **Cross-Language Track** at TREC is concerned with the ability of information systems to retrieve relevant documents independent of the language in which the document is written. Most research in Information Retrieval deals with the english language, but we have to recognize that information retrieved in english is not very useful if the user doesn't understand english. Therefore the Cross-Language Track is relevant.

A few other tracks exist at TREC but the tasks are quite similar to the ones that we have already presented.

1.2.2 Link models for ranking relevant pages

In this section we shall explore models for link structure more deeply. We encountered such models in the web searching task in section 1.2.1. The link models presented here are motivated by the application of statistical models to ranking relevant pages returned from a web search. Furthermore these models are interesting, because we want to find models, which are capable of predicting links. More on that in section 1.3.

Using link structure has shown to be useful in ranking relevant pages in a web search. In the web searching track at the TREC-2002 conference two different tasks were considered: Topic distillation and Named page finding. The number of relevant pages relevant to a query is the fundamental difference between the two tasks. For topic distillation we expect to find a large number of relevant pages on the topic of interest. On the other hand only a few pages are likely to match a named page finding query. For topic distillation the difficult task is to rank the relevant pages and return the most relevant first.

Bibliometrics

The basic idea behind ranking relevant pages according to link structure is that pages linked to by many other pages are more important and should therefore be given a high rank. This idea was inspired by *Bibliometrics* - the study of written documents and their citation structure [18]. We shall look a bit into how the ranking of papers are done by the use of citations. The simplest method of ranking is based on a pure counting of the number of citations received by a paper. A very important extension is then based on the observation that not all citations are equally important [18]. A paper is more influential if it is cited by other *influential* papers. We introduce the following:

$w_j \equiv$ Influence weight of a paper j.
$A \equiv$ Citation structure matrix with entries A_{ij} denoting the fraction of citations from paper i that go to paper j.

Now the influence, w_j, of paper j can be calculated by not only summing all the citations received by paper j but by weighting each citation with the influence, w_i, of the citing paper i. That is

$$w_j = \sum_i A_{ij} w_i, \quad \forall i \sum_j A_{ij} = 1 \tag{1.4}$$

If we define the vector of influence weights, w, this can be written in matrix notation as

$$A^T w = w \tag{1.5}$$

where the rows of A sums to 1. This formulation implies that w is the eigenvector of A^T with eigenvalue 1.

PageRank

The popular search engine Google has basically implemented the idea from Bibliometrics of influence of journal papers in the WWW domain [3]. The rank of the relevant pages is determined by the influence of the page calculated by a calculation very similar to equation 1.4. Google has named their algorithm **PageRank**. We redefine the entries of the citation matrix, A, so that

$A' \equiv$ Link matrix where A'_{ij} is equal to 1 if there is a link from i to j and 0 otherwise.

In our notation, PageRank can be written as [23]

$$\text{PageRank}(j) = \frac{(1-\beta)}{N} + \beta \sum_i \frac{A'_{ij}}{|F_i|} \text{ PageRank}(i), \quad 0 \leq \beta \leq 1 \tag{1.6}$$

where β is a small constant close to 0, N is the total number of documents and F_i is the set of outlinks from page i. PageRank(j) is seen to be very similar to the weight, w_j, introduced in equation 1.4.

Hubs and Authorities

The ranking of web page results inspired by Bibliometrics and introduced by Google has been very successful. But the difference between the citation structure of journal papers and the link structure of web pages has motivated new suggestions for ways to rank relevant web pages. The concepts of "hubs" and "authorities" introduced by Kleinberg [18] has been very influential. The motivation is that in the domain of scientific papers influential papers can easily cite each other, but in the WWW domain some of the most influential or "authoritative" pages often don't want to cite each other. E.g. two different search engine sites such as **www.altavista.com**

and **www.google.com**, which would definitely be considered authoritative pages, are not very likely to link to each other for commercial reasons. Therefore Kleinberg introduces the concept of a "hub", which acts as an intermediate stop. A prototype hub is a site that has collected a lot of good links for a specific topic, but doesn't provide any information on the topic itself. Thus good hubs point to many good authorities. On the other hand good authorities are pointed to by many good hubs. Thus we have a reinforcing relationship between hubs and authorities. With each page, p, we associate an authority weight, $x(p)$ and a hub weight, $y(p)$. We introduce the following:
$x \equiv$ Vector of authority weights.
$y \equiv$ Vector of hub weights.
The reinforcing relationship between hubs and authorities can be expressed mathematically using our previously introduced notation as

$$\begin{aligned} x &= A'^T y \\ y &= A' x \end{aligned} \tag{1.7}$$

$\Leftrightarrow$

$$\begin{aligned} x &= A'^T A' x \\ y &= A' A'^T y \end{aligned} \tag{1.8}$$

Thus x is the eigenvector to $A'^T A'$ with eigenvalue 1, and likewise y is the eigenvector to $A' A'^T$ with eigenvalue 1.

When returning web pages relevant to a query the authority weight, $y(p)$, denotes the rank of a page. A user study was performed evaluating the quality of the authorities returned by a system based on the hub and authority framework [18]. A number of searches were performed on topics and the top five authorities and top five hubs returned were compared to a manually generated compilation of pages within a topic from the WWW search service on *Yahoo!* [30]. Users were asked to assess the quality of the pages returned by each approach. The results were that the quality of pages returned by the system based on hubs and authorities were comparable with the manually generated compilation - even a little better [18].

Compared to the PageRank algorithm introduced by Google, the hub and authority framework is a two-level model in which the hubs and authorities reinforce one another. PageRank is a one-level model in which authorities reinforce authorities. Of course the computational complexity of the two-level approach is higher. Whereas it is feasible to carry out the PageRank calculation on the entire set of web pages at crawl time, according to Brin et al. [3] it can be done in a couple of hours, it is too demanding for the two-level approach. Kleinberg argues that it is not necessary to carry out the calculation on the entire set of crawled pages [18]. Instead it can be done locally on the set returned by the search engine at query time. But

this is also demanding for search engines that need to process millions of queries per day and in [23] it is argued that this is not feasible.

The Intelligent Surfer - Query dependent PageRank

The link models considered so far have in common that they don't take the contents of documents into consideration when determining authoritative pages within an interconnected set of documents - they only consider the link structure. The Intelligent Surfer model [23] recognizes the idea that pages with many inlinks should be considered authoritative and also the PageRank idea that the inlinks should be weighted by the authoritative weight of the linking page. But the Intelligent Surfer is motivated by the observation that the PageRank score "ignores whether or not the page is relevant to the query at hand ... If one issues a query containing the word "jaguar", pages containing the word "jaguar" that are also pointed to by many other pages containing "jaguar" are more likely to be good choices than pages that contain "jaguar" but have no inlinks from pages containing it..." [23].

In order to take the relevance to the query at hand into consideration the inlinks, A'_{ij}, to page j are not only weighted by the (modified) authority weight, $\text{PageRank}_q(j)$, but also by a measure of relevance of page j to query q. Mathematically the extension to equation 1.6 can be stated as [23]

$$\text{PageRank}_q(j) = (1-\beta)P'_q(j) + \beta \sum_i A'_{ij}\ \text{PageRank}_q(i)\ P_q(i \to j) \quad (1.9)$$

where $P_q(i \to j)$ and $P'_q(j)$ denote a normalized relevance measure of document j to the query q. If $R_q(j)$ is the non-normalized relevance measure we have that

$$P_q(i \to j) = \frac{R_q(j)}{\sum_{k \epsilon F_i} R_q(k)} \qquad P'_q(j) = \frac{R_q(j)}{\sum_{k \epsilon W} R_q(k)} \quad (1.10)$$

where F_i is the set of outlinks from page i and W is the set of all pages. One simple query dependent relevance score could be $R_q(j) = 1$ if the term q appears on page j and 0 otherwise.

In a simple setup PageRank_q scores can be precomputed at crawl time for all terms and all documents and then combined at query time assuming that the terms are independent. This makes it a feasible approach for a search engine, it is argued in [23].

PLINK

Like the Intelligent Surfer model, the PLINK model introduced by Hofmann and Cohn [7] also recognizes the idea of combining link and document

content, but the PLINK model takes a quite different approach than the ones presented so far when determining authoritative pages within a topic. PageRank, the Hubs and Authorities framework and the Intelligent Surfer were basically inspired by the work in Bibliometrics. The calculation of page authority was determined by eigenvector calculations. In Bibliometrics the authority/importance of pages was computed as the primary eigenvector to the citation structure matrix, A.

In PLINK the eigenvector calculation is replaced by a probabilistic model based on the idea of a number of underlying topics. The underlying topics generate the words and links according to estimated frequencies. Thus a topic, z_1 can be represented by its probabilities of generating each word, $p(w|z_1)$, and its probabilities of generating each link or citation, $p(c|z_1)$. Also, according to [7], the PLINK model gives the strengths of each document within a topic, $p(d|z_1)$ [1]. Finding authoritative pages within a topic, z_1, is accomplished by retrieving the documents with the highest values in $p(d|z_1)$. We will get back to how all these values are calculated in section 2.1.6.

The PLINK model is capable of finding authoritative pages for relevant web pages, but more importantly it does so by representing word and link information in a solid statistical model. Therefore all sorts of inferences are possible, such as predicting links of a document by knowing its word content.

1.3 Problem description

We have now looked into applications of statistical text modelling, which is the basic building block of many computer systems dealing with Information Retrieval, Data mining and related areas. This has given us a feeling of the current status on how successful these models are. In some areas the performance of computer systems is comparable with humans. For instance ranking the one relevant page to a query among 1.25 millions of pages as number 1.39 on average seems like it couldn't be done any better by humans. On the other hand retrieving only 25 % relevant pages relevant to a query in a web searching task seems like a human would be better at assessing the relevance of the web pages if time was not a critical factor.

The survey of statistical text modelling applications has lead us to a class of statistical models dealing with link structure between text documents. This is a very interesting class of statistical models, because the link modelling problem and our initial matching problems are very much alike. Whereas the link models are concerned with links between text documents,

[1] Actually the form of the PLINK model described in [7] does not give the possibility of computing $p(d|z_1)$. A different version is necessary. See "Implications of the chosen form of PLINK" in section 3.4.5 for an explanation.

the matching problem is concerned with matches between text documents such as job applicants, dating profiles and peers in P2P networks. Predicting links by knowing the document content is therefore a very similar application to predicting good matches.

Therefore in this report we shall focus on link models. We will try to build a model that can predict links between documents and work under the assumption that if this task can be solved reliably the same model can also be used on the matching problem. The link problem has the advantage that it is very easy to get data as opposed to getting data for the matching problem. E.g. document collections interconnected by links can be retrieved from the WWW. That makes the link prediction a feasible task for this work.

Building link models requires a good understanding of statistical models for text documents. Therefore the first goal will be to get a good understanding of relevant statistical models for text documents. After that, link models will be analyzed. The statistical models will be worked on both from a theoretical point of view and from an experimental point of view. That is, the theory and mathematics behind the models will be presented and experiments with the models will be presented.

In particular the focus of this work will be on the PLINK model and related models. The PLINK model has been chosen, because, as we have seen, it provides a statistical model in which document contents and links can be combined. Other related statistical models are presented for comparison as well.

Chapter 2

Theory

2.1 Statistical Models in Information Retrieval

In this section we shall explore some specific statistical models, which has been applied to the area of information retrieval. The objective is to give background information on the nature of such statistical models by looking in detail at a number of models. Also it is the objective to find statistical models suitable for our task - that is models capable of representing both word content and link structure, which will provide us with the tools necessary to solve the problem of predicting links between documents.

2.1.1 Vector Space Model

Before any information can be retrieved from a document collection the documents first have to be indexed according to some document representation. The first task of indexing is therefore to decide on a representation of each document in the collection. The simplest choice is to save the entire document content, but this is often too space consuming and, more importantly, does not facilitate efficient information retrieval. Rather a document representation extracts a number of features from each document and a document is then represented as a vector of those features. Next the indexing task has to consider the importance of each feature in content identification [27]. That is, some features might not be as important in distinguishing between documents as others and should therefore be assigned lower weights of importance. The last task to consider is how to measure the similarity between documents or between queries and documents when retrieving information. The three indexing tasks of assigning features to documents, assigning weights of importance to features and deciding on a similarity measure compose what we will call a Vector Space Model for information retrieval [14].

The most popular Vector Space Model is the one that uses as features the terms appearing in each document. It is obvious that the terms of

a document tells a lot about the content of this document and therefore justifies this choice of features. On the other hand it is also obvious that a lot of information is not contained in this representation. Specifically all information related to the ordering of the words is not preserved. As a weight of importance the terms are first weighted by their frequency in each document. Now we can form the term-document matrix [27] to represent the entire document collection given by

$$\underset{txd}{X} = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1d} \\ f_{21} & f_{22} & \cdots & f_{2d} \\ \vdots & \vdots & & \vdots \\ f_{t1} & f_{t2} & \cdots & f_{td} \end{bmatrix} \tag{2.1}$$

where f_{ij} denotes the term frequency of term i in document j.

If we assume that the lengths of the documents are not important in the content representation then we want to normalize the document vectors, that is the columns of the term-document matrix.

The weighting of the terms can be extended by other measures of importance. In [27] a number of measures are examined. An important point is that very common words are not very useful in distinguishing between documents whereas terms occurring only in a few documents provides a better way to distinguish those documents from the rest. A very common term weighting to employ is the inverse document frequency, which captures exactly this concept [17]. It is given by

$$idf_i = \log(\frac{N}{n_i}) \tag{2.2}$$

where N is the total number of documents and n_i is the number of documents in which term i occurs. This gives us a different weight for each row in the term-document matrix which is multiplied by the term frequencies.

Other extensions to the vector space model presented in [17] include the use of a stop list and stemming. The use of a stop list removes the most common words (like "and", "the" and "for") which are known not to be important for the meaning of a document. Stemming also reduces the dimensionality by mapping words to their root form. The result is that words with different suffixes but with the same semantic like "find" and "finding" are mapped to the same feature in the document representation.

The similarity measure usually applied with the vector space model is the dot product of document vectors. If the vectors are normalized this corresponds to the cosine to the angle between the document vectors. Another possibility is the euclidian distance between pairs of document vectors.

2.1.2 Latent Semantic Indexing (LSI)

The basic idea behind latent semantic indexing is that instead of representing the documents in the high dimensional term space, the documents can be represented in a low dimensional latent semantic space. This has a number of advantages apart from the obvious one of saving space.

Dimensionality reduction

The dimensionality reduction is performed by a Singular Value Decomposition (SVD) of the term-document matrix [19] such that

$$\underset{txd}{X} = \underset{txd}{U} \cdot \underset{dxd}{S} \cdot \underset{dxd}{V^T} \approx \underset{txd}{\widehat{X}} = \underset{txk}{U} \cdot \underset{kxk}{S} \cdot \underset{kxd}{V^T} \tag{2.3}$$

where k is the dimensionality of the new space. This value has to be chosen. E.g. it can be chosen to be the number of topics that the document collection contains. U and V are called the left and right singular vectors respectively and S contains the singular values. U tells us how the k principal components describing the low dimensional space are composed of the terms of the original space. V tells us the representation of each document in the new space. That is the indexing has changed from representing documents by term frequencies to representing documents by their value for each of the k principal components.

The approximation is performed by choosing the k largest singular values of the SVD and discarding the rest. By doing this it is obvious that some information is lost. The question is whether it is desirable to loose this information. In [28] it is argued that performing the LSI, which corresponds to dropping small eigenvalues, eliminates noise in data (statistically dubious information) and in the case of representing queries in the latent space it compensates for the lack of an exhaustive word count in the query (specification error). Also LSI is believed to have some other desirable properties. Because the components used to describe the documents are composed of the old terms, it is possible that a query using the word "buy" matches documents that doesn't contain this term, but contains other related terms such as "purchase". That is the new components represent an artificial "topic" described by values of the old terms, and it deals with synonymy of words, which are believed to have high values in the same component. In this way LSI captures the latent semantic of the query. Polysemy is only partially solved by LSI . Since every term is represented by just one point in space it is not possible to represent multiple meanings of a word. LSI provides a kind of averaging over the different meanings of polysemous words [9].

As with the simple vector space model, the usual similarity measure is the dot product - also called the cosine similarity. In LSI the dot product is just computed between document vectors in the low dimensional latent semantic space. Usually one is interested in comparing queries with documents. The

query does not contribute to the SVD, but the query is represented in the same space by projecting the query vector onto the k components, and then the dot product is computed with this new query representation.

Applications

LSI is a popular model in information retrieval applications. The task of retrieving a number of documents from a document collection most relevant to a given query is a common task. The task includes matching the query with the documents in the collection. In the LSI framework this is handled by representing the query in the latent semantic space, applying the similarity measure between the query and all the documents in the collection and then returning the documents most similar to the query. The projection of the query onto the LSI space is computed by rewriting the term-document decomposition from equation 2.3 to obtain

$$\underset{kxd}{V^T} = \left(\underset{txk}{U} \cdot \underset{kxk}{S}\right)^T \cdot \underset{txd}{\widehat{X}} \tag{2.4}$$

If we have a query, q, and we want to find the representation, q_k, in the latent semantic space, we project it onto $\left(\underset{txk}{U} \cdot \underset{kxk}{S}\right)^T$ in the following way:

$$\underset{kx1}{q_k^T} = \left(\underset{txk}{U} \cdot \underset{kxk}{S}\right)^T \cdot \underset{tx1}{q} \tag{2.5}$$

The cosine similarity is now measured between q_k and the columns in V^T corresponding to each document in the collection.

Clustering is another application of the LSI model. Later in the report we shall look at different classification schemes e.g. based on clustering, but a very simple way of clustering with the LSI model is to use a number of clusters, k, equal to the number of principal components. Each document is assigned to the cluster for which it has the largest principal component.

2.1.3 Independent Component Analysis (ICA)

Independent Component Analysis is another technique for decomposing the term-document matrix into a latent semantic space. As such we could just state the decomposition similar to equation 2.3, but in order to understand the reasoning behind the ICA decomposition it is useful to have a look at the main application of ICA, which is separating acoustic signals.

ICA is a technique which can be used to separate a number of source signals which for some reason have been mixed together. Here we shall use the example of recovering acoustic signals. A number of sources, k, are

assumed to produce acoustic signals. They can be thought of as "speakers". Correspondingly a number of "microphones", t, record the acoustic signals, but the recorded signals, X, are a mix of the acoustic signals from all the speakers and we cannot tell what the original source signals were. ICA is a technique to recover the original source signals, S. That is, the recorded signals, X, can be decomposed into the source signals, S, and a mixing matrix, A, [19] according to

$$\underset{txd}{X} = \underset{txk}{A} \cdot \underset{kxd}{S} \tag{2.6}$$

The acoustic signals are sampled d times, such that S gives us d values for each of the k sources. A is called the mixing matrix because it denotes how the source signals are mixed together at each of the "microphones", t. Applications for acoustic signal recovering include hearing aids, which corresponds to the "microphones". A hearing aids user is interested in recovering the original sources of acoustic signals, which could be humans or noise. By recovering the source signals filtering out the noise is possible and very helpful to the users.

In the text document domain the interpretation of sources and recorded signals is different. In this domain the sources can be thought of as "topics", which are sampled d times - where d denotes the number of documents. What we observe is a mix of these "topics" in the term-document matrix. Each term is a mix of topics and the mixing matrix, A, tells us how the topics are mixed for each term. The decomposition of the term-document matrix in equation 2.6 can be used to recover how each document was produced as a mix of topics.

Interpreting each document as generated by a mix of underlying topics is exactly the same interpretation as in LSI. The big difference between LSI and ICA is the assumptions on the underlying topics or components that generate the documents. Since LSI is based on SVD the assumption is that the components are orthogonal. As a consequence the topics found by LSI have very little in common. ICA on the other hand restricts the components to be statistically independent. We will not go further into the definition of "statistically independent", but emphasize that in ICA it is perfectly fine that two topics are very much alike and have a lot in common.

Another way to look at the decomposition is to look at how it will be used in information retrieval. The approach here is similar to the LSI decomposition. We have a document collection, which is represented in the high dimensional term space, but it is possible to represent the documents in a lower dimensional ICA latent semantic space. The ICA decomposition in equation 2.6 describes how the original term-document space can be decomposed into the k-dimensional latent semantic space. The features of this space are the k components, and each document is indexed by its value for each of the components given by the source signal matrix, S.

So far we haven't discussed how the decomposition is computed. The LSI decomposition was computed by the well-known SVD, but ICA decomposition is more complicated and different techniques exist. As opposed to SVD iterative updating rules for A and S have to be used for ICA. One derivation of the updating rules are based on maximum likelihood, but we will not go further into details with this subject here. For details see e.g. [19].

It is possible to take advantage of the LSI decomposition before ICA decomposition takes place as described in [19]. Instead of using the original term-document matrix as input to the ICA decomposition the matrix product $\underset{kxk}{S} \cdot \underset{kxd}{V}^T$ from the LSI decomposition is used as input. The resulting decomposition then looks like this:

$$\underset{txd}{X} \approx \underset{txd}{\widehat{X}} = \underset{txk}{U} \cdot \underset{kxk}{S} \cdot \underset{kxd}{V^T} = \underset{txk}{U} \cdot \underset{kxk}{A} \cdot \underset{kxd}{S} \tag{2.7}$$

2.1.4 Probalistic Latent Semantic Indexing (PLSI)

A lot of work has focused on explaining why LSI works (e.g. [28]). One reason might be that LSI suffers from a lack of a solid statistical foundation. With this in mind a similar model, PLSI , has been proposed by Hofmann [14], which provides the desired solid statistical foundation. Like LSI, PLSI also assumes that the documents described in the high dimensional term space can be represented in a lower dimensional latent semantic space. But unlike LSI the components of the new space is not found by SVD. Instead PLSI presents a latent variable model, which associates with each observation an unobserved class variable, z. One observation, (w, d), is then a specific occurrence of a term, w, in a document d.

The parameters of the model without latent variables are the $p(w, d)$'s, which gives us $T \cdot D$ parameters in total. But by introducing the conditional independence assumption that conditioned on the latent variable, z, the words are generated independently of the specific document, we end up with only $T \cdot N_z + D \cdot N_z + N_z$ parameters, where N_z is the number of latent variables. That is

$$p(d, w) = \sum_z p(z)p(d, w|z) = \sum_z p(z)p(w|z)p(d|z) \tag{2.8}$$

In LSI one document could be described by its values for each principal component. But in PLSI documents are characterized by a mixture of factors with weights $p(z|d)$. The hidden factors can describe underlying topics in the documents and one document is then characterized by a mixture of these topics, such that it has high values for topics, which are dominant in the document.

While in LSI one component could be characterized by its weight for each term, in PLSI one factor can be described by its probabilities of generating each term, $p(w|z)$.

The generative model

The focus so far has been on the latent semantic space of the PLSI model as opposed to the LSI model. The document representations have been compared as well as the components of the latent space. Another way to look at models is as generative models. That is, what are the assumptions about how data is generated introduced by the model? This is a useful way of thinking about the model for several reasons. One reason is that we want to use a model, which has reasonable assumptions about data generation. The assumptions should be close to the way that we think data is generated in the specific problem that we are looking at. Another reason why it is useful to look at statistical models as generative models is that this gives us a way to choose among different models or different sets of parameters. If we have data for the specific problem we can choose the model in the following way: Among the different possible models we want to choose the one, which has the greatest probability of generating the data for the problem. This probability is called the likelihood of data if the parameters or model is what we are optimizing. We shall use this approach when estimating the parameters for the PLSI model.

So how is data generated according to the assumptions of the PLSI model? Actually there are several interpretations, but the general idea is this: We obtain a number of observation pairs, (w, d), which denotes a specific word in a specific document. The PLSI model generates data according to the probability, $p(w, d)$, that it assigns to all possible observation pairs, (w, d). One immediate interpretation is then given according to equation 2.8:

- Pick a latent variable, z, with probability $p(z)$.
- Generate a document, d, and a word, w, according to the probabilities $p(d|z)$ and $p(w|z)$ respectively.

A different interpretation of the generative model can be given by writing $p(w, d)$ in a different way:

$$p(w, d) = p(d)p(w|d) = p(d) \sum_z p(w|z)p(z|d) \tag{2.9}$$

Now data can be thought of as generated in the following way:

- Pick a document, d, with probability $p(d)$.
- Pick a latent variable, z, with probability $p(z|d)$.
- Generate a word, w, with probability $p(w|z)$.

In both interpretations we obtain the observation pairs, (w, d).

Estimation of model parameters

In order to estimate the model parameters, we want to maximize the likelihood of the observed data. Since it is easier to maximize the log likelihood of the data, we will do that. We can describe the data observations, (w, d), by $n(w, d)$, which denotes how many times a word occurred in a document. The likelihood of the entire data set, X, becomes

$$\begin{aligned} p(X|model) &= p(w_1, d_1)^{n(w_1,d_1)} p(w_2, d_1)^{n(w_2,d_1)} \cdots p(w_t, d_d)^{n(w_t,d_d)} \\ &= \prod_d \prod_w p(w, d)^{n(w,d)} \end{aligned} \tag{2.10}$$

And the log likelihood of the data becomes

$$\begin{aligned} \ln p(X|model) &= \sum_d \sum_w n(w, d) \ln p(w, d) \\ &= \sum_d \sum_w n(w, d) \ln \sum_z p(z)p(w|z)p(d|z) \end{aligned} \tag{2.11}$$

This function is difficult to optimize, because it contains the logarithm to a sum. We will make use of Jensen's inequality (see [1] p66), which says that

$$\ln(\sum_j \lambda_j x_j) \geq \sum_j \lambda_j \ln(x_j) \tag{2.12}$$

where the λ_j's have to sum to one. We will introduce the posterior probabilities of the latent variables, $p(z|w, d)$, which will play the role of the λ_j's, since they also have to sum to one.

$$\begin{aligned} \ln p(X|model) &= \sum_d \sum_w n(w, d) \ln \sum_z p(z|w, d) \frac{p(z)p(w|z)p(d|z)}{p(z|w, d)} \\ &\geq \sum_d \sum_w n(w, d) \sum_z p(z|w, d) \ln \frac{p(z)p(w|z)p(d|z)}{p(z|w, d)} \\ &\equiv F(\Theta, p(z|w, d)) \end{aligned} \tag{2.13}$$

This new function, F, introduced presents a lower bound on the likelihood of the data, because of the inequality. If we want to maximize the likelihood of data then we also have to maximize L. The function depends partly on the parameters of the model, $\Theta \equiv \{p(z), p(w|z), p(d|z)\}$, and partly on the mixing proportions, $p(z|w, d)$. By optimizing F with respect to the parameters and the mixing proportions respectively we will arrive at the two iterative updating steps for the EM algorithm. The reason why it is necessary to

iterate is that the parameters will depend on the mixing proportions and the other way around.

When we optimize F we have to take constraints on the densities into consideration, since they have to sum to one. Therefore we define a Lagrangian function, $L(\Lambda, \Theta, p(z|w,d))$, where we have added to F, the constraints multiplied by Lagrange multipliers, Λ. Now the gradient, ∇L, has to be zero in order to satisfy the constraints and in order to maximize F.

$$\begin{aligned} L(\Lambda, \Theta, p(z|w,d)) = & \sum_d \sum_w n(w,d) \sum_z p(z|w,d) \ln \frac{p(z)p(w|z)p(d|z)}{p(z|w,d)} \\ & + \sum_z \lambda_z (\sum_w p(w|z) - 1) + \sum_z \mu_z (\sum_d p(d|z) - 1) \\ & + \gamma (\sum_z p(z) - 1) + \sum_{w,d} \rho_{w,d} (\sum_z p(z|w,d) - 1) \end{aligned} \tag{2.14}$$

We first find the updating rules for the parameters by differentiating L with respect to the parameters.

$$\begin{aligned} 0 = \frac{\partial L}{\partial p(z)} &= \sum_{d,w} p(z|w,d) \frac{p(z|w,d)}{p(z)p(w|z)p(d|z)} \frac{p(w|z)p(d|z)}{p(z|w,d)} + \gamma \\ \Leftrightarrow p(z) &= -\frac{1}{\gamma} \sum_{d,w} n(w,d) p(z|w,d) \end{aligned} \tag{2.15}$$

The constant, $-\frac{1}{\gamma}$, can be found by summing over z and utilizing the constraint that $p(z)$ has to sum to one. That is

$$\begin{aligned} \sum_z p(z) = 1 &= -\frac{1}{\gamma} \sum_{d,w} n(w,d) \sum_z p(z|w,d) \\ \Leftrightarrow -\frac{1}{\gamma} &= \frac{1}{\sum_{d,w} n(w,d)} \end{aligned} \tag{2.16}$$

And so we obtain the updating rule for $p(z)$ to be

$$p(z) = \frac{\sum_{d,w} n(w,d) p(z|w,d)}{\sum_{d,w} n(w,d)} \tag{2.17}$$

The derivation of the updating rules for the other parameters, $p(w|z)$ and $p(d|z)$ are similar, and we end up with the following [14]:

$$p(w|z) = \frac{\sum_d n(w,d)p(z|w,d)}{\sum_{d,w} n(w,d)p(z|w,d)} \tag{2.18}$$

$$p(d|z) = \frac{\sum_w n(w,d)p(z|w,d)}{\sum_{d,w} n(w,d)p(z|w,d)} \tag{2.19}$$

Updating the parameters, $p(z)$, $p(w|z)$ and $p(d|z)$ composes the M-step of the EM algorithm, because the parameters are maximized given the current value of the latent variable, z.

Now we want to find the updating rule for the mixing proportions, $p(z|w,d)$. The approach is similar.

$$\begin{aligned} 0 = \frac{\partial L}{\partial p(z|w,d)} &= n(w,d)(\ln \frac{p(z)p(w|z)p(d|z)}{p(z|w,d)} \\ &+ p(z|w,d)\frac{p(z|w,d)}{p(z)p(w|z)p(d|z)}(-\frac{p(z)p(w|z)p(d|z)}{p(z|w,d)^2})) + \rho_{w,d} \\ \Leftrightarrow p(z|w,d) &= e^{-\frac{\rho_{w,d}+1}{n(w,d)}} p(z)p(w|z)p(d|z) \end{aligned} \tag{2.20}$$

where the constant in front of the parameters can be found by summing over z and utilizing the constraint on $p(z|w,d)$, so that

$$\begin{aligned} \sum_z p(z|w,d) = 1 &= e^{-\frac{\rho_{w,d}+1}{n(w,d)}} \sum_z p(z)p(w|z)p(d|z) \\ \Leftrightarrow e^{-\frac{\rho_{w,d}+1}{n(w,d)}} &= \frac{1}{\sum_z p(z)p(w|z)p(d|z)} \end{aligned} \tag{2.21}$$

And the updating rule for $p(z|w,d)$ becomes

$$p(z|w,d) = \frac{p(z)p(w|z)p(d|z)}{\sum_z p(z)p(w|z)p(d|z)} \tag{2.22}$$

When estimating parameters with the EM algorithm, updating $p(z|w,d)$ composes the E-step, because we update the expected value of the hidden/latent variable, z.

Folding-in new documents

For information retrieval purposes it is important, as we have discussed, to be able to evaluate the similarity between documents or documents and queries. In PLSI a document is represented by $p(z|d)$. Now, when estimating the model parameters these estimates can be computed for all the training

documents by virtue of $p(d|z)$ and Bayes' rule, but for unseen documents we have no document representation so far. The way to obtain the $p(z|d_{new})$ is by folding-in the new documents with the EM algorithm [14] to obtain $p(d_{new}|z)$. This is necessary because $p(d_{new}|z)$ depends on $p(z|w, d_{new})$ and vice versa.

In practice the folding-in of new documents is done by keeping from the trained model only the $p(w|z)$ and $p(z)$ parameters. That is, we fix these parameters when estimating the $p(z|d_{new})$ parameters, which is done by iterating between the EM steps 2.19 and 2.22. The old $p(z|d)$ are discarded during the folding-in of new documents.

This is a far more complicated and computationally heavier procedure compared to e.g. the LSI model. Some short-cuts have been suggested, though. In [11] and online version of EM is described for folding-in unseen documents where only one iteration is necessary.

Also another approach is suggested here. Again we compare the PLSI model to the LSI model. In LSI a representation of a new document would be obtained by projecting the term frequencies of the new document onto the left eigenvectors, U, as with the query, q, in equation 2.5. Similarly a document representation in the PLSI model could be obtained by projecting the term frequencies onto the $p(w|z)$ vectors in the PLSI latent space, so that

$$p(z|d_{new}) = \sum_{w} p(w|z)n(w, d) \tag{2.23}$$

Applications

The decomposition of the original high dimensional documents represented by the word frequencies has some advantages similar to the ones described for LSI. One advantage of the decomposition is that the sparse term-document matrix is decomposed into the not sparse $p(z|d)$ representations where less data is needed to facilitate meaningful applications.

In information retrieval it is often the case that we want to retrieve documents from a collection similar to a given query. This task can be solved in our PLSI framework, by training the PLSI model on the document collection to obtain the document representations, $p(z|d)$. The query representation is then computed by folding-in the query to obtain $p(z|q)$. Now the similarity between the query and all the documents in the collection can be determined for instance by computing the dot product (cosine similarity) between the query representation and all the documents, and the documents with the greatest similarities are retrieved.

Another application is clustering of the documents in a collection. We want to cluster the documents that are similar into one cluster thus obtaining a predetermined number of document clusters, k. This can be done in

several ways, but again the starting point is the document representations, $p(z|d)$. One simple way is to assign each document to the cluster for which it has the greatest value of $p(z|d)$. This relates to more conventional clustering methods like e.g. gaussian mixture models. In mixture models, k distributions are fitted to the observed data, x, maximizing the posteriori distribution, $p(x|\theta)$ where θ denotes the parameters of the k distributions. Each data point or document, x_n, is assigned to the most likely likely component determined by comparing $p(k|x_n)$ for all components. Thus in PLSI $p(z|d)$ plays the same role as $p(k|x_n)$ in clustering with mixture models. For a more detailed description on how to use Generalizable Gaussian Mixture models for clustering see [13].

The factor decomposition of a document collection allows us to determine what topics are present in the collection. This is possible because the factors, z, can be thought of as the topics present in the collection, and the components, $p(w|z)$, tell us what words describe that topic. The words with the largest values for a given topic in $p(w|z)$ provide a keyword description for that topic. This application is also useful in combination with clustering. If the documents have been clustered, then the $p(w|z)$ can again provide a description of each cluster.

If we have labeled documents and want to classify new documents, that is, assign a label to each new document, the PLSI model is also useful. There are many different ways to perform classification and we shall explore some of them later in this report. One approach is to start by clustering the documents and then assign labels to each cluster. New documents are then given the label of the cluster into which they are put (see section 3.2.1). A more advanced approach is to extend the PLSI model to perform supervised learning. This approach is taken in section 3.3.

2.1.5 Latent Dirichlet Allocation (LDA)

PLSI has been criticized by Blei, Ng and Jordan [2] for not being a fully generative model. The reason is that the model learns the topic mixtures $p(z|d)$ only for those documents on which it is trained and therefore there is no clean way to assign probabilities to previously unseen documents. The folding-in of new documents is therefore a complicated process, because $p(z|d_{new})$ has to be estimated with the EM algorithm as we have seen.

Furthermore there is another drawback of letting the training documents take part in the model parameters explicitly. The number of parameters grow linear in the number of training documents and this imposes the risk of overfitting training data. Therefore in [2] the PLSI model is also criticized for introducing the need for "tempering" heuristic to smooth the parameters. That is the Tempered EM algorithm used by [14],

LDA is introduced to overcome the shortcomings of PLSI. Like PLSI LDA is also a generative model. But in LDA the training documents do not take part in the model parameters explicitly. Instead of the $p(z|d)$ parameters, a random variable θ is introduced. θ has the same dimension as z, that is θ is the mixture of topics in the documents. But instead of estimating the mixture probabilities for each training document ($p(z|d)$), θ is sampled from a Dirichlet distribution for each document. Data is generated in the following way:

- Obtain a mixture of topics, $\theta = (\theta_1, ..., \theta_{N_z})$, for the document, d, by sampling θ from the Dirichlet distribution, $D(\theta, \alpha)$.
- Pick a topic z_k with probability $p(z_k|\theta) = \theta_k$.
- Pick a word, w, with probability $p(w|z_k)$.

Compared to the generative model for the PLSI model the big difference is that in PLSI we pick a document, d, with probability $p(d)$ to obtain the mixture of topics, $p(z|d)$ for that document. In LDA we sample θ to obtain the mixture of topics $\theta = (\theta_1, ..., \theta_{N_z})$. Now topics are picked either according to $p(z|d)$ or according to θ. Finally words are sampled according to $p(w|z)$ in either models, where $p(w|z)$ are parameters of the models estimated based on training data. They denote a multinomial distribution over the words for each topic.

There is a lot fewer parameters in the LDA model compared to the PLSI model. Whereas there is $N_z \cdot D$ of $p(z|d)$ parameters in the PLSI model there is only N_z parameters in the Dirichlet distribution from which θ is sampled for each document. These parameters are denoted $\alpha = (\alpha_1, .., \alpha_{N_z})$. The number of parameters grow linear in the number of documents in the PLSI model. In the LDA model on the other hand the α parameters have to capture a sort of averaging over the mixture of topics in all the documents, but no extra parameters are added by introducing new documents and therefore the risk of overfitting is minimized compared to the PLSI model.

The full LDA model introduced by [2] for the probability of a document

looks like this:

$$\begin{aligned} p\Big(d|\alpha, p(w|z)\Big) &= \int_\theta \prod_w \{\sum_k p(w|z_k)p(z_k|\theta)\}^{n(w,d)} D(\theta|\alpha)d\theta \\ &= \int_\theta p(d|\theta)D(\theta|\alpha)d\theta \end{aligned} \tag{2.24}$$

where $D(\theta|\alpha)$ denotes the Dirichlet density , $n(w,d)$ is the number of times w occurs in d and $p(z_k|\theta) = \theta_k$. A document in LDA is just treated as the vector of words that it contains. It is interesting to compare this with the probability of a new document in the PLSI model. We have not stated it yet, but it would look like this:

$$\begin{aligned} p\Big(d_{new}|p(z), p(w|z), p(d_{train}|z)\Big) &= \sum_{d_{train}} \{\prod_w \{\sum_z p(w|z)p(z|d_{train})\}^{n(w,d)}\} p(d_{train}) \\ &= \sum_{d_{train}} p(d_{new}|d_{train})p(d_{train}) \end{aligned} \tag{2.25}$$

where $p(d_{train}) = \sum_z p(d_{train}|z)p(z)$.

It is now interesting to see that whereas we integrate $p(d|\theta)$ over the continuous Dirichlet distribution in LDA, in PLSI we sum $p(d_{new}|d_{train})$ over all training documents, which is a discrete sum. So actually there is a way to assign probabilities to previously unseen documents but apparently it is not clean according to [2], because the training documents are not really treated as documents composed of words but more like a dummy variable.

In [12] the similarities between PLSI and LDA are examined. It is claimed that PLSI is actually a special case of LDA when the Dirichlet distribution is uniform. One way to look at estimation of the parameters in LDA is that for each document we want to find the maximum a posteriori value for θ_d that is the most likely mixing proportions of topics for that document given the data. If we estimate θ_d for every document then the parameters, $\alpha_1..\alpha_{Nz}$, of the Dirichlet distribution and the $p(w|\theta)$ parameters can be estimated by maximum likelihood estimation. In [12] it is said that if we let all the α parameters equal 1 such that the Dirichlet distribution is uniform, then the estimations of $p(w|z)$ in LDA equals the estimations of $p(w|z)$ from PLSI. The folding-in of new documents in PLSI can also be seen as estimation of the Dirichlet variable for the new document.

2.1.6 PLINK - a probalistic model for links and terms

In this section a model, PLINK, is introduced, which is an extension to PLSI. The extension is proposed by Hofmann and Cohn in [7]. Whereas PLSI only considers the terms as features in the documents, PLINK assumes that there exists links between the documents, which are also used as features. E.g. the links can take the form of hyperlinks as is the case of HTML documents or they can take the form of citations, which is the case of articles citing other articles. The hypothesis is that the links contribute to the semantic context of the documents and thereby enhance the chance of successful applications. If two documents have a similar citation pattern it is also more likely that they share the same context than documents with different citation patterns.

Joint probalistic model

The general idea is the same as in the PLSI model. With each observation, an unobserved class variable, z, is associated. In this case observations not only consist of word-document pairs (w, d), but also citation-document pairs, (c, d). The assumption of the model is that words and citations are created by the underlying factors represented by z. A straight forward way to express this is:

$$p(w,d) = \sum_z p(w|z)p(d|z)p(z), \quad p(c,d) = \sum_z p(c|z)p(d|z)p(z) \tag{2.26}$$

Parameters of the model would then be $p(w|z)$, $p(c|z)$, $p(d|z)$ and $p(z)$. Estimates of the model parameters would be derived by maximizing the likelihood of the observations given by

$$P(X|model) = \prod_d \left(\prod_w p(w,d)^{n(w,d)} \prod_c p(c,d)^{n(c,d)} \right) \tag{2.27}$$

But the approach taken in [7] is slightly different. In order to normalize the lengths of the documents, such that documents of different lengths are given the same importance, the word observations considered by the model are $\frac{n(w,d)}{\sum_w n(w,d)}$ and the citation observations are $\frac{n(c,d)}{\sum_c n(c,d)}$ where $n(w,d)$ and $n(c,d)$ denotes the number of words and citations in the document respectively. The corresponding probabilities for given observations are $p(w|d)$ and $p(c|d)$ respectively. Again the basic assumptions of the model is that the words and citations are generated by underlying factors, z, which make the words and the citations independent of the documents given the hidden factors, z. This can be expressed as

$$p(w|d) = \sum_z p(w|z)p(z|d), \qquad p(c|d) = \sum_z p(c|z)p(z|d) \tag{2.28}$$

The parameters of the PLINK model are therefore $p(w|z)$, $p(c|z)$ and $p(z|d)$. The common parameter, $p(z|d)$, for the conditional probabilities of words and citations couples the content and link features together.

Estimating the model parameters

When estimating the parameters, we consider a data sample and want to fit the data with the most likely parameters that generated the data. In [7] it is suggested to introduce a weighting factor α of the words vs. the links in the likelihood function, which facilitates deciding on the relative importance of the two types of features a priori. This gives us the following likelihood function, where X denotes the data sample

$$P(X|model) = \prod_d \left(\prod_w p(w|d)^{\alpha \frac{n(w,d)}{\sum_w n(w,d)}} \prod_c p(c|d)^{(1-\alpha)\frac{n(c,d)}{\sum_c n(c,d)}} \right) \quad (2.29)$$

The log likelihood becomes

$$\begin{aligned} \ln P(X|model) &= \sum_d \Big(\sum_w \alpha \, \frac{n(w,d)}{\sum_w n(w,d)} \ln p(w|d) \\ &\quad + \sum_c (1-\alpha) \frac{n(c,d)}{\sum_c n(c,d)} \ln p(c|d) \Big) \\ &= \sum_d \Big(\sum_w \alpha \, \frac{n(w,d)}{\sum_w n(w,d)} \ln \sum_z p(w|z)p(z|d) \\ &\quad + \sum_c (1-\alpha) \frac{n(c,d)}{\sum_c n(c,d)} \ln \sum_z p(c|z)p(z|d) \Big) \end{aligned} \quad (2.30)$$

By following the EM approach as in section 2.1.4 it is possible to derive the parameter updating rules for the parameters $p(w|z), p(c|z)$ and $p(z|d)$ respectively and the mixing proportions $p(z|w,d)$ and $p(z|c,d)$. These are given by Hofmann and Cohn in [7], but except for the mixing proportions they are not entirely correct. Therefore the derivation is outlined here.

We introduce the fractions of a word and a link in a document respectively as

$$W_d \equiv \frac{n(w,d)}{\sum_w n(w,d)}, \qquad C_d \equiv \frac{n(c,d)}{\sum_c n(c,d)} \quad (2.31)$$

Now we move on with the log likelihood function by introducing the mixing

proportions and taking advantage of Jensen's inequality.

$$\begin{aligned}
\ln P(X|model) &= \sum_d \Big(\sum_w \alpha W_d \ln \sum_z p(z|w,d) \frac{p(w|z)p(z|d)}{p(z|w,d)} \\
&\quad + \sum_c (1-\alpha) C_d \ln \sum_z p(z|c,d) \frac{p(c|z)p(z|d)}{p(z|c,d)} \Big) \\
&\geq \sum_d \Big(\sum_w \alpha W_d \sum_z p(z|w,d) \ln \frac{p(w|z)p(z|d)}{p(z|w,d)} \\
&\quad + \sum_c (1-\alpha) C_d \sum_z p(z|c,d) \ln \frac{p(c|z)p(z|d)}{p(z|c,d)} \Big) \\
&= F(\Lambda, \Theta, p(z|w,d), p(z|c,d))
\end{aligned} \tag{2.32}$$

When optimizing F the constraints on the parameters are taken into consideration by introducing the lagrangian function, L as in section 2.1.4.

$$\begin{aligned}
L(\Lambda, \Theta, p(z|w,d)) &= F(\Lambda, \Theta, p(z|w,d), p(z|c,d)) \\
&\quad + \sum_z \lambda_z (\sum_w p(w|z) - 1) + \sum_z \mu_z (\sum_c p(c|z) - 1) \\
&\quad + \sum_d \gamma_d (\sum_z p(z|d) - 1) + \sum_{w,d} \rho_{w,d} (\sum_z p(z|w,d) - 1)
\end{aligned} \tag{2.33}$$

Here we will focus on the derivation of one of the parameters, $p(w|z)$. When taken the derivative of L we get

$$\begin{aligned}
0 = \frac{\partial L}{\partial p(w|z)} &= \sum_d \alpha W_d \frac{p(z|w,d)}{p(w|z)} + \lambda_z \\
\Leftrightarrow p(w|z) &= \frac{\alpha}{\lambda_z} \sum_d W_d \; p(z|w,d)
\end{aligned} \tag{2.34}$$

The constant $-\frac{\alpha}{\lambda_z}$ can be found by using the constraint that $p(z|w)$ has to sum to one. So

$$\begin{aligned}
\sum_w p(w|z) = 1 &= \frac{\alpha}{\lambda_z} \sum_d W_d \; p(z|w,d) \\
\Leftrightarrow -\frac{\alpha}{\lambda_z} &= \frac{1}{\sum_w \sum_d W_d \; p(z|w,d)}
\end{aligned} \tag{2.35}$$

Finally the updating rule for $p(w|z)$ is obtained

$$p(w|z) = \frac{\sum_d W_d \; p(z|w,d)}{\sum_w \sum_d W_d \; p(z|w,d)} \tag{2.36}$$

Compared to the updating rules given by Hofmann and Cohn this is the same except that they don't have the normalizing term in the denominator. However, According to [15] Hofmann and Cohn actually used normalizing constants in their experiments. By deriving the other updating rules one notices that the same is true for the other parameters, that is, normalizing constants are missing. Here they are given in a normalized version according to the EM algorithm.

$$\begin{aligned} p(c|z) &= \frac{\sum_d C_d\, p(z|c,d)}{\sum_c \sum_d C_d\, p(z|c,d)} \\ p(z|d) &= \frac{\alpha \sum_w W_d\, p(z|w,d) + (1-\alpha) \sum_c C_d\, p(z|c,d)}{\sum_z \left(\alpha \sum_w W_d\, p(z|w,d) + (1-\alpha) \sum_c C_d\, p(z|c,d)\right)} \end{aligned} \tag{2.37}$$

The formulae in equation 2.36 and 2.37 compose the M-step of the EM algorithm. Updating the mixing proportions compose the E-step and the updating rules are given like in [7] as

$$\begin{aligned} p(z|w,d) &= \frac{p(w|z)p(z|d)}{\sum_z p(w|z)p(z|d)} \\ p(z|c,d) &= \frac{p(c|z)p(z|d)}{\sum_z p(c|z)p(z|d)} \end{aligned} \tag{2.38}$$

In [7] it is shown that incorporating links in the model increases the classification accuracy in comparison with PLSI. Also other applications are suggested such as determining the most likely citation pattern based on the words in a new document. Similarly the most likely words can be determined based on the citations. An intelligent web crawler is suggested based on a principle of following links in those documents that are most likely to have links leading to the topic at interest. The topic is represented by a query in the latent semantic factor space. The intelligent web crawler is based on the concept of *Reference flow*, which will be introduced in section 3.4.5.

2.2 Choosing between learning algorithms

When performing experiments with statistical models often the goal is to choose the best one among two or more statistical models. In that case one needs some measure of performance to decide which algorithm achieves the best results. In this section we will look into comparing classifiers which is probably the most widely used application for statistical learners. When comparing classifiers one is interested in the performance on unseen data. Therefore the measure of performance is usually the error rate on a test set, that is the number of misclassifications performed by the classifier over the number of test examples. The reason why it is not a straight forward task to compare the error rates of two classifiers is that the results obtained from different runs are not equal. Therefore even if we train and test the classifiers several times and get that one classifier is better than the other, we still cannot be sure whether the opposite happens in the next case. That is why we need statistics to quantify how certain we are about our conclusions regarding the superiority of a specific classifier.

To conclude that one classifier is superior we can perform a statistical test. The goal is to quantify the conclusion with some level of confidence. But many different tests have been suggested in the literature. E.g. in [10] a number of tests are examined. The reason why there is no consensus about which test is the best one is that it is difficult to construct a test in which the assumptions of the test are fully fulfilled. The limited amount of available data is usually the troublemaker. For instance let's look at one of the simplest tests for deciding whether two classifiers perform equally well. One common assumption is that if we have enough test examples then the error rates of the classifiers, p_A and p_B, are approximately normally distributed. Furthermore p_A - p_B can be viewed as normally distributed, if we assume that p_A and p_B are independent [10]. The only problem is that usually they are not independent. When data is limited usually the classifiers are trained or tested on the same data. If training data by accident is a very bad representation of the entire population then both classifiers will perform worse than otherwise and in that case their error rates are highly correlated. This is just one example of violated assumptions. In general we should look for a test for which the assumptions are best fulfilled.

When constructing a test the thing we have to pay attention to is the sources of variation. If there was no variation then we didn't have to perform any test statistics, because we could be certain about the results from a single run. According to [10] four sources of variation have to be considered. These are

- Variation caused by variation in test data
- Variation in training data
- Internal randomness in the learning algorithm
- Noise in the labels of data

Because we only train and test on a fraction of the entire population we cannot expect to find the true error rate. If we have some randomness in the training procedure then the classifier will perform differently in different runs even on the same training and test data. Finally if the labels given to us are noisy then no learner can expect to perform better than the noise level in the labels [10].

In this report we shall consider classifiers which posses inherent randomness in the learning algorithm. Also data will be limited, so the variation caused by variation in training and test data is relevant. We will assume no noise in the labels. The internal randomness is caused by random initialization of the model parameters. To take this variation into account we need to train the model a number of times, because this will give us variation in the error rate and this variation can be taken into account in the statistical test. When comparing two classifiers we are interested in estimating the difference in error rates, $p_A - p_B$. If we can construct a 95% confidence interval for this quantity, and if 0 is not included in this interval then we can conclude that with 95% certainty one classifier is better than the other.

The basic assumption is that p_X is binomially distributed due to the sampling from the entire population, that is, the variation in training and test data. But for a large number of test examples this can be approximated by a normal distribution. We will also assume that the variation caused by the random initialization of the model parameters are normally distributed, that is, if there was no variation in training and test data then p_X would be normally distributed due to the internal randomness of the learning algorithm. In total p_X is assumed to be normally distributed due to the internal randomness and the variation in training and test data. Thereby $p = p_A - p_B$ is also assumed to be normally distributed. A number of measurements on p can be performed, and if these are independent a 95% confidence interval for the mean of this quantity can be found as ([5])

$$p = \overline{p} \pm \frac{s}{\sqrt{n}} t(n-1)_{0.975} \tag{2.39}$$

where n is the number of measurements of p, $\overline{p}$ is the mean of the measurements and s is the standard deviation of p given by

$$\overline{p} = \frac{1}{n-1} \sum_i (p_i - \overline{p})^2 \tag{2.40}$$

In practice a k-fold cross validation is performed in order to estimate p. This is done by dividing the data into k sets of equal size. Each of the k iterations are done by training the model on $k-1$ sets and measuring p on the set left out. The strength of this approach is that each test example is only used once in the estimation of p, which works in favor of the independence assumptions of the measurements of p. Since the training sets overlap the measurements are not fully independent though.

Chapter 3

Experiments

3.1 Data sets

This section introduces the data sets that will be used in the report. The two data sets are real world data. Later in the report also synthetic data sets will be constructed. Synthetic data is useful because it is easier to construct a data set such that you know exactly what results to expect from the statistical models.

3.1.1 MED data set

The MED data set is a collection of medical abstracts, which has been studied before, e.g. in [9]. Originally it consists of 1033 abstracts partially labeled with 30 different labels. As was done by Kolenda et al. in [19] in this report we shall use a subset consisting of only the first five classes containing 124 abstracts. Here's a brief outline of the five classes copied from [19]:

- The crystalline lens in vertebrates, including humans.
- The relationship of blood and cerebrospinal fluid oxygen concentrations.
- Electron microscopy of lung or bronchi.
- Tissue culture of lung or bronchial neoplasms.
- The crossing of fatty acids through the placental barrier. Normal fatty acid levels in placenta and fetus.

There's a couple of comments to make on the MED data set. First of all it is chosen because it has been worked on before and has been shown to be an easy data set to classify. In [19] a classification error rate of 8 % is obtained by using independent components to do classification [1]. Also it

[1] In a cross validation experiment a test set of 20 documents are classified in each resample. The best result is obtained with four independent components obtaining an average of 1.55 errors out of 20. This gives an error rate of $\frac{1.55}{20} = 8\%$

is reported that 3 classes are very easy to distinguish, but the last 2 classes tend to lump together.

In [9] the degree to which the MED data set appears like a real data set is in question. According to [9] a particular label is assigned to all the documents returned by a specific keyword search. Thus 30 different keyword searches have been performed resulting in 30 different labels, some of the classes with overlapping documents. This way of data set construction probably makes the data set particularly easy for classification. But the results on the MED data set are still important, since it doesn't really matter how the documents are labeled, as long as we don't use this information in our classifier.

3.1.2 WebKB data set

The WebKB data set contains 8,282 web pages collected from computer science departments from 4 universities in 1997 by the World Wide Web Knowledge Base project [8]. The web pages have been manually classified into 7 classes:

- student (1641)
- faculty (1124)
- staff (137)
- department (182)
- course (930)
- project (504)
- other (3764)

In this report we shall use a subset of the original WebKB data set. As in [13] only the 4 largest classes will be used and these are: *student, faculty, course and project*. *other* will not be used, since it is very noisy.

A desirable feature of the WebKB data set is that apart from containing word content it also contains links between the documents. Since this work focuses on relations between word contents and links data set this is a nice feature.

The WebKB data set is a lot more difficult data set than the MED data set. Hofmann and Cohn [7] obtains a classification error rate of 64 % using both word and link features [2]. In [13] good classification results are reported on the four-classes subset, though. Here error rates down to 20 % are obtained for the subset that we shall also consider in this report.

[2] A classification accuracy of 0.36 is obtained giving an error rate of $1 - 0.36 = 64\%$

3.2 PLSI

The objective of this section is to examine whether the PLSI model is useful in text processing tasks. This is done by implementing the PLSI model presented in section 2.1.4. First the model is trained on a synthetic data and then the trained model is analyzed in order to see if we get what we expect. After that we want to expose the model to a real data set. The MED data set (see section 3.1.1) is known from previous works and therefore it is used to compare PLSI to other models. In particular we compare PLSI with LSI (see section 2.1.2 for the theoretical background of LSI) and ICA (see section 2.1.3). Having analyzed the models, we want to let PLSI solve a classification task. That is, we compare the classification performed by PLSI with results from classification with LSI and ICA on the MED data set. The objective is not to obtain better classification performance than seen before as is often seen in the literature, but simply to show that the PLSI model works.

3.2.1 Model implementation

The PLSI model developed in 2.1.4 has been implemented in Java [16]. The input to the model is a term-document matrix, which is extracted from a data set of documents beforehand. This task is achieved using the Mole Library [20]. The training task is performed by implementation of the EM updating rules, and the output of the training phase is parameter estimates for $p(z), p(w|z)$ and $p(d|z)$. The parameters are initialized randomly.

For information retrieval and classification purposes the PLSI implementation provides means to compute the document representations, $p(z|d)$, for each document. Also the folding-in of unseen documents have been implemented according to section 2.1.4. The output is the new parameters, $p(d_{new}|z)$.

Classification schemes

A few simple classification schemes have been implemented. We will call them Nearest Neighbor, Majority label and Mixed label classification. They are all what one can call unsupervised-then-supervised classification schemes. Unsupervised learning is used to train the model, that is, we don't use the labels of the training documents. But then to decide what labels the test documents should be assigned, we use the labels from the training documents in a supervised fashion. Nearest neighbor classification simply assigns the label of the nearest neighbor in in the latent space determined by the cosine similarity measure [3] between the $p(z|d)$'s. Majority label classification

[3] The cosine similarity corresponds to the dot product between vectors such as the $p(z|d)$.

clusters all documents according to their largest value of $p(z|d)$, gives each cluster the label, which is shared by the majority of the documents in that cluster, and then assigns the new document to a cluster and assigns the majority label of that cluster to the new document. Mixed label classification is suggested here as an extension to majority label classification. Instead of letting each cluster represent one label, each cluster represents a mix of labels, $p(l|z)$, according to the documents in that cluster. That is

$$p(l|z) = \sum_{d_{\text{train}}} p(l|d)p(d|z) \tag{3.1}$$

where $p(l|d)$ is either 1 or 0 according to the true label of the document.

Now the label of a test document can be determined according to

$$\underset{l}{\text{argmax}}\Big(p\left(l|d_{test}\right)\Big) = \underset{l}{\text{argmax}} \sum_{z} p(l|z)p(z|d_{test}) \tag{3.2}$$

As opposed to Majority label classification the Mixed label classification scheme takes advantage of the presence of multiple topics within one document and therefore it is claimed to be a more suitable classification scheme than Majority label classification.

After classification it is possible to view the results by obtaining a list of the documents assigned to each class. Also it is possible to compute some statistics on the classification performance - that is a confusion matrix can be obtained and the number of correctly and incorrectly classified documents can be obtained respectively.

3.2.2 Initial analysis on a synthetic data set

Constructing the synthetic data set

A data set has been created in order to analyze the model, which is the outcome of the PLSI training procedure. In figure 3.1 the contents of the 9 documents in the data set has been visualized.

The data set has 15 different words and 9 documents. The documents are composed such that the same 5 words always co-occur. The first document only has the first 5 words, the second document only has the middle 5 words and the third document only has the last 5 words. The remaining documents have combinations of the first 3 documents.

Analyzing factor and document representations

In figure 3.2 the document representations of all the training documents can be seen. E.g. the first document has the values $p(z_1|d_1) = 1.0$, $p(z_2|d_1) = 0$ and $p(z_3|d_1) = 0$. We notice that the first 3 documents only contain one factor each, whereas the remaining documents contain different fractions of

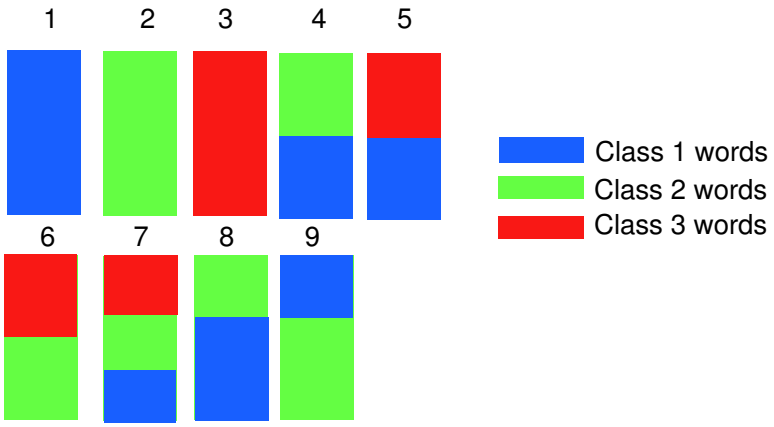

Figure 3.1: The contents of a simple synthetic data set. The data set contains 9 documents and 15 words. The 15 words are divided into 3 classes, such that the same 5 words always co-occur. The 9 documents then have different combinations of the 3 classes of words.

the three factors. This is exactly what we would expect, because that is how the data set was constructed. We notice the strength of the PLSI model here that it is capable of representing documents with a mixture of topics. The values of $p(z|d)$ are not just probabilities of belonging to clusters, rather it is perfectly fine that a document has content generated by different topics.

In figure 3.3 the representation of the 3 factors can be seen. It is seen that each factor is composed of five words and nothing else. That is also exactly what we would expect considering the way the documents were constructed.

One thing we should be aware of is that the parameters are initialized randomly and therefore it is not necessarily the case that we end up with the same parameters on every run. We therefore examine the parameter outputs of 100 runs of the training procedure. In figure 3.4 a histogram of the sum of squared distances between the found $p(z|d)$ parameters and the expected ones is shown. It can be seen that in most cases the found parameters are very close to the expected ones. Only in a few percent of the cases the $p(z|d)$ parameters are very different. In these cases the trained model has completely different factors than the ones expected. One case is the document and factor representations seen in figure 3.5. We should keep the initialization sensitivity of the model in mind when using it e.g. for classification purposes. We should also notice that this training set is very simple and therefore in a real data set we might see more different results in every run. To the left in figure 3.4 the deviation has been plotted vs. the log likelihood (LL) of the model trained on the training data. It is obvious that there is a relationship between the LL on the data and the factors estimated. In this experiment we could therefore use the LL as an indication of whether

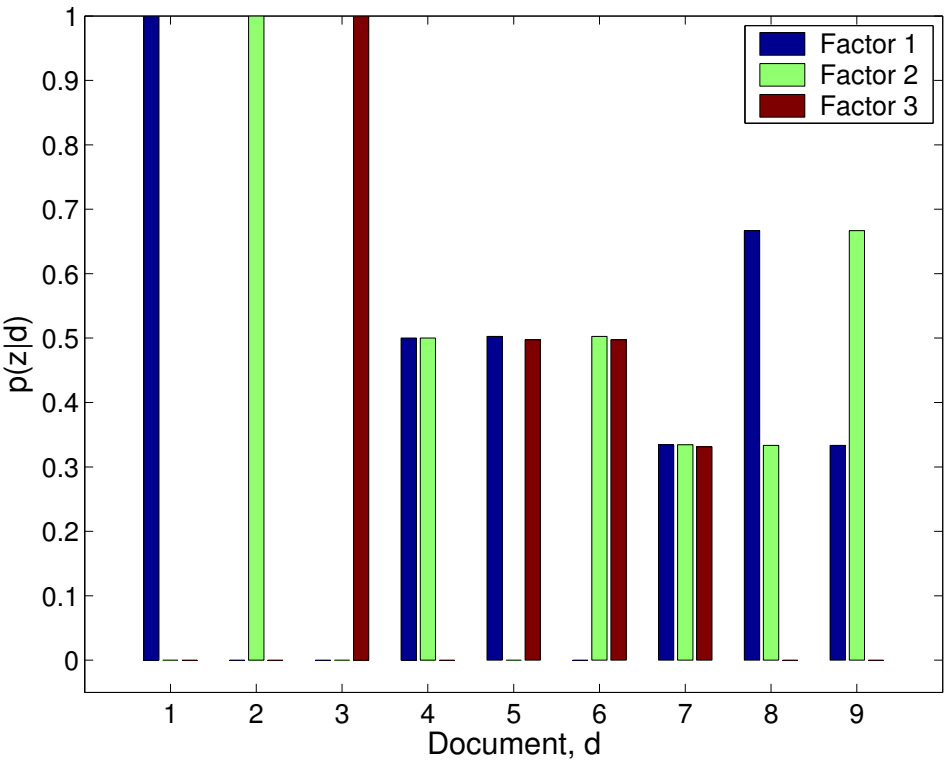

Figure 3.2: The PLSI model has been trained on a synthetic data set. The values of $p(z|d)$ for the 3 factors for the 9 documents are shown. To each document along the x-axis belongs 3 bars, and each bar represents the value of $p(z|d)$ for one of the 3 factors. The mixture of factors for each document is as expected according to the mixture of topics in the constructed documents.

we have found the "true" factors.

Analyzing the folding-in of new documents procedure

During the training phase the PLSI model estimates the document representations for the training documents, $p(z|d_{train})$. But when new documents are to be analyzed they have to be folded-in in order to obtain the document representations for the test documents, $p(z|d_{test})$, as described in section 2.1.4.

To test the implementation of the folding-in of documents procedure we first try to fold-in the training documents themselves, because we know that they should get the same representation as in the trained model, that is $p(z|d_{test}) = p(z|d_{train})$. To the left in figure 3.6 one case of the folded-in training documents can be seen. Comparing with figure 3.2 we conclude that they are almost identical. To the right in figure 3.6 statistics on the deviation from the model representation has been visualized. It is noted that

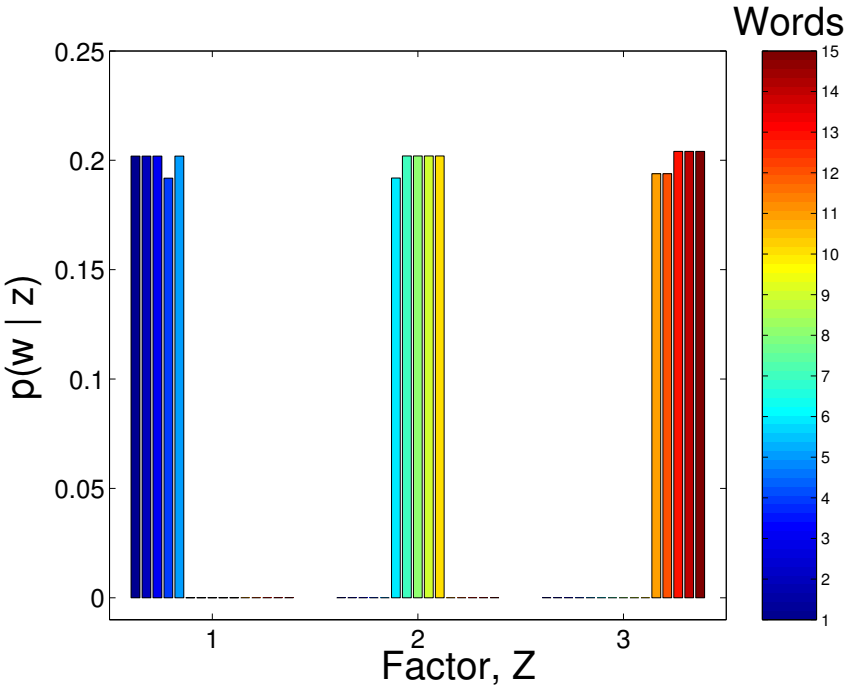

Figure 3.3: The PLSI model has been trained on a synthetic data set. The values of $p(w|z)$ for the 3 factors for all words are shown. To each factor belongs 15 bars, each representing the value of $p(w|z)$ for each of the 15 words. Most of the values are 0, though, such that each factor only has 5 bars. That is because each factor is specialized to only representing 5 words each as we would expect.

only in very few percent of the cases we fail to find the same representations.

In order to test the fold-in procedure further we fold-in a new unseen document. This test document is composed of the first five words and the last five words repeated once such that it contains 15 words in total. In figure 3.7 the result of the fold-in can be seen. As we would expect the document is represented by approximately $\frac{1}{3}$ of the first factor and $\frac{2}{3}$ of the third factor.

In conclusion the fold-in of test documents seem to work on this simple test set with no noise, though we also notice that the folding-in procedure is sensitive to the random initialization of the parameters and therefore occasionally end up with wrong estimates for the new documents.

3.2.3 Results on the MED data set

In order to test the PLSI model on real data, in this section we shall apply it on the MED data set. The MED data set consists of 124 documents, which has been labeled manually with five different class labels (see section 3.1.1).

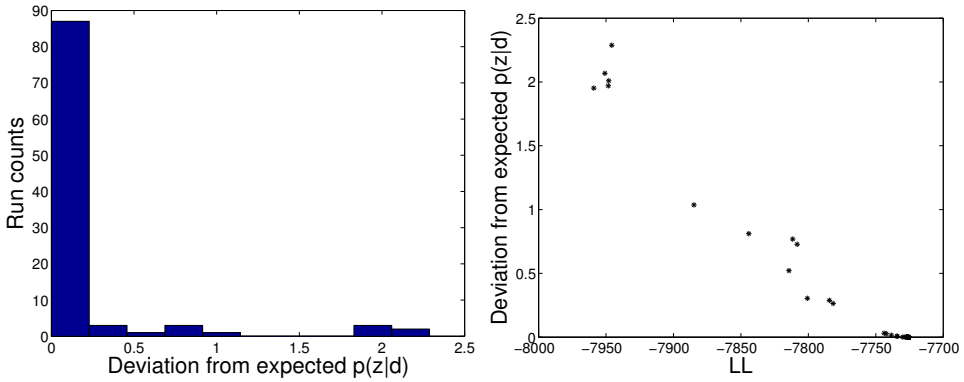

Figure 3.4: To the left a histogram of the deviation from the "true" $p(z|d)$. The histogram shows that we don't always get the document representations, $p(z|d)$, for the synthetic data set which we would expect and which are shown in figure 3.2. To the right the deviation from the expected document representations vs. the LL of the model document representations is shown. The relationship between the LL of the model and the correctness of the document representations is obvious.

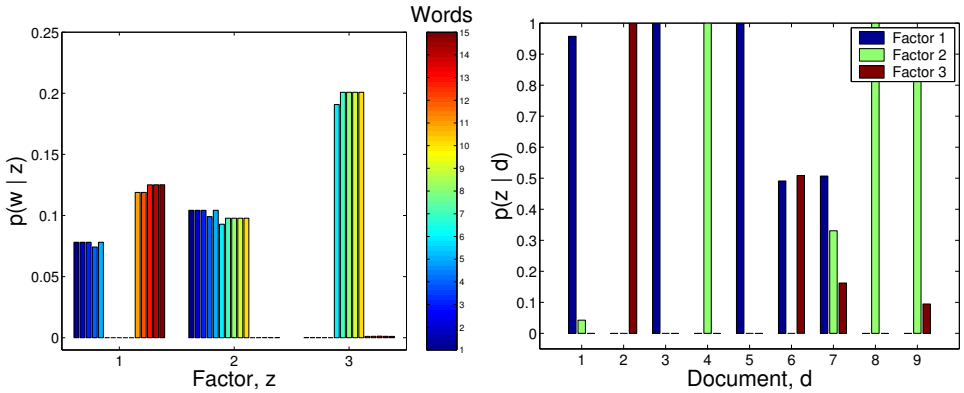

Figure 3.5: Factor representations, $p(w|z)$, (left) and Document representations, $p(z|d)$, (right) different from the ones expected for the PLSI model trained on a synthetic data set. In this case the factors are not specialized. Factor 1 and factor 2 have significant values, $p(w|z)$, for 10 different words and therefore the factors overlap. As a consequence the document representations, $p(z|d)$, become completely different than the ones shown in figure 3.2.

First we will analyze the training procedure for PLSI on this more complicated data set compared to the synthetic one used in the last section. Then the document representations will be compared to LSI and ICA. Second the data set will be classified with the three models and the results compared.

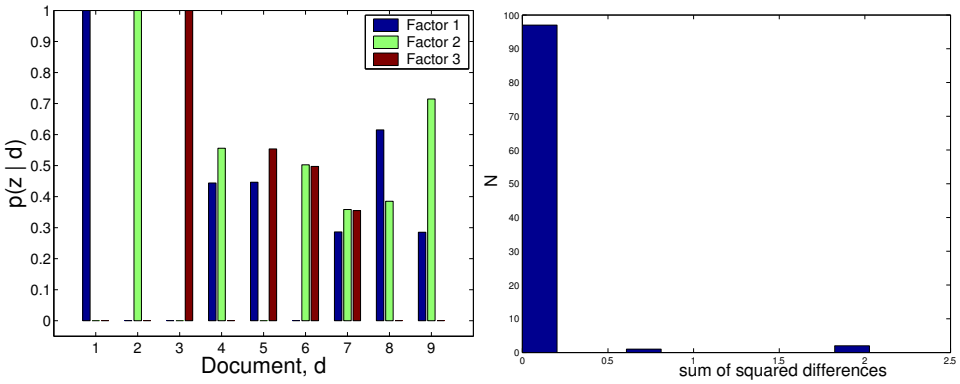

Figure 3.6: Results of folding-in training documents on a synthetic data set. To the left the new $p(z|d_{test})$ and to the right the histogram of the deviation from the original $p(z|d_{train})$'s. It is shown that folding-in the training documents gives almost identical document representations as the ones estimated by the model in the first case.

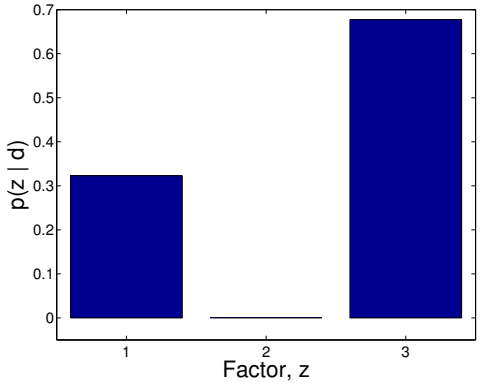

Figure 3.7: The PLSI model has been trained on a synthetic data set and a test document has been folded-in. The document representation, $p(z|d_{test})$ of the test document is shown. Because the document contains 1 x the first 5 words and 2 x the last five words, the document representation is approximately $\frac{1}{3}$ of the first factor and $\frac{2}{3}$ of the last factor as we would expect.

Training the model

First the data set has been divided randomly into a test and a training set - each set with 62 documents. The PLSI model has been trained on the training documents and the test documents are folded-in according to the procedure described previously. Figure 3.8 shows that the EM training procedure converges both on the training and on the test set. In order to compute log likelihood (LL) on the test set we had to compute the $p(w, d_{test})$ parameters according to equation 2.11, which is not straight forward, because the model doesn't know the $p(d_{test}|z)$ parameters. Therefore the test

documents had to be folded-in on every EM iteration in order to compute LL on the test set.

Looking at figure 3.8 we notice that the LL is larger on the test set than on the training set. The reason is that the new words in the test documents are not folded-in and therefore there are fewer $p(w, d)$ parameters to use for LL calculation on the test set. This gives a larger LL, since each observation contributes negatively to the LL. It is interesting to notice that no overfitting takes place, since the LL on the test set does not deteriorate. This is in contrast with [14], who introduces a tempered EM training procedure to avoid overfitting. And also [2] criticizes PLSI for the risk of overfitting.

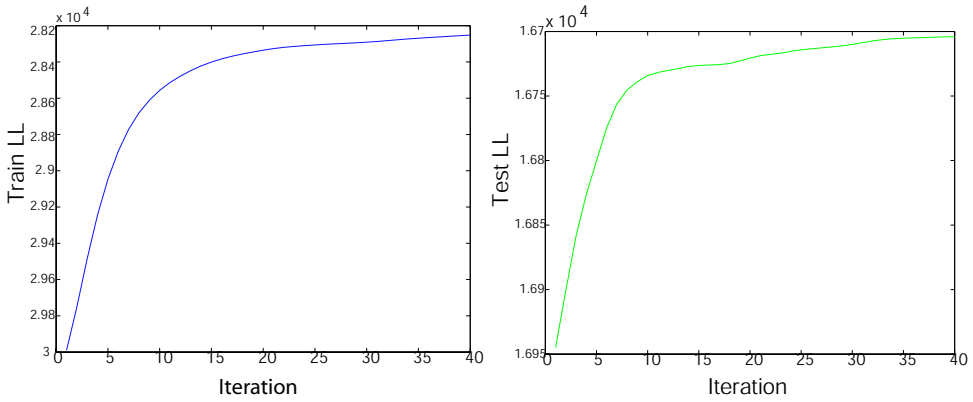

Figure 3.8: Convergence of the PLSI EM training algorithm on the MED data set. The figure shows that the training procedure converges both on the training set and on the test set. Also no overfitting takes place, since the LL on the test set keeps on increasing.

In figure 3.9 the training document representations for the PLSI model trained on the MED data has been visualized by plotting $p(z_1|d)$ against $p(z_2|d)$ and so forth. One dot represents one document in factor space and each document has been colored according to its true class. The plots indicate that documents assigned to the same class tend to have high values for the same factor, which is also what we would hope for. In figure 3.10 the similar plot is seen for the folded-in test documents. Here too, the plot indicates that documents from the same class have similar document representations in the latent PLSI space, even though the documents tend to have a greater mix of topics than with the training documents.

LSI and ICA models have been trained on the same data set and in figure 3.11 and 3.12 the document representations for these models have been visualized. It is clear that the two models provide some means to discriminating among the true classes. Also it seems easier to discriminate than with the PLSI model.

Summing up on the trained PLSI model, the EM training converges both on training and test data, and it is clear that it does find some underlying

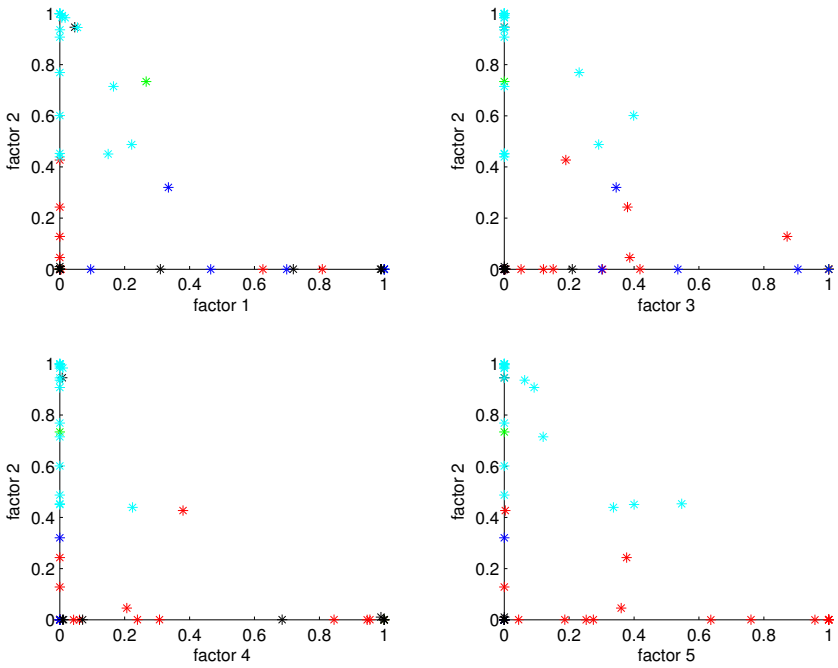

Figure 3.9: PLSI document representations $p(z|d_{train})$ for training documents. The PLSI model has been trained on the MED data set. One dot represents one document and the documents have been colored according to their true class. The figure shows that documents from the same class tend to have high values for the same factors. E.g. the documents from the red colored class tend to have high values for factor 4 and 5.

meaning in the MED data set, since the document representations of similar documents according to the true label are also similar.

Classification

In this section we shall apply the PLSI model as a classifier on the MED data set. It is compared to LSI and ICA as classifiers. A subgoal is to evaluate and compare the three classification schemes introduced in section 3.2.1.

The results of classifying the MED data set in a particular run with PLSI, LSI and ICA can be seen in figure 3.13. The Majority label classification scheme has been used for all models. LSI and ICA have only been used to classify the training data. For PLSI the model parameters, $p(z|d)$, have been used for classification. It is seen that the error rate of PLSI is 39% which is in between LSI and ICA with error rates of 58% and 18% respectively. We therefore conclude that PLSI is comparable to other classification models even though it is desirable with statistics from more runs.

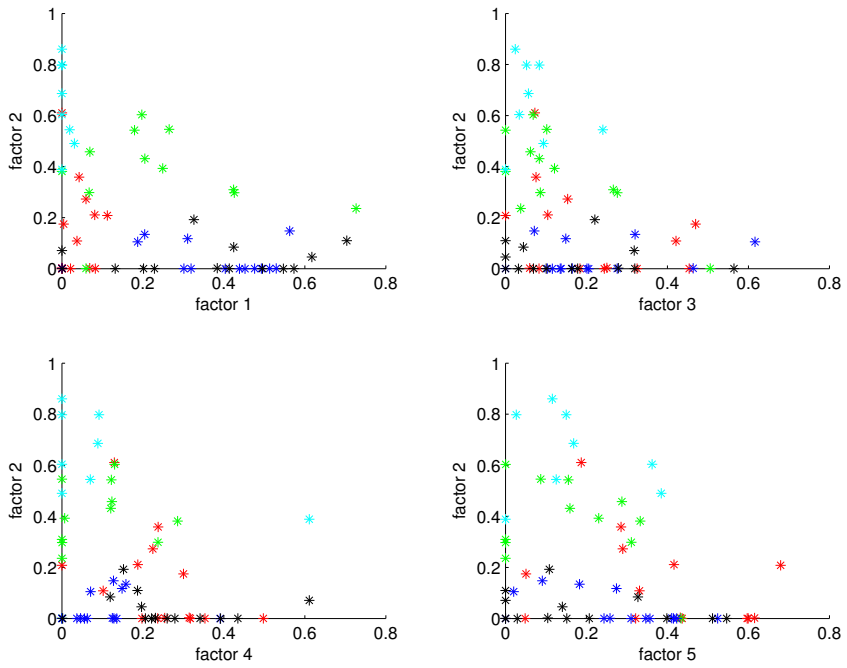

Figure 3.10: PLSI document representations $p(z|d_{test})$ for test documents. The PLSI model has been trained on the MED data set. One dot represents one document and the documents have been colored according to their true class. The figure shows that the test documents have a larger mix of factors than the training documents, but the documents from the same class still tend to have high values for the same factors. E.g. the cyan colored class tend to have high values for factor 2.

Classification scheme	Train error	Test error
Nearest Neighbor	4.0%	51%
Majority Label	40%	48%
Mixed Label	36%	37%

Table 3.1: Error rate average from 10 runs of classification on the MED data set for 3 different classification schemes. The Mixed Label classification scheme is shown to perform better than the others on the test data and is therefore our preferable classification scheme.

In order to get more statistics on classifying with the PLSI model and evaluating the Mixed label classification scheme the average error rates of classifying with the three classification schemes have been collected in table 3.1. Nearest neighbor classification is by far the best one on training data, but this is quite obvious, because each training document is given the label of the nearest neighbor in the latent semantic space, which is obviously itself, and therefore it gets its own true label. On test data the Mixed

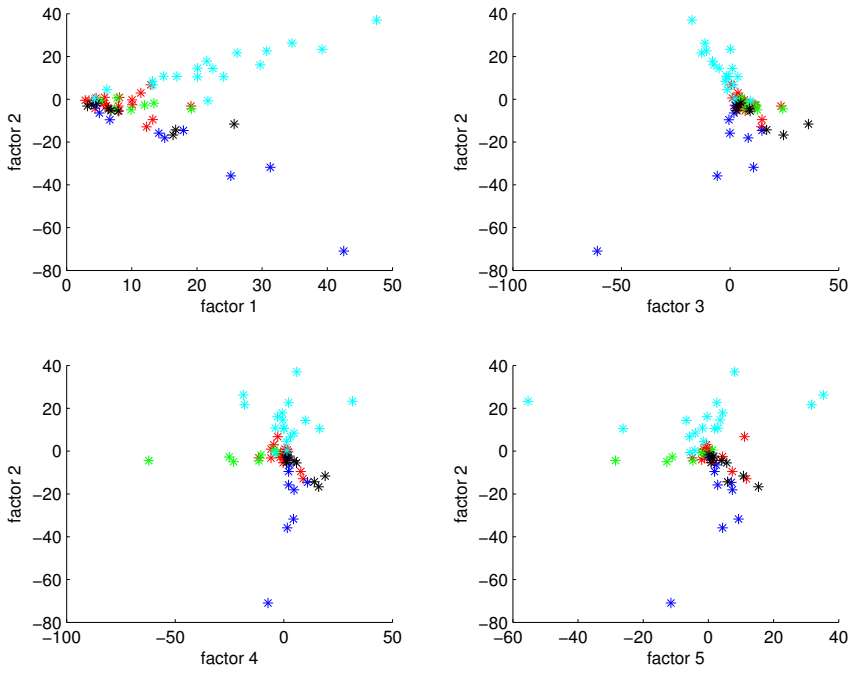

Figure 3.11: LSI document representations for the LSI model trained on the MED data set. The values plotted are the rows of the matrix product $S \cdot V^T$ according to equation 2.3. The figure shows that the documents from the same class tend to group together making it possible to discriminate among documents from different classes.

label classification scheme outperforms the others, and therefore confirms the assumption that this is a more suitable classification method for the PLSI classifier.

In conclusion the PLSI model is capable of classifying on a real data set. The best error rate, which we obtain here, is 36 %. Compared to the error rate of 8 % reported in section 3.1.1 obtained using ICA on the same data set this is not too impressive. Even though the error rate is not impressive, it is still far better than average, which would be 80% for the five classes in the MED data set

Initialization sensitivity We have seen that the PLSI model does capture some meaning in the data, but we have also seen that it is sensitive to the random initialization of the parameters. The resulting model parameters are not the same on every run, and they also perform differently in the classification task. In order to get more consistent results we could train the model a number of times and then pick the best one. What we need is a measure of performance of the model so that we know which model to choose. One could suggest that the model that does the best classification

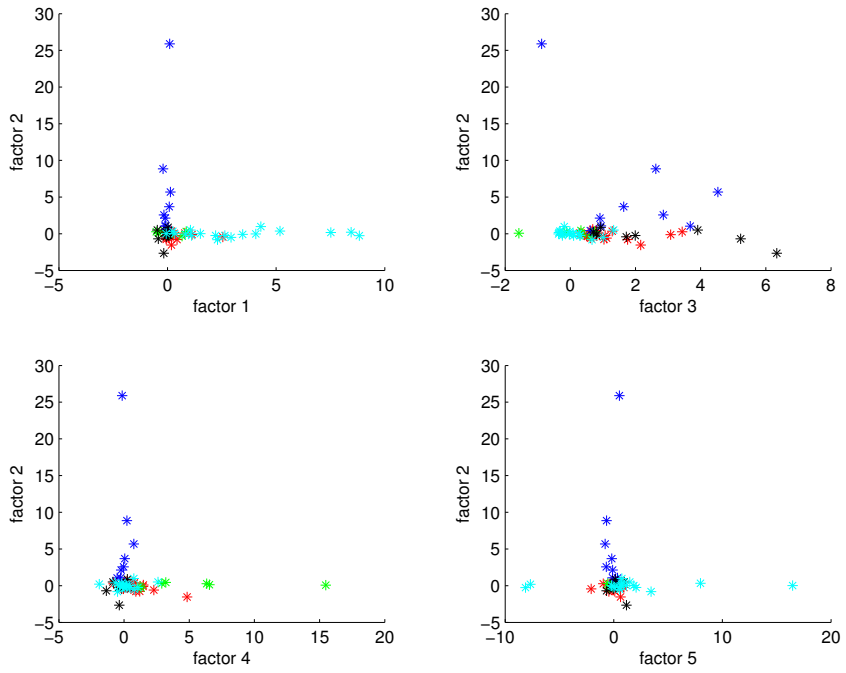

Figure 3.12: ICA document representations for the MED data set. The plotted vales are the rows of the matrix S according to equation 2.6. The figure shows that ICA clearly separates the documents according to their classes such that documents from the same class are aligned along one particular component.

on data should be chosen, but since that also depends on the classification scheme this is not a preferable solution. Instead we can use the LL of data as a measure of performance and then choose the best model as the one with the largest LL. To justify the feeling that this is a good choice, in figure 3.14 the LL has been plotted vs. the classification error rate on the test set for 50 different trained models. It can be seen that it is not always the case that a model with greater LL is also a better classifier, but there seems to be a tendency.

To be sure about the relationship between LL and classification error rate a statistical test can be performed. A regression model can be estimated such that

$$E_i = \alpha + \beta \left(LL_i - \overline{LL} \right) \tag{3.3}$$

where E_i and LL_i is the error rate and LL of the i'th experiment respectively.

The hypothesis that there is no relation between error rate and LL corresponds to testing whether $\beta = 0$. Under the assumption that $\beta = 0$ a test quantity, t, can be computed, which is distributed according to an F

```
****Training documents classification****

PLSI
        class1  class2  class3  class4  class5  Total
class1  12      0       0       0       1       13
class2  4       8       0       0       0       12
class3  2       0       6       0       1       9
class4  1       6       4       0       4       15
class5  1       0       0       0       12      13

Summary:
Number of correctly classified documents: 38
Errors: 24
Error rate: 0.39

LSI
        class1  Class2  Class3  class4  class5  Total
class1  0       0       0       13      0       13
class2  0       0       0       12      0       12
class3  0       4       5       0       9
class4  0       0       1       14      0       15
class5  0       0       0       5       8       13

Summary:
Number of correctly classified documents: 26
Errors: 36
Error rate: 0.58

ICA
        class1  class2  class3  class4  class5  Total
class1  12      0       0       0       1       13
class2  0       12      0       0       0       12
class3  0       0       0       9       0       9
class4  0       0       0       15      0       15
class5  0       0       0       1       12      13

Summary:
Number of correctly classified documents: 51
Errors: 11
Error rate: 0.18

****Test documents classification****

PLSI
        class1  class2  class3  class4  class5  Total
class1  16      7       0       0       1       24
class2  0       1       3       0       0       4
class3  2       3       6       0       2       13
class4  1       3       1       0       3       8
class5  1       0       0       0       12      13

Summary:
Number of correctly classified documents: 35
Errors: 27
Error rate: 0.44
```

Figure 3.13: Confusion matrix results on the MED data from one run of classification. The confusion matrices for the training data is shown for PLSI, LSI and ICA. The confusion matrix for the test data is shown for PLSI only. The rows in the matrices indicate the true class of the documents and the columns indicate the class predicted by the models. Majority label classification has been used for all models.

distribution (see [5]):

$$\hat{\beta} = \frac{SOP_{E,LL}}{SS_{LL}} = \frac{\sum_i (E_i - \overline{E})(LL_i - \overline{LL})}{\sum_i (LL_i - \overline{LL})^2} = -2.99 \cdot 10^{-4} \tag{3.4}$$

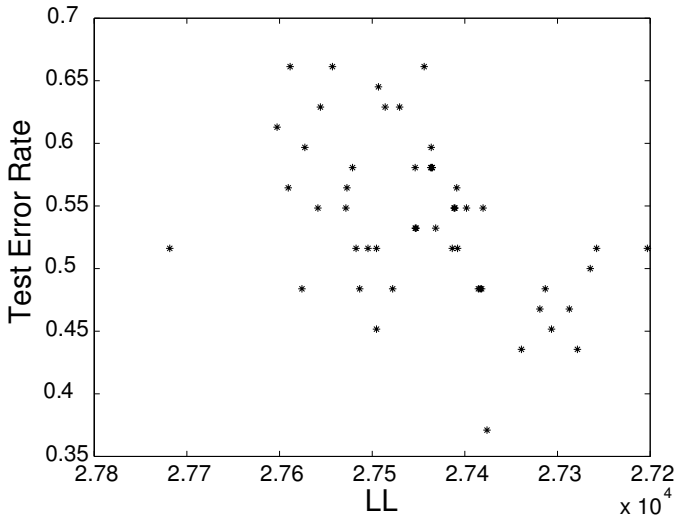

Figure 3.14: The test classification error rate on the MED data set using the Mixed Label classification scheme vs. the LL of the test data. The figure indicates a correlation between the LL on training data and the classification error rate on a test set. This means that LL can be used as a way to choose between a number of trained models and thus increasing performance and reducing initialization sensitivity.

where SOP $\equiv$ Sum Of Products and SS $\equiv$ Sum of Squares

$$t = \frac{\hat{\beta}^2 \sum_i (LL_i - \overline{LL})^2}{\left(\sum_i \left(E_i - \overline{E} - \hat{\beta} \left(LL_i - \overline{LL} \right) \right)^2 \right) / (N-2)} = 14.13 \tag{3.5}$$

If we want to test on a 5% significance level, we should compare the test quantity, t, with a value in the F distribution equal to

$$F(1, N-2)_{0.95} = F(1, 48)_{0.95} = 4.03 \tag{3.6}$$

Because $t > F(1, N-2)_{0.95}$ we have to reject the hypothesis that $\beta = 0$ and therefore the relation between LL and classification error rate is statistically significant.

In conclusion it is possible to use LL on training data as a measure of how good the model is that we have just trained. Thus if we want to reduce the initialization sensitivity we can train a number of models and then choose the one with the largest LL.

3.3 Supervised PLSI

In this section the PLSI model is extended to take advantage of labeled documents in a classification task. The assumptions of the model is that an underlying factor is responsible of generating words, documents and labels. The classification approach used in section 3.2 was a so called unsupervised-then-supervised approach, because the labels were not considered when training the model, but first considered when assigning labels to new documents. This is what we want to change in this section. The extension can also be seen as an extension to the Mixed Label Classification scheme introduced in section 3.2.1, which was one of the unsupervised-then-supervised classification schemes. As with Mixed Label Classification, documents will be classified according to

$$\underset{l}{\operatorname{argmax}}\Big(p\left(l|d\right)\Big) = \underset{l}{\operatorname{argmax}} \sum_{z} p(l|z)p(z|d) \tag{3.7}$$

but now the parameters $p(l|z)$ will not be estimated by the labels of the training documents according to equation 3.1. Instead $p(l|z)$ will explicitly take part in the model parameters.

3.3.1 A joint probalistic model

As opposed to the approach taken by PLINK described in section 2.1.6, no weighting between the words and labels is introduced. The observations are composed of word-label-document triples, (w, l, d). By virtue of the conditional independence of the words, labels and documents given the latent variable, z, the probability of one observation is given by

$$p(w, l, d) = \sum_{z} p(w|z)p(l|z)p(d|z)p(z) \tag{3.8}$$

where $p(w|z), p(l|z), p(d|z)$ and $p(z)$ are the parameters of our extended model. It is noticed that there is one extra set of parameters compared to the basic PLSI model, that is, the $p(l|z)$ parameters.

Estimation of model parameters

In this section we shall derive the parameter updating rules by using the EM algorithm. The likelihood of the entire data set, X, becomes

$$p(X|model) = \prod_{d} \prod_{l} \prod_{w} p(w, l, d)^{n(w,l,d)} \tag{3.9}$$

And the log likelihood of the data set is then

$$\begin{aligned}
\ln p(X|model) &= \sum_d \sum_l \sum_w n(w,l,d) \ln p(w,l,d) \\
&= \sum_d \sum_l \sum_w n(w,l,d) \ln \sum_z p(w|z)p(l|z)p(d|z)p(z) \frac{p(z|w,l,d)}{p(z|w,l,d)} \\
&\geq \sum_d \sum_l \sum_w n(w,l,d) \sum_z p(z|w,l,d) \ln \frac{p(w|z)p(l|z)p(d|z)p(z)}{p(z|w,l,d)} \\
&= F(\theta, p(z|w,l,d))
\end{aligned} \tag{3.10}$$

where the mixing proportions, $p(z|w,l,d)$ have been introduced in order to take advantage of Jensen's inequality. Now Lagrange multipliers, Λ, are introduced in order to satisfy the constraints that all the probabilities have to sum to one.

$$\begin{aligned}
L(\Lambda, \theta, p(z|w,l,d)) = F(\theta, p(z|w,l,d)) &+ \sum_z \lambda_z (\sum_w p(w|z) - 1) \\
&+ \sum_z \mu_z (\sum_d p(d|z) - 1) + \gamma (\sum_z p(z) - 1) \\
&+ \sum_{w,l,d} \rho_{w,l,d} (\sum_z p(z|w,l,d) - 1) + \sum_z \tau_z (\sum_l p(l|z) - 1)
\end{aligned} \tag{3.11}$$

In order to find the updating rules for the parameters, L is differentiated with respect to the mixing proportions, $p(z|w,l,d)$, and with respect to the parameters, θ, respectively. The derivation is analog to the derivation of the updating rules for PLSI in section 2.1.4, so we will not go into any more details here, but instead just state the updating rules. The mixing proportions are computed in the E-step according to

$$p(z|w,l,d) = \frac{p(w|z)p(l|z)p(d|z)p(z)}{\sum_z p(w|z)p(l|z)p(d|z)p(z)} \tag{3.12}$$

And the updating rules for the parameters are computed in the M-step according to

$$\begin{aligned} p(w|z) &= \frac{\sum_d \sum_l n(w,l,d)p(z|w,l,d)}{\sum_d \sum_l \sum_w n(w,l,d)p(z|w,l,d)} \\ p(l|z) &= \frac{\sum_d \sum_w n(w,l,d)p(z|w,l,d)}{\sum_d \sum_l \sum_w n(w,l,d)p(z|w,l,d)} \\ p(d|z) &= \frac{\sum_l \sum_w n(w,l,d)p(z|w,l,d)}{\sum_d \sum_l \sum_w n(w,l,d)p(z|w,l,d)} \\ p(z) &= \frac{\sum_d \sum_l \sum_w n(w,l,d)p(z|w,l,d)}{\sum_d \sum_l \sum_w n(w,l,d)} \end{aligned} \tag{3.13}$$

Notice that introducing labels into the model is actually just like introducing another feature of the documents. The labels don't play any special role compared to the words of the documents. Therefore the derivation of the EM updating rules for supervised PLSI would work for any new feature of the documents introduced.

3.3.2 Implementing Supervised PLSI

The supervised PLSI model developed here has been implemented in Java [16]. When training the model the input is the term-document matrix and a label-document vector, and the output of the training phase is parameter estimates for $p(w|z), p(l|z), p(d|z)$ and $p(z)$. For classification purposes, according to equation 3.7, we need the $p(l|z)$ parameters and the document representations, $p(z|d)$. In order to compute the document representations we cannot fold-in new documents with the EM updating rules derived in section 3.3.1, because the new documents don't have labels. Therefore new documents are folded-in according to the updating rules for the basic PLSI model as described in section 2.1.4 where only iteration between updating $p(z|w,d)$ and $p(d|z)$ is performed.

3.3.3 Results

First, it is investigated whether the training procedure presented here converges. In figure 3.15 it is shown that this is the case. In particular the training procedure converges on the test set, even though no tempered training procedure has been used as is done by Hofmann in [14] to avoid overfitting.

Next thing to investigate is the parameter estimates of the trained model. The model has been trained on the MED data set, and now we want to see if

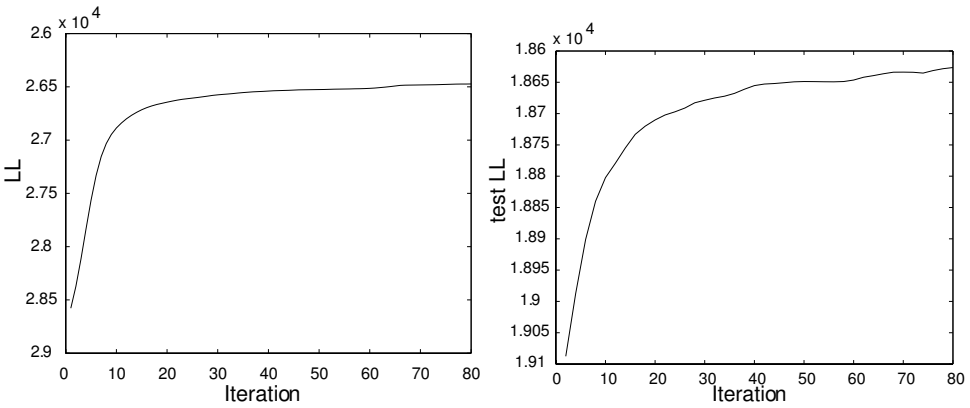

Figure 3.15: Convergence of the supervised PLSI model on training and test set. The model has been trained on the MED data set (see section 3.1.1). The figure shows that the supervised training procedure converges both on the training set and the test set. Furthermore no overfitting takes place.

the parameters of the model seem reasonable. The number of factors chosen is eight, and the number of classes is five.

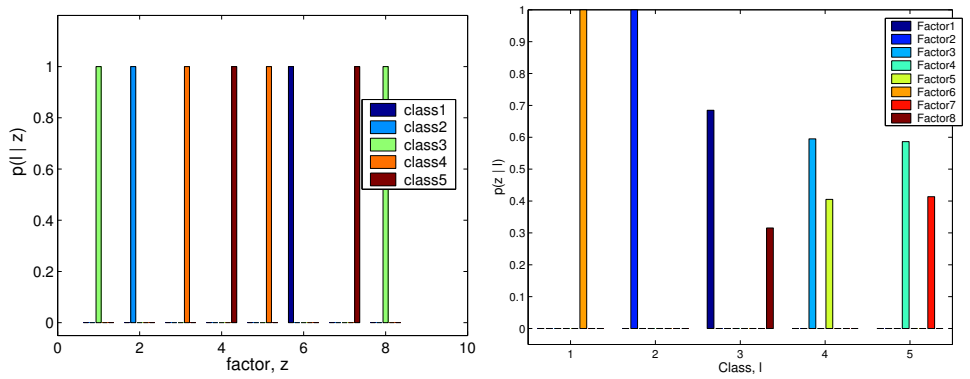

Figure 3.16: The relation between factors and classes for the supervised PLSI model using 8 factors. The left graph shows that each factor belongs to exactly one class. The right graph shows that some classes contain a mix of two factors whereas others only contain one factor. E.g. class 3 contains a mix of factor 1 and factor 8.

First we investigate the relation between factors and classes. To the left in figure 3.16 we can see that each factor belongs to exactly one class. And to the right in figure 3.16 we can see that some classes contain exactly one factor and some classes contain a mix of two factors. If we interpret the factors as topics present in the documents it seems reasonable that the documents in one class can deal with different subtopics. This could indicate that some of the classes could be further divided into subclasses, but by analyzing the document representations, $p(z|d)$, which we shall do shortly, it is possible to

see that this is not the case. On the other hand we can say already that the same topic is not present in documents belonging to different classes. E.g. one could imagine that class 1 dealt with documents discussing topic 1 and 2, and class 2 dealt with documents discussing topic 2 and 3, but that is not the case, because no topic has more than one class.

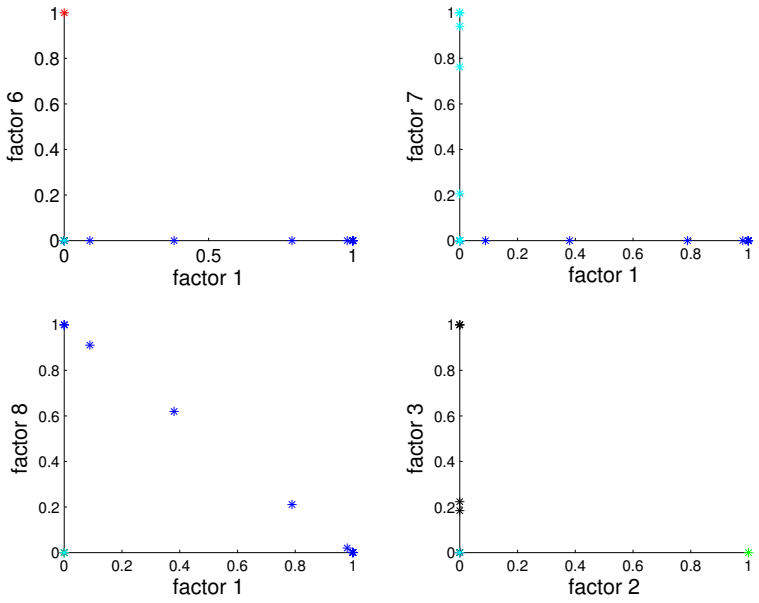

Figure 3.17: The training document representations, $p(z|d_{train})$, for the supervised PLSI model trained on the MED data set. Each color corresponds to one class. The figure shows that there is a very strong separation of the classes. Also it shows that some documents are a mix of two factors. E.g. the documents in class 3 is a mix of factor 1 and factor 8.

In figure 3.17 some of the $p(z|d)$ parameters for the training documents have been visualized. First we notice that there is a very strong separation of the classes and that most training documents strictly belong to one factor. Second we notice by looking at factor 1 vs. factor 8 that the class with blue dots tend to be a mix of factor 1 and factor 8. By comparing with the plot of $p(z|l_3)$ in figure 3.16, which shows that class 3 contains a mix of factor 1 and factor 8, we conclude that class 3 contains documents that have a mix of subtopics within the same document. That is the reason why the classes cannot be further divided into subclasses, because each document cannot belong to more than one class. This also shows a strength of the PLSI model, which is that documents can belong to a mix of factors and not only to one cluster.

In figure 3.18 some of the $p(z|d_{test})$ parameters for the test documents have been plotted. Again there is a good separation of the classes even though the test documents tend to have a larger mix of factors than the

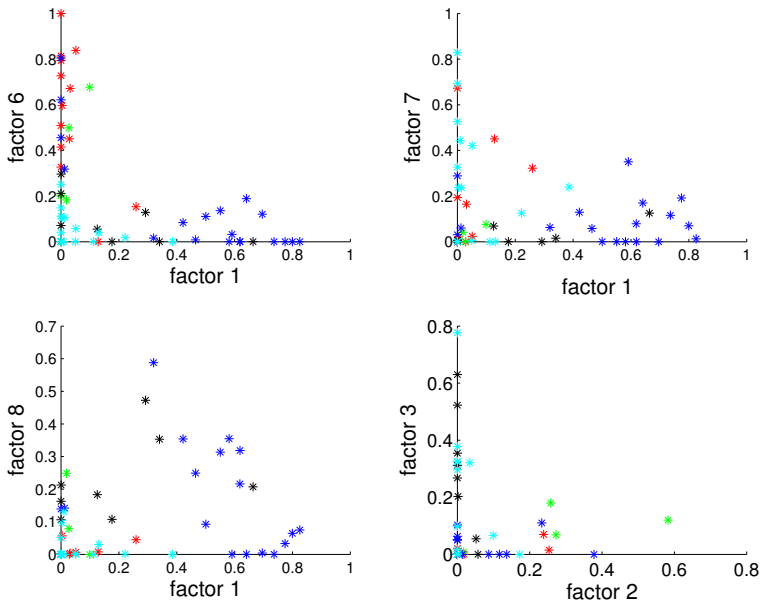

Figure 3.18: The test document representations, $p(z|d_{test})$, for the supervised PLSI model trained on the MED data set. The figure shows that the test documents are not as easy to separate as the training documents, but that there is still good separation.

training documents.

Classification and comparison with PLSI

In this section we shall explore the performance of the supervised PLSI classifier. Classification results on the MED data set as well as the WebKB data set will be presented. Finally we will try to answer the question whether the supervised version of PLSI is a better classifier than the basic PLSI model, because that was our motivation for extending PLSI to performing supervised learning.

In section 2.2 we examined how to compare two classifiers in order to test whether one is better than the other. The major problem with the statistical tests for such comparisons is that the assumptions of the tests are not fully fulfilled. In the k-fold cross-validation technique we measure the difference in classification error rates for the two classifiers in each run. The assumption is that this difference is normally distributed. Before doing the actual comparison between supervised and non-supervised PLSI we shall first examine to what extent this quantity is normally distributed.

Figure 3.19 shows a histogram of the error rates. This figure could indicate that the error rate is normally distributed. Figure 3.20 then shows that this is actually not the case, because the data points don't follow the

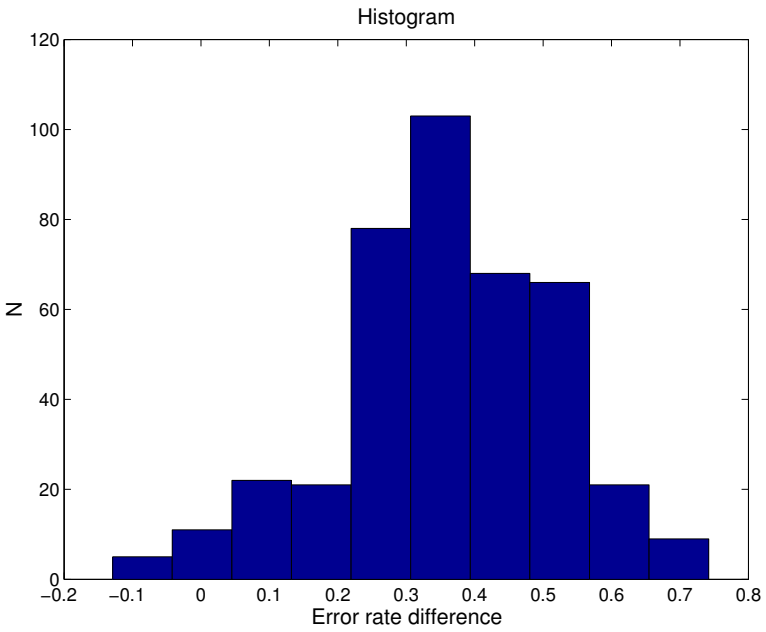

Figure 3.19: The histogram of the differences between PLSI supervised and non-supervised classification error rates on the MED data set. The figure shows that the error rates could be normally distributed.

straight line. In particular there are too few low error rates.

It is also possible to perform a statistical test of whether the error rate is normally distributed. One such test is Lilliefors test for goodness of fit to a normal distribution [21]. Such a test can be performed by Matlab. The hypothesis is that the data points are normally distributed. Matlab gives the following results when tested at a 5 % level of significance:

$$\text{Test value} = 0.0693 \qquad \text{Cutoff value} = 0.0441 \tag{3.14}$$

Since the test value is larger than the cutoff value we have to reject the hypothesis that the error rate is normally distributed. This is not too surprising, since we knew from the discussion in section 2.2 that this was a problem with the test quantity, since the error rates in each run are not really independent. On the other hand figure 3.20 shows that the error rate is not very far from being normally distributed, so we can just hope that the normal distribution assumption of the test quantity is not too strict. That is, that it is not too important whether the error rate is entirely normally distributed.

A 4-fold cross-validation experiment has been performed on the MED data set. Altogether there are 124 documents, so in each trial the model is trained on 93 documents and the error rate is measured on 31 test doc-

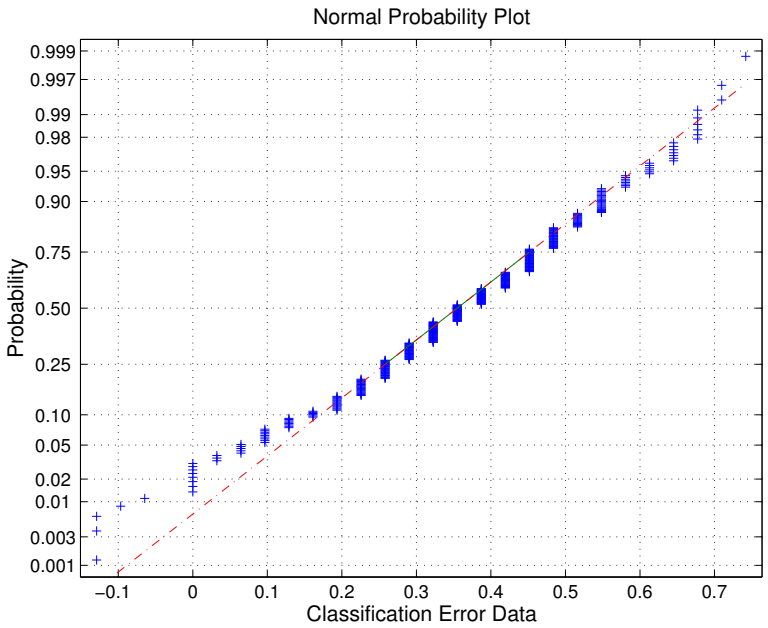

Figure 3.20: A graphical test of whether the difference between PLSI supervised and non-supervised error rates on the MED data set are normally distributed. The graph plots the experimental distribution vs. the normal distribution. In the case where the experimental distribution is actually a normal distribution this results in the same x and y values for every data point. Therefore the data points should be close to the straight line. The figure shows that the error rates are not quite normally distributed. In particular there are too few low error rates according to the normal distribution.

uments. In this experiment 8 factors have been used. The Mixed Label Classification scheme has been used for the basic PLSI model. In table 3.2 the results can be seen. The confidence intervals are computed according to equation 2.39, and they are

$$p_{plsi} = 0.40 \pm 0.28 \tag{3.15}$$

$$p_{supervised} = 0.20 \pm 0.14 \tag{3.16}$$

$$p_{plsi} - p_{supervised} = 0.19 \pm 0.17 \tag{3.17}$$

The variance of the error rates are rather large, which shows up in large confidence intervals. The reason for this is that the variance is caused by the internal randomness in the learning algorithm and the variance in the training and test data. Even though the variance is large we can still conclude that the superiority of the supervised PLSI classifier compared to the basic PLSI model is statistically significant, because 0 is not included in the 95% confidence interval for $p_{plsi} - p_{supervised}$.

Model/Trial	1	2	3	4
p_{plsi}	0,45	0,16	0,58	0,39
$p_{supervised}$	0,29	0,097	0,26	0,16
$p_{plsi} - p_{supervised}$	0,16	0,065	0,32	0,22

Table 3.2: Classification error results on the MED data set for the basic PLSI model and the supervised PLSI model based on a 4-fold cross-validation setup. Each value denotes the error rate for one trial.

Model/Trial	1	2	3	4	5	6	7	8
p_{plsi}	0.7	0.53	0.47	0.47	0.33	0.47	0.53	0.6
$p_{supervised}$	0.63	0.43	0.3	0.27	0.37	0.47	0.23	0.37
$p_{plsi} - p_{supervised}$	0.07	0.01	0.17	0.2	-0.03	0.0	0.3	0.23

Table 3.3: Classification results on the WebKB data set for the basic PLSI model and the supervised PLSI model based on an 8-fold cross-validation setup in which 240 documents are classified.

The WebKB data set has been classified with cross validation in the same manner as with the MED data set. With the WebKB data set we have to choose a subset of the entire data set, because the available computer resources cannot handle the entire data set. 240 documents are chosen randomly and a 8-fold cross validation is performed. So in each trial the models are trained on 210 documents and tested on the remaining 30 documents. In this experiment 8 factors have been used and no parsing of the HTML text has taken place. That is, the HTML tags are kept in the documents. This approach tends to achieve the best results. In table 3.3 the results can be seen. And the 95% confidence intervals of the error rates are

$$p_{plsi} = 0.51 \pm 0.091 \tag{3.18}$$

$$p_{supervised} = 0.38 \pm 0.11 \tag{3.19}$$

$$p_{plsi} - p_{supervised} = 0.13 \pm 0.097 \tag{3.20}$$

The error rates are larger than on the MED data set, but again we can conclude that the performance of the supervised PLSI model is better tested on a 5% level of significance, because 0 is not included in the 95% confidence interval for $p_{plsi} - p_{supervised}$.

In this section we obtained an error rate of 20 % on the MED data set using the supervised PLSI. This is not very impressive compared to the 8 % reported in section 3.1.1 using ICA on the same data set. On the WebKB data set we obtained an error rate of 38 % using supervised PLSI. Compared to the error rate of 20 % reported in section 3.1.2 obtained on the same data set this is not very impressive either. But compared to Hofmann and Cohn [7] who obtained an error rate of 64 % as described in section 3.1.2 using

PLSI our results seem good. Hofmann and Cohn used a larger fraction of the WebKB data set, though.

3.4 PLINK

In this section the basic PLSI model is extended to consider not only the words of a document but also links to other documents. This assumes that we are working with a data set where documents have some kind of connections. Two examples of such data sets are HTML documents, which are interconnected through hyperlinks, and scientific articles, which are interconnected through citations. The PLINK model was introduced by Hofmann and Cohn [7] as described in section 2.1.6.

3.4.1 Choosing the form of the PLINK model

There is more than one way to incorporate links into a statistical model. Here we are restricted to use the PLSI framework and extend it with link features, but other statistical models are certainly possible. E.g. the Latent Dirichlet Allocation model described in section 2.1.5.

But there is also more than one way to extend the PLSI model to incorporate link features. One way is the one suggested by Hofmann and Cohn [7], in which the observations are normalized word-document pairs (w, d) and citation-document pairs (c, d) [4]. The corresponding probabilities for the observations are $p(w|d)$ and $p(c|d)$. This model was described in section 2.1.6. Another way to extend the PLSI model is to use the approach that was used when extending PLSI to do supervised learning in section 3.3. In this approach the observations are word-label-document triples, and the corresponding probabilities for the observations are $p(w, l, d)$. As noted in this section no special characteristics of the label features have been used when deriving the EM algorithm for the supervised extension. Therefore exactly the same formulae can be used when using link features instead of label features. The two approaches have pros and cons.

Keeping the observations separately in word and link observations provides a way to assign the relative importance of the two types of features through the α parameter as described in section 2.1.6. This gives the model more freedom and perhaps enhances the chance of successful performance in different domains or experiments where the relative importance of words and links differs. On the other hand the triple observations approach is a more clean approach, because mathematically there is no reason why the observations should be treated differently.

The two approaches differs in another very important way. The computational complexity and the memory consumption of the pair observations approach is a lot more attractive. During the EM training procedure there are some quite big matrices that has to be updated in each iteration. The biggest ones are the mixing proportions, that is the $p(z|w, d)$ and the $p(z|c, d)$ parameters for the pair observation approach and the

[4] I.e. the observations are $\frac{n(w,d)}{\sum_w n(w,d)}$ and $\frac{n(c,d)}{\sum_c n(c,d)}$

$p(z|w,c,d)$ for the triple observations approach respectively. Let's look at the sizes of these matrices. The separate observations approach uses $N_z N_w N_d + N_z N_c N_d = (N_w + N_c)(N_z N_d)$ entries for the mixing proportions matrices, where N_x denotes the number of words, citations, factors or documents. The triple observations approach uses $N_w N_c N_z N_d$ entries. Whereas the two types of features, words and links, contribute additively in the pair observations approach they contribute multiplicatively in the triple observations approach. This makes a huge difference in the computational requirements and in the space consumptions of the two approaches. Let's look at a small sized data set of 100 documents, 1000 words, 500 links and 5 factors and compute the space consumption of the two approaches.

$$\begin{aligned} \text{Space}_{\text{pair-observations}} &= (N_w + N_c)(N_z N_d) \text{ entries} \frac{8B}{\text{entry}} \\ &= (1000 + 500)(5 \cdot 100) \text{ entries} \frac{8B}{\text{entry}} \\ &= 750.000 \text{ entries} \frac{8B}{\text{entry}} = \underline{\underline{6MB}} \end{aligned} \tag{3.21}$$

$$\begin{aligned} \text{Space}_{\text{triple-observations}} &= (N_w N_c N_z N_d) \text{ entries} \frac{8B}{\text{entry}} \\ &= (1000 \cdot 500 \cdot 5 \cdot 100) \text{ entries} \frac{8B}{\text{entry}} \\ &= 250 \ 10^6 \text{ entries} \frac{8B}{\text{entry}} = \underline{\underline{2GB}} \end{aligned} \tag{3.22}$$

On top of that the smaller matrices have to be added. Even with moderate sized data sets it very soon gets too demanding for normal PC's too handle the triple-observations approach.

The separate observations approach is chosen for implementation in this work, mainly because of the computational advantages, but also because it gives the possibility of experimenting with weighting of importance. The joint probabilistic model presented in section 2.1.6 is repeated here for convenience:

$$p(w|d) = \sum_z p(w|z)p(z|d), \qquad p(c|d) = \sum_z p(c|z)p(z|d) \tag{3.23}$$

The updating rules for the parameters are given by formulae 2.36, 2.37 and 2.38.

3.4.2 Model implementation

The implementation of PLINK facilitates training the model, folding-in unseen documents and computing log likelihood of data. The training procedure takes as input a term-document matrix and a link-document matrix. The model parameters are computed by implementing the EM updating rules given by the formulae 2.36, 2.37 and 2.38.

Unseen documents are folded-in using an approach very similar to the one used for the PLSI model described in section 2.1.4. Now iteration between updating $p(z|d)$ and $p(z|w,d)$, $p(z|c,d)$ is performed. It is actually possible to fold-in new documents in different ways according to what features are present for the new documents. Both words and links or just one of them can be used. If we want to predict the links for new documents for which we only have the word features then they are folded-in using the term-document matrix and it is then possible to predict the links through the hidden factors, z. More on link prediction in section 3.4.5

The log likelihood computation takes as input a test term-document matrix, a test link-document matrix and $p(z|d_{test})$ parameters for the test documents. The latter can be computed by folding-in the test documents as described above. The term-document observations vs. the link-document observations are weighted in the log likelihood computation with the α parameter according to equation 2.30.

3.4.3 Synthetic data set

A synthetic data set has been constructed in order to evaluate the PLINK model. The data set is actually the MED data used earlier with a number of links added. The way the links are added adheres to the assumption of the PLINK model, that there is a connection between contents and links. The MED data set consists of a number of classes, which means that documents in the same class have similar contents. The links are added in a way so that all documents in the same class also share exactly the same link pattern. The challenge for the PLINK model is to capture this connection.

In figure 3.21 it is shown how the link patterns are added on top of the MED data set. The links are added so that all the documents in the same class contain links to the same documents. E.g. the documents in class 1 all contains links to the documents in class 2. If all the documents in class 1 contain links to all the documents in class 2 and if the same is the case for the other classes then we have a fully connected data set.

When using a model on a fully connected data set we would expect the model to perform well, because it is easy to see the link structure. E.g. classification should be easy, because all the documents in the same class share exactly the same links. One way to complicate the task of finding patterns in the data set is by making it more sparse. That is, we still make

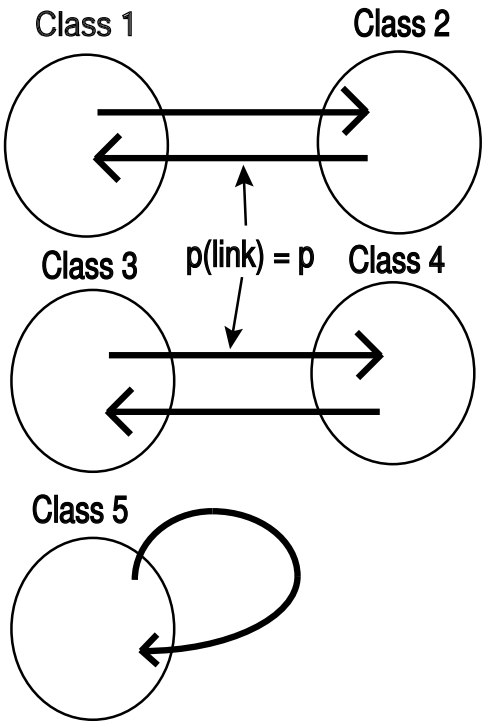

Figure 3.21: A synthetic data set constructed by adding links on top of the five classes in the MED data set. The figure shows how the links are added such that all the documents in the same class have links to the same class. E.g. all the documents in class 1 have links to documents in class 2. Furthermore all the links are added with a certain probability, p, called the sparsity factor.

sure that e.g. documents in class 1 only have links to class 2, but now each document in class 1 only has links to some of the documents in class 2. It is now a more complicated task to find the link structure, but also a more realistic one, since real-world data is not going to have a fully connected link structure.

The construction of the sparse link structure has been implemented by introducing a sparsity factor, p, which denotes the probability that each link of the fully connected data set is actually added to a specific document. E.g. if the sparsity factor $p = 0.5$ then for each document in class 1 we add a link to each document in class 2 with a probability of 50 %. The documents in class 1 will all have different links, but on average each one will link to half of the documents in class 2.

3.4.4 Normalizing the updating rules

The EM algorithm for PLINK developed in section 2.1.6 was shown to be different from the one given by Hofmann and Cohn [7]. The difference is

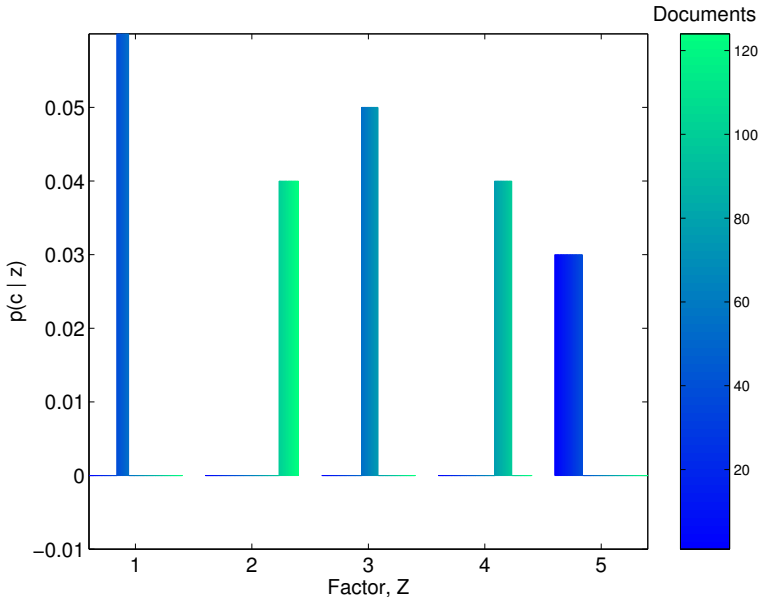

Figure 3.22: The link patterns for the 5 factors of the PLINK model trained with normalizing constants. The model has been trained on the fully connected MED data set described in section 3.4.3. The values of $p(c|z)$ has been sorted according to the class to which it links. Thus the figure shows that each factor only has links to one class. This shows that when the model is trained with normalizing constants then each factor captures the link pattern corresponding to exactly one class in the MED data set as we would expect.

the normalizing constants appearing in the updating rules for the parameters. Here some experiments will show that the normalizing constants are necessary to obtain good results.

The PLINK model has been trained both with and without the normalizing constants. The data set used is the fully connected MED data set (see section 3.4.3), for which it is obvious what link structure to expect the factors to capture. There are five different link patterns, one for each class, and we expect each factor to capture exactly one pattern corresponding to the class that it represents.

In figure 3.23 it is shown that the PLINK model without normalizing constants lets some factors represent multiple link patterns whereas others don't represent any link patterns. The PLINK model trained with normalizing constants captures the five link patterns that we would expect it to do as seen in figure 3.22. An intuitive explanation why the normalization is necessary is that if e.g. the $p(c|z)$ parameters are not normalized then they can have arbitrary high values for all links and as a consequence they end up representing more than one class of links. On the other hand if they are normalized then they are sort of "punished" for having high values for other

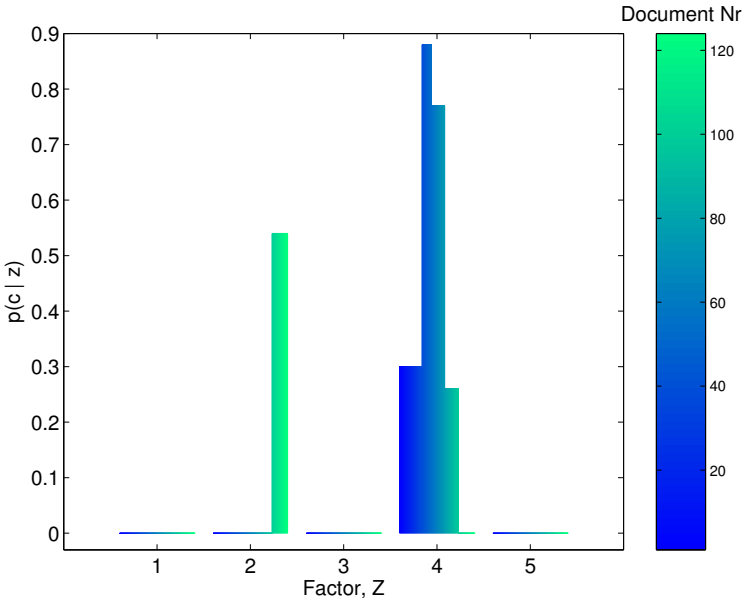

Figure 3.23: The link patterns for the 5 factors of the PLINK model trained without normalizing constants. The model has been trained on the fully connected MED data set described in section 3.4.3. The values of $p(c|z)$ has been sorted according to the class to which it links. The figure shows that factor 2 only has links to one class whereas factor 3 has links to the remaining 4 classes. The other factors don't have links to any classes. This shows that when the PLINK model is trained without normalizing constants then some factors end up representing multiple link patterns and some factors don't represent any link patterns.

classes than the most likely ones. In this case the likelihood becomes largest if all factors each have high values for one particular class.

3.4.5 Predicting links

Given a trained PLINK model and a new unseen document is it possible to predict the links that the document will have based on the content of the document? This is the task, which we will try to solve in this section. The reason why we are interested in this question in particular is that the hypothesis of this report is that the task of predicting links from content is a task very similar to the matching problem. E.g. in the job matching problem we want to find matches between jobs and applicants based on the contents of the job offer and the application. That is, if we apply the PLINK model to a job matching task, we assume that we can represent matches between jobs and applicants as links.

Methods of evaluation

Previously the performance of the models used in this report has been evaluated by using them as a classifier. That was the case with both the basic PLSI model and with the supervised PLSI model. Classification is possible with the PLINK model as well and we shall use classification results as a way of evaluating the model, but apart from that we need a way to evaluate the capability of the model to predict links for a document based on the content.

The link structure for new documents predicted by the model is given by

$$p(c|d_{test}) = \sum_z p(c|z)p(z|d_{test}) \tag{3.24}$$

where $p(z|d_{test})$ is computed by folding-in the new documents based on their contents, that is the term-document observations, and $p(c|z)$ are model parameters computed when training the model. The model trained on the synthetic data set visualized in figure 3.21 gives good predictions if e.g. a test document belonging to class 1 have high values for links in $p(c|d_{test})$ pointing to documents in class 2.

Inspired by the well-known precision and recall scores a new score value, which we will call the *link score*, will be introduced to measure how well the model predicts the links of new documents based on the content. The precision and recall scores are used in information retrieval tasks to measure the results of a number of documents retrieved from a document collection. In our case we are dealing with link retrieval from a link collection, so the precision and recall scores look like this:

$$\text{precision} = \frac{N_{\text{correct links retrieved}}}{N_{\text{links retrieved}}}, \quad \text{recall} = \frac{N_{\text{correct links retrieved}}}{N_{\text{correct links}}} \tag{3.25}$$

In our case, $p(c|d_{test})$ can be thought of as retrieving links with a certain probability. Thus the number of correct links retrieved for one document corresponds to summing the values of $p(c|d_{test})$ over the correct links. The total number of links retrieved for one document, that is the denominator in the precision score, corresponds to summing over all links in $p(c|d_{test})$ for that document, which always sums to one. Also the total number of correct links, that is the denominator in the recall score, corresponds to a value of one, because if $p(c|d_{test})$ gave perfect predictions it would only have values different from 0 for correct links, and these would then sum to one. These observations make the precision and recall scores equal in our case. Now the link score is introduced as the mean precision/recall score taken over all the test documents. That is,

$$\text{link score} = \frac{1}{N_{d_{test}}} \sum_{d_{test}} \sum_{\text{correct links}} p(c|d_{test}) \tag{3.26}$$

The link score is actually very similar to the log likelihood of the link-document observations given by

$$LL_{\text{ link-document}} = \sum_{d_{test}} \sum_{c} \frac{n(c, d_{test})}{\sum_c n(c, d_{test})} \ln p(c|d_{test}) \tag{3.27}$$

However, by not using the logarithm the link score becomes a number, which is easier to interpret, because we can think of the link score as the percentage of the correct links predicted.

Experiments

In this section experimental results of the PLINK model will be presented. The synthetic data set described in section 3.4.3 will be used. The link part of the data contains no noise, because documents from different classes don't share any links. The link pattern can be made more or less sparse by varying the sparsity factor, p, which will make it more or less difficult to predict the links. Another source of variation is the model parameter α, which determines how content and link information should be weighted relatively. With $\alpha = 1$ only content is prioritized and with $\alpha = 0$ only links are prioritized. Any value in between prioritizes both content and links. The methods of evaluation are classification error rate, link score for the link predictions of test documents and log likelihood on link-document observations for test documents.

In figure 3.24 the error rates vs. α for data sets with different sparsities can be seen. There is a large variation in data, and if we were to make statistically valid conclusions it would be difficult to say anything. However, there certainly seems to be some trends. The following observations should be seen in the light of that. For the fully connected data set with the

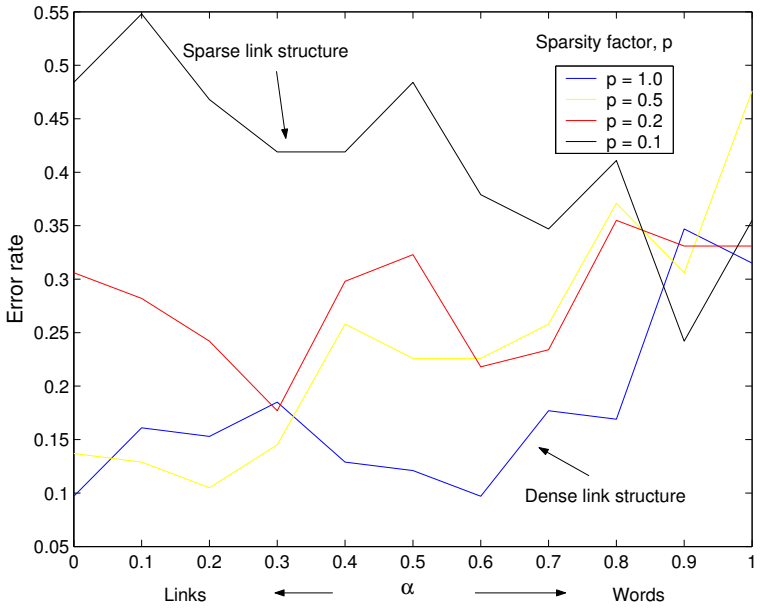

Figure 3.24: Classification error rate vs. α for different link sparsities. The PLINK model is trained and tested on the MED data set extended with synthetic links as described in section 3.4.3. The figure shows that for a sparse link structure ($p = 0.1$) the error rate gets lower the more we weight word information in the model. For a dense link structure ($p = 1.0$) the error rate gets higher the more we weight word information. And for a sparsity factor of $p = 0.2$ and $p = 0.5$ the lowest error rate is obtained by combining link and word information.

sparsity factor $p = 1$, the error rate varies between 0.1 and 0.35. When α is increased the link information is weighted less and in general then the error rate also increases. The reason is that in the fully connected model it is much easier to distinguish between the different classes by looking at the link information than by looking at word information. The opposite is the case for the sparse link structure with a sparsity factor of 0.1. Here the error rate decreases from 0.5 at $\alpha = 0$ to 0.35 at $\alpha = 1$. The reason is that the link information is so sparse that it is easier to distinguish the classes based on the content information. That is even though the content information is noisy, because the same words are shared among different classes. For a sparsity factor value of 0.2 the lowest error rate is found at an α value of 0.3, which indicates that in this case better results are obtained by using both content and link information than by using just one type of information. The result that using both content and link information gives a lower error rate is also the conclusion that Cohn and Hofmann draw in [7].

Looking at figure 3.26 showing the link score vs. α the same observations as with the error rate can be made. For the fully connected data set,

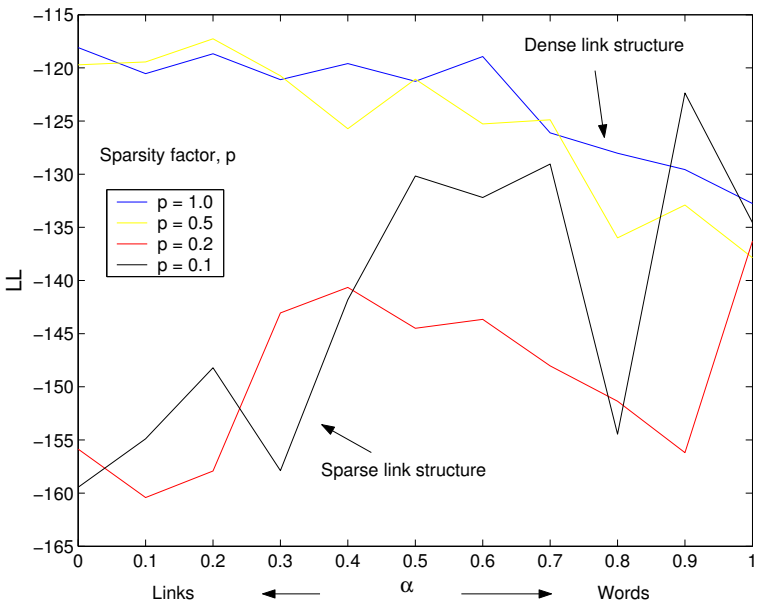

Figure 3.25: Test log likelihood vs. α for different link sparsities. The PLINK model is trained and tested on the MED data set extended with synthetic links as described in section 3.4.3. The LL varies more than the error rate which is showed in figure 3.24, but the general trends are the same.

increasing α gives poorer results, that is a lower link score. For the sparse set with a sparsity factor of 0.1 increasing α gives better results, that is a higher link score. And for the data set with a sparsity factor of 0.2 we get the highest link score of 0.4 at $\alpha = 0.3$. The same conclusions can be drawn from looking at the log likelihood on the test data vs. the α value in figure 3.25.

Another point to make is that the more links, that is the more dense the link structure is, the better results we obtain in general. This is the case for all the three measures of performance. In practice this is not a very useful result, since the sparsity of the links in documents is not a factor that we can control. But it confirms our expectations to the behavior of the model.

Yet another observation confirming our expectations is the following. When $\alpha = 1$ the link information is not used and the model is reduced to the PLSI model. We notice that the sparsity of the link data doesn't matter, because we get similar results for all values of the sparsity factor, p. The error rates are approximately 0.35, which is similar to the classification results obtained with the basic PLSI model presented in section 3.2.3.

Finally we want to answer the question: Is the model useful in predicting links based on the contents of unseen documents? For the fully connected data set we obtain a link score of 0.6. This means that 60 % of the correct links are retrieved, which seems like a good result. We cannot expect real

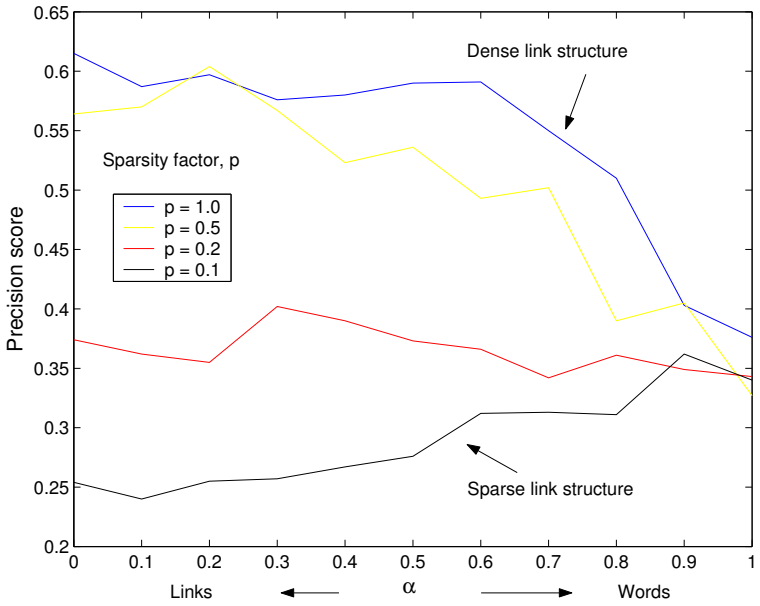

Figure 3.26: Link score vs. α for different link sparsities. The PLINK model is trained on the MED data set with synthetic links added as described in section 3.4.3. Link score is a measure of the capabilities to predict links and it is computed based on $p(c|d_{test})$ according to equation 3.26. The figure shows that for a sparse link structure ($p = 0.1$) the link score gets higher the more words are weighted in the model. For a dense link structure ($p = 1.0$) the link score gets higher the more we weight links in the model. And for sparsity factor of p = 0.2 and $p = 0.5$ the link score is the highest if both link and word information are weighted in the model.

data to be so well behaved, so we also have to consider more sparse link data. In the case of a sparsity factor of 0.2 we can still obtain a link score of 0.4, which means that 40 % of the correct links were retrieved, which still appears useful.

Reference Flow

In the previous section we have experienced that the link information is useful if it is not too sparse. In a lot of domains, though, the link data is rather sparse. E.g. that is the case in the WWW domain. It is the hope that the concept of "reference flow" will be useful for predicting links in domains where links are sparse. The concept of reference flow will be introduced and experiments will be presented and compared with the previous experiments on link prediction.

Reference flow is introduced by Hofmann and Cohn in [7]. The idea is that points in factor space, $\overrightarrow{z} = \{p(z_1|d), p(z_2 d), ..., p(z_k|d)\}$, represent

topics and the links between those points comprise a reference flow indicating the strengths of the connection between topics. In figure 3.27 the reference flow is shown for the WebKB data set (see section 3.1.2). All links in the data set contribute to the reference flow between any two points in factor space. This is the motivation for using reference flow to predict links based on sparse link data. If we have very few links then it seems desirable that all links in the data set contribute to the prediction of links for a new document.

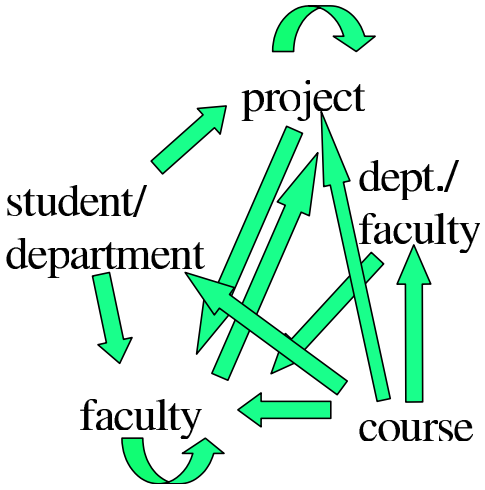

Figure 3.27: The reference flow is shown between topics in the WebKB data set 3.1.2. The figure is copied from [7]

Let's look at how to compute the reference flow from a source point in factor space, $\overrightarrow{z_s}$, to a target point, $\overrightarrow{z_t}$. Each link in the training set contributes to this reference flow. Let's consider a link, c_k from a document d_i to a document d_j. If we introduce the probability that the source point, $\overrightarrow{z_s}$, is responsible for that link, $p(c_k|\overrightarrow{z_s})$ and the probability that the target point is responsible for the target document, $p(d_j|\overrightarrow{z_t})$ then the contribution of the link to the reference flow is their product $p(c_k|\overrightarrow{z_s})p(d_j|\overrightarrow{z_t})$. The concept is illustrated in figure 3.28.

Summing over all links in the training set gives us the total reference flow from a source topic, $\overrightarrow{z_s}$, to a target topic, $\overrightarrow{z_t}$:

$$\text{ReferenceFlow}_{\text{ts}} = \sum_{\text{links, } c_k} p(c_k|\overrightarrow{z_s})p(d_j|\overrightarrow{z_t}) \tag{3.28}$$

We still need to specify how to compute $p(c_k|\overrightarrow{z_s})$ and $p(d_k|\overrightarrow{z_t})$, however. These quantities denote the probability that the link was generated by the source topic and that the target document was generated by the target topic respectively. How to compute this is not specified by Hofmann and Cohn [7]. Here we choose to compute it in the following way:

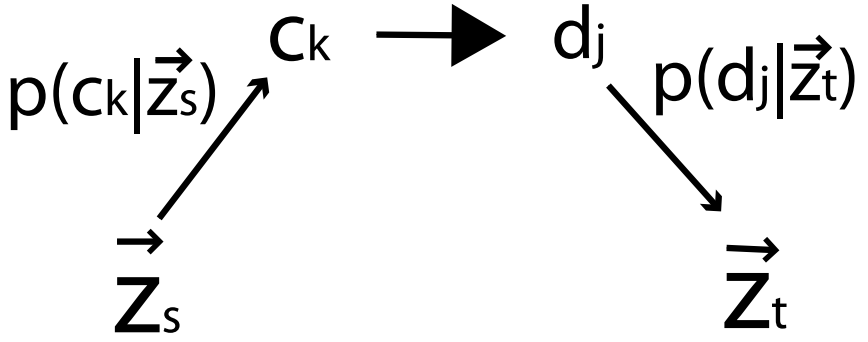

Figure 3.28: The contribution of a link c_k to the reference flow from topic z_s to topic z_t. The figure illustrates that the contribution of the link c_k depends on how likely the topic z_s is to generate c_k and how likely the topic z_t is to generate d_j.

$$\overrightarrow{z_s} = \{p(z_1|d_s), p(z_2|d_s), ..., p(z_k|d_s)\} \tag{3.29}$$

$$p(c_k|\overrightarrow{z_s}) = \sum_z p(c_k|z)p(z|d_s), \qquad p(d_j|\overrightarrow{z_t}) = \sum_z p(z|d_j)p(z|d_t)^5 \tag{3.30}$$

With the computation in place we turn to the application of reference flow again. Hofmann and Cohn suggest an interesting application of reference flow - an intelligent web crawler [7]. When crawling web pages, e.g. for indexing by a search engine, one approach is to randomly follow links in the documents that the crawler visits on its way. But the intelligent web crawler can be used to pick the links in those documents which are most likely to be connected to new pages on a certain topic of interest. If we consider a certain point in time, where the crawler can choose between following links in a number of documents then the intelligent crawler should choose to follow links from the document that has the highest reference flow to the topic at interest. The candidate pages can be represented in factor space corresponding to the source point, $\overrightarrow{z_s}$ and the topic of interest corresponds to the target point in factor space, $\overrightarrow{z_t}$. We then choose to follow links from the document that has the highest reference flow from $\overrightarrow{z_s}$ to $\overrightarrow{z_t}$

The motivation for introducing reference flow was to be able to predict links in a sparse link domain. The way to use reference flow for this task is to compute the reference flow from all test documents to all training documents. A certain test document has the highest probability of linking to those training documents to which it has the highest reference flows.

We can use the link score introduced in section 3.4.5 to evaluate reference flow. Now, however, the link score is only evaluated over links to training

[5]This is not a "clean" way to compute $p(d_j|\overrightarrow{z_t})$. See the subsection "Implicatins of the chosen for of PLINK" later in this section for an explanation.

documents, since we are not able to obtain $\overrightarrow{z_t}$ for documents linked to out of the training set. Thus the link score for reference flow is:

$$\text{link score} = \frac{1}{N_{d_{\text{test}}}} \sum_{d_{\text{test}},\ s} \sum_{d_{\text{train}},\ t} \text{ReferenceFlow}_{\text{ts}} \tag{3.31}$$

Experiments have been performed on the MED data set with synthetic links added as described in section 3.4.3. We are interested in comparing the performance of reference flow with the performance of the link prediction technique used in section 3.4.5 based on $p(c|d_{test})$.

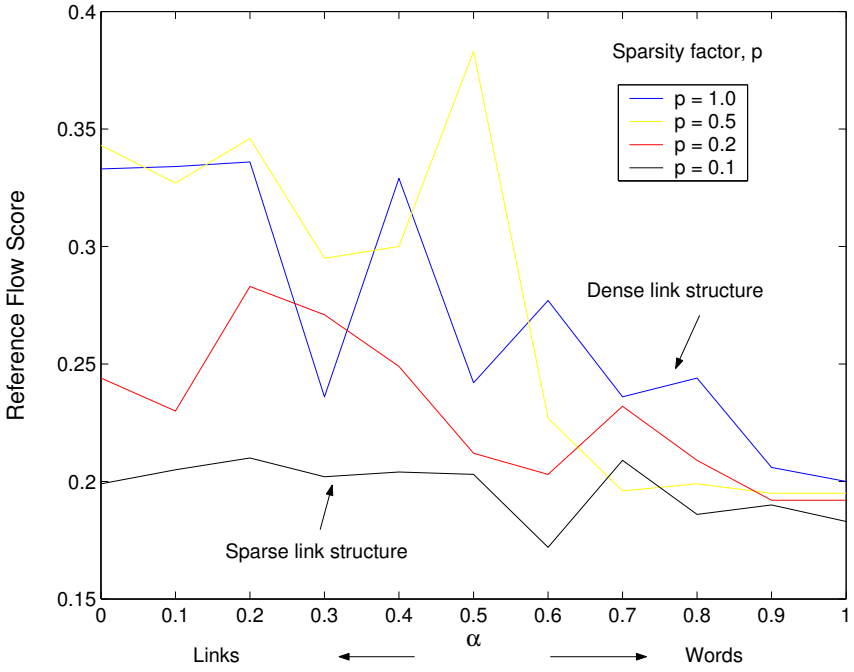

Figure 3.29: The link score based on reference flow vs. α computed for different link sparsities. The PLINK model has been trained on the MED data set with synthetic links added as described in section 3.4.3. The link score is computed according to equation 3.31. The figure shows that the reference flow obtains a smaller link score than the link score based on $p(c|d_{test})$ when compared to figure 3.26. This implies that it is not desirable to use reference flow instead of $p(c|d_{test})$.

The measure of performance is link score. In figure 3.29 the link score for reference flow (Reference flow score) is plotted vs. the α parameter weighting word and link information for different link sparsities. This figure should be compared to figure 3.26 showing the link score for the first approach. When comparing the scores it is noted that the reference flow doesn't perform any

better. Not even for sparse link information do we gain anything by using reference flow to predict links.

The reason why reference flow is not better than the first approach based on computing $p(c|d_{test})$ might be that we thought that we gained something by letting our prediction of links be based on all links in the training set. But actually $p(c|d_{test})$ is implicitly based on all links too. All the links in the training set are used when estimating the model parameters, $p(c|z)$ and $p(d|z)$, on which $p(c|d_{test})$ is based (see equation 3.24).

Implications of the chosen form of PLINK

Another reason why reference flow is not very successful might be that the way we compute the reference flow is not a "clean" way. In equation 3.30 $p(d_j|\overrightarrow{z_t})$ is computed based on $p(z|d_t)$, which should actually have been $p(d_t|z)$, but with the chosen form of the PLINK model it is not possible to compute $p(d_t|z)$.

In [7] Hofmann and Cohn chooses to model the normalized observations $\frac{n(w,d)}{\sum_w n(w,d)}$ for words and $\frac{n(c,d)}{\sum_c n(c,d)}$ for links instead of modelling the un-normalized observations $n(w,d)$ and $n(c,d)$ according to equation 2.26 and 2.27. This choice has some implications, which are both good and bad. First it has the desired implication that all the documents are given the same importance, because the word and link observations are normalized by the total number of words and links in the particular document. But another implication is the following: Now we are not estimating the joint probabilities $p(w,d)$ and $p(c,d)$ anymore. Instead we are estimating the conditional probabilities $p(w|d)$ and $p(c|d)$ according to equation 2.28. This gives us a different set of parameters, which means that we cannot compute all the probabilities, which is possible in the first approach. In particular it is not possible to compute $p(d|z)$ in the approach taken by Hofmann and Cohn in [7]. It should be computed like this:

$$p(d|z) = \frac{p(z|d)p(d)}{p(z)} = \frac{p(z|d)p(d)}{\sum_d p(z|d)p(d)} \tag{3.32}$$

but since we do not have the $p(d)$ parameters it is not possible to do this. $p(d|z)$ is the authority of the document d within the topic z. So even though it is claimed in [7] that it is possible to compute the authority based on the PLINK model, this is actually not the case with the PLINK model in the given form.

If we had taken the other approach based on equation 2.26 and 2.27 we already had the quantity $p(d|z)$ and it would be possible to compute $p(z|d)$ according to

$$p(z|d) = \frac{p(d|z)p(z)}{p(d)} = \frac{p(d|z)p(z)}{\sum_z p(d|z)p(z)} \tag{3.33}$$

Thus with this approach both the authority of a document within a topic, $p(d|z)$, and the document representation, $p(z|d)$, could be computed.

3.4.6 Discussion on the PLINK model

In this section we have presented experiments with the PLINK model. The experiments have been performed on a synthetic data set constructed by adding synthetic links on top of the MED data set.

Two different approaches to building the PLINK model were described. The first one was based on separate observations of word-document and link-document pairs. The second one was based on word-link-document triples. The first one was chosen mainly because of smaller space consumption, which made it feasible to implement on a normal PC.

It was shown that normalizing the EM updating rules was necessary in order to get good experimental results. If no normalizing was implemented then the factors did not capture the link patterns that we expected. In particular some factors represented multiple link patterns and others represented no link patterns. After performing these experiments I have had email contact with Thomas Hofmann and David Cohn [15] in order to clarify their opinions on the results. They recognized that normalizing is necessary and that the normalizing constants should take part in the EM updating rules as we discovered in section 2.1.6.

The experiments have focused on the capabilities of the PLINK model to predict links based on the contents of the documents. The experiments have shown that the PLINK model is capable of predicting links to some degree. Based on the link score introduced in this section the PLINK model was able to retrieve 60% of the links for a dense link data set and 40% for a more sparse link data set. The results for an even sparser link data set were not too encouraging, because link information didn't contribute in any positive way to the link prediction capabilities. The concept of reference flow was introduced as an alternative approach to predicting links in a sparse linked data set. But it was shown that reference flow didn't perform any better than what we had already seen. Also when computing the reference flow we discovered a weakness of the chosen form of the PLINK model using normalized word and link observations. We noticed that we cannot compute the quantity $p(d|z)$. This quantity is also necessary for the application of PLINK to ranking relevant pages retrieved from a document collection as described in section 1.2.2.

Chapter 4

Conclusion

This work has focused on statistical text modelling and in particular on suitable models for the matching problems presented in the introductory chapter. In this final chapter we shall sum up the main results of this work and conclude on how suitable the models found are for the matching problems which motivated this work.

4.1 Main results

Tasks and models

A survey of tasks and models in the Information Retrieval domain has been conducted. Concerning the models used in information retrieval they are manifold, but there are two main approaches to building such models. The knowledge based approach uses mathematical logic for representation and inference. The data-driven one is based on statistical data collected from documents. A subtle understanding of semantics in language, which e.g. is required by a question answering system, appeals more to a logic representation system. A statistical model on the other hand is more useful when the required semantic understanding is a more topic oriented one.

This work has been concerned with the statistical approach. Even though the logic representation approach has had more success in some areas it is still desirable to have statistical models performing the tasks, since they require less human work. Statistical models can be trained to perform the tasks using data collected from the domain. In knowledge based systems experts have to code their knowledge into the system explicitly. This is time consuming but also sometimes hard to compete with for statistical models.

Concerning the results in the information retrieval domain some tasks are solved very reliably and efficiently by automatic systems such as finding a specific web page among millions of other web pages. On the other hand other tasks definitely require more work on statistical models. This work should be seen in this perspective.

State-of-the-art statistical text models

Through a detailed look into the mathematics of state-of-the-art text models such as Latent Semantic Indexing (LSI), Independent Component Analysis (ICA) and Probalistic Latent Semantic Indexing (PLSI) common trends can be observed. One observation to make is that they all consider a decomposition of the high-dimensional space in which documents are represented by term-vectors into a lower dimensional topic space.

PLSI is such a model, which has some desirable features. It was introduced to overcome the shortcomings of the lack of a solid statistical foundation of LSI. What we have found in this work is that the solid foundation made it very easy to extend PLSI to incorporate different kinds of information besides terms such as labels and links. Experimental results with the PLSI model show that it is capable of capturing semantics of text documents. In particular this is shown by using PLSI as a classifier. The classification results were not too impressive, but still far better than average.

Problems do exist with the PLSI model, though. One problem is scalability. Because document indices take part in the model explicitly the number of model parameters grow linear in the number of training documents. This problem is suggested solved by drawing topic mixtures for each document from a distribution instead of representing each document topic mixture as a parameter. The model suggested is called Latent Dirichlet Allocation.

The extension of PLSI to incorporating labels form the supervised PLSI model. The extension is straight forward and still it is one of the main achievements of this work. Experimentally it is shown that the supervised PLSI model is a better classifier that the basic PLSI model as we would expect.

If we were to use PLSI for a task, which required a more subtle semantic understanding we wouldn't expect the basic topic oriented PLSI model to perform very well. But one approach would be to extend the model with more fine-grained Natural Language features like we have extended it with labels and links. Such features could be word classes, word ordering and sentence length.

4.2 PLINK as a matching model

The motivation for looking at link models was the idea that predicting links was a task very similar to predicting matches. The PLINK model is formed by extending the basic PLSI model with link information. Here we shall answer the question: How useful is PLINK as a matching model?

We have shown that the solid statistical PLINK model is theoretically capable of predicting links. Therefore under the assumption that the matching problem is just like the link prediction problem, the PLINK model is a suitable model.

Since we haven't been able to gather any data for a real matching problem then the assumption of task similarity between matching and link prediction has not been investigated. This of course has to be done if PLINK is to be used for matching applications.

The shortcomings of PLSI applies to PLINK as well. That is, the scalability is still a problem theoretically even though it didn't play any role in the experiments of this work. Also, PLINK is a topic oriented model and as such it doesn't capture the subtle semantics of documents. But seen in the perspective of matching applications the model appears suitable, since matching some kind of profiles e.g. is about matching the overall type of persons. It is not about the small ambiguities in text understanding.

Another drawback of the PLINK model is concerned with the way that links are represented in the model. Just like the documents are treated as indices in the model parameters, so are the links. This means that a link is treated just like a word appearing in the document - it is just another type of document content. But the nature of links is different from the nature of words, since links connect to other documents. The point is that the contents of the documents linked to doesn't matter - even though one would think that the contents of documents linked to does make a difference for how to interpret a document. The links influence the estimation of the underlying topics and as such what words the topics should contain, but still, the contents of the documents linked to does not influence the estimation of the topics.

Experiments on link prediction with the PLINK model has shown that it is capable of predicting links to some degree - at least if the link structure is not too sparse. In a more sparse environment the PLINK model was not shown to be very successful. In terms of matching problems this doesn't seem to be a big problem, however. E.g. in the job matching problem one applicant often matches many jobs and one job often matches many applicants. At least if we consider a first round matching and not the task of choosing the final employee. In future work PLINK should be tested on real job matching data and also the results should be compared to other models capable of predicting links.

Compared to other possible ways to do matching the PLINK model has some advantages. Another possible way to do matching would be to use clustering. That is, a clustering algorithm would run e.g. on all applicants and job offers and if an applicant and a job offer end up in the same cluster we would consider them a match. A drawback with a clustering approach is that it matches the applicants and jobs without considering the difference inherent in applications vs. job offers. Even though they might describe the same person they would do it in a totally different way with the risk that they are not considered very similar by the clustering algorithm. On the other hand the PLINK model is capable of representing the asymmetry inherent in the job matching problem, because there is no restrictions on

what the links can point at. Therefore it is perfectly fine that we predict a match between not very similar profiles.

In conclusion this work has successfully investigated statistical models capable of modelling text. Motivated by the matching problems we have found that using models such as the PLINK model, which are capable of predicting links, is a feasible way to approach the task of predicting matches. Statistical text modelling requires a thorough understanding of the models. Strengths and weaknesses of the models have to be evaluated in relation to the task at hand, and that is in particular what this work has done.

Bibliography

[1] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 1995.

[2] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[3] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[4] TREC Conference. http://trec.nist.gov/, September 2003.

[5] Knut Conradsen. *En Introduktion til Statistik*. Department of Mathematical Modelling, Technical University of Denmark, 1999.

[6] Nick Craswell and David Hawking. Overview of the trec-2002 web track. In *Proceedings of TREC-2002*, Gaithersburg, Maryland USA, November 2002. http://www.ted.cmis.csiro.au/∼nickc/pubs/trecweb2002.pdf.

[7] Cohn D. and Hofmann T. The missing link - a probabilistic model of document content and hypertext connectivity. *Advances in Neural Information Processing Systems*, 13, 2001.

[8] WebKB data set. http://www-2.cs.cmu.edu/∼webkb/, August 2003.

[9] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. "indexing by latent semantic analysis". *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[10] Thomas G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

[11] D. Gildea and T. Hofmann. Topic-based language models using em. In *Proceedings of the 6th European Conference on Speech Communication and Technology(EUROSPEECH)*, 1999.

[12] M. Girolami and A. Kaban. On an equivalence between plsi and lda. In *Proc 26th Annual ACM Conference on Research and Development in Information Retrieval, SIGIR*, pages 433–434.

[13] L. K. Hansen, S. Sigurdsson, T. Kolenda, F. Nielsen, U. Kjems, and J. Larsen. Modeling text with generalizable gaussian mixtures. In *In-*

ternational conference on acoustics, speech and signal processing, volume 4, pages 3494–3497, 2000.

[14] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval*, pages 50–57, Berkeley, California, August 1999.

[15] Thomas Hofmann and David Cohn. *Private email correspondence*. August.

[16] Java. http://www.java.sun.com, September 2003.

[17] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Prentice-Hall, 2000.

[18] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[19] T. Kolenda, L. K. Hansen, and S. Sigurdsson. *Independent Components in Text*, pages 229–250. Springer-Verlag, 2000.

[20] MOLE Java library. http://mole.imm.dtu.dk/mole/www/matlab_java/links.htm.

[21] Matlab R13 online documentation. *http://www.mathworks.com/access/helpdesk_r12p1/help/toolbox/stats/lillietest.shtml*. September 2003.

[22] Edgar Osuna, Robert Freund, and Federico Girosi. Support vector machines: Training and applications. Technical Report AIM-1602, 1997.

[23] Mathew Richardson and Pedro Domingos. The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

[24] Matei Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *Proceedings of International Conference on Peer-to-peer Computing*, 2001.

[25] Stephen Robertson and Ian Soboroff. The trec 2002 filtering track report. In *NIST Special Publication: SP 500-251. The Eleventh Text Retrieval Conference (TREC 2002)*, 2002. Available at http://trec.nist.gov/pubs/trec11/t11_proceedings.html.

[26] Dan Roth, Chad Cumby, Xin Li, Paul Morie, Ramya Nagarajan, Nick Rizzolo, Kecing Small, and Wen tay Yih. Question-answering via enhanced understanding of questions. In *NIST Special Publication: SP 500-251. The Eleventh Text Retrieval Conference (TREC 2002)*, 2002. Available at http://trec.nist.gov/pubs/trec11/t11_proceedings.html.

[27] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, 1984.

[28] Roger E. Story. An explanation of the effectiveness of latent semantic indexing by means of a bayesian regression model. *"Information Processing & Management"*, 32(3):329–344, 1996.

[29] Ellen M. Voorhees. Overview of the trec-2002 question answering track. In *NIST Special Publication: SP 500-251. The Eleventh*

Text Retrieval Conference (TREC 2002), 2002. Available at http://trec.nist.gov/pubs/trec11/t11_proceedings.html.

[30] YAHOO. http://www.yahoo.com, September 2003.