NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

USE OF PROBABILISTIC TOPIC MODELS FOR SEARCH

by

Marco Draeger

September 2009

Thesis Advisor:	Kevin M. Squire
Second Reader:	Samuel E. Buttrey

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2009	Master's Thesis

4. TITLE AND SUBTITLE
Use of Probabilistic Topic Models for Search

5. FUNDING NUMBERS

6. AUTHOR(S)
Marco Draeger

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Department of the Navy

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

This thesis solves a common issue in search applications. Typically, the user does not know exactly which terms are used in a document he is searching for. Several attempts have been made to overcome this issue by augmenting the document model and/or the query. In this thesis, a probabilistic topic model augments the document model. Probabilistic document models are formally introduced and inference methods are derived. It is shown how these models can be used for information retrieval tasks and how a search application can be implemented. A prototype was implemented and the implementation is tested and evaluated based on benchmark corpora. The evaluation provides empirical evidence that probabilistic document models improve the retrieval performance significantly, and shows which preprocessing steps should be made before applying the model.

14. SUBJECT TERMS
Document Modeling, Information Retrieval, Semantic Search,
Bayesian Nonparametric Methods, Hierarchical Bayes

15. NUMBER OF PAGES
91

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

Standard Form 298 (Rev. 8–89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

USE OF PROBABILISTIC TOPIC MODELS FOR SEARCH

Marco Draeger
Captain, German Army
Diploma, University of the Federal Armed Forces, 2004

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 2009

Author: Marco Draeger

Approved by: Kevin M. Squire
Thesis Advisor

Samuel E. Buttrey
Second Reader

Robert F. Dell
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis solves a common issue in search applications. Typically, the user does not know exactly which terms are used in a document he is searching for. Several attempts have been made to overcome this issue by augmenting the document model and/or the query. In this thesis, a probabilistic topic model augments the document model. Probabilistic document models are formally introduced and inference methods are derived. It is shown how these models can be used for information retrieval tasks and how a search application can be implemented. A prototype was implemented and the implementation is tested and evaluated based on benchmark corpora. The evaluation provides empirical evidence that probabilistic document models improve the retrieval performance significantly, and shows which preprocessing steps should be made before applying the model.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

Problem Description and Proposed Solution

The Software Hardware Asset Reuse Enterprise (SHARE) "provides a capability for discovering, accessing, sharing, managing, and sustaining reusable assets for the Navy Surface Domain's programs." (Johnson and Blais, 2008). The purpose of this project is to avoid costly parallel development of system components and subcomponents.

SHARE consists of two physically separated parts, an asset library and a card catalog. The asset library collects combat systems software and supporting artifacts. Many of the documents in the asset library are therefore classified. The card catalog is a Web-based interface with unclassified descriptions of the assets that allows asset search. Besides the search, it provides functions as account registry, asset submission assistance, and asset retrieval request. For the scope of this thesis only the search feature is of interest.

Typical implementations of search applications are based on keywords, and count the number of occurrences of query terms in the documents and rank documents accordingly. A keyword-based implementation, however, requires the searcher to know the terminology used in the document that should be returned by the application. It is conceivable that similar components are used in different domains, in which different terminologies are used. This issue can be overcome by semantic search methods.

This thesis focuses on a small facet of semantic document modeling: probabilistic topic models. These models group word types in a collection of documents, and the groups are referred to as topics. As in the case of sets in fuzzy logic, a word type can belong to more than one of these groups. Therefore, for each word type a probability distribution over all topics can be defined.

Similarly, each document can be described by a probability distribution over all topics. For the search application, this allows the augmentation of a user query by terms that belong to the respective topics.

In practice, a search application that is based only on the topic model produces a feature space that is too coarse. The result is that almost all documents in a collection are returned by the application. This issue can be overcome by using the topic model as an augmentation of a keyword-based algorithm. Both methods combined can improve the retrieval performance significantly.

Explored Models and Implementation

In this thesis, the focus is on two different Bayesian document models: Latent Dirichlet Allocation (LDA), introduced by Blei *et al.* (2003), and Hierarchical Dirichlet Processes (HDP), described by Teh *et al.* (2006).

LDA is a parametric Bayesian model that specifies a prior probability distribution on the topics that are covered in a document. These topics form a latent feature set that

describes a document collection better that just the words in a dictionary. Using this model, it is possible to use keywords from a query to infer the most likely topics associated with the query. The next step in the search process is then to find documents that cover these topics. As mentioned, LDA is a parametric model. The parameter is the dimensionality of the topic space, or simply the number of topics that should be used in the model. In LDA, this number cannot be inferred from the data directly.

This issue is overcome by HDP, a nonparametric Bayesian model. In HDP, the number of topics is an outcome of the model and not an input parameter. HDP requires greater computational effort than LDA, which mitigates the advantage of not having to specify the number of topics in advance.

Both document models were implemented separately and combined with a simple keyword-based model. Implementation for LDA is entirely written in Java, using publicly available libraries. The model for HDP is written in Matlab and the results are imported into a Java-based application.

In addition, a search engine that can use either one of the introduced models was created. In order to compare results from different models and parameter settings, an evaluation program was also written. The whole application collection is available as a Java library, which allows the implementation and testing of search applications. Currently, all applications are command-line-based, but it is an easy task to add a graphical user interface or attach the applications to a web server (*e.g.,* Apache Jakarta Tomcat [1]).

Experimental Results and Recommendations

In the experiments, publicly available benchmark collections that are also used in the information retrieval literature were used. The results show significant improvements over the baseline. In addition to the comparison of the probabilistic topic model effects, the preprocessing steps necessary to prepare the documents before they can be processed in a document model for information retrieval were examined.

The examined preprocessing steps are stemming and removing of rare types. Stemming reduces words in a document to a common stem (*e.g.,* "running" becomes "run"). Rare types are words that are used less than five times throughout the collection. Since LDA and HDP try to discover correlations between words, such rare types can reduce the quality of the probabilistic topic model. In the experiments, a positive effect of stemming was shown empirically. Removing rare types, however, slightly hurt the retrieval performance. The reason is that the topic model augments a keyword-based model, which is improved by rare types, because rare types make documents more distinguishable.

The experimental results suggest that probabilistic topic models should be implemented in the SHARE search application or other search applications for specialized domains. The implemented search application works fast enough to be used in online ad hoc retrieval tasks.

[1] The Apache Software Foundation. Apache Tomcat. `http://tomcat.apache.org/`. Online, last accessed 26 August 2009

Acknowledgements

First, I need to thank my wonderful wife, Ute, who supported all my academic efforts within the last two years.

The list of people, whose advice was crucial for my research, would be too long to fit in here. A few people, however, shall be named here. I thank my advisors Prof. Kevin Squire and Prof. Sam Buttrey for their feedback, Prof. Craig Martell for providing the resources that made my research possible and for treating me as a member of his lab, and Prof. Ralucca Gera, who introduced me to the language modeling group led by Prof. Martell. Last, not least, I have to thank Prof. Robert Koyak for extensive discussions about stochastic modeling and statistical evaluation approaches.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1: Introduction

1.1 Background and Problem Description

The results of this thesis are thought to be applicable to the Software Hardware Asset Reuse Enterprise (SHARE) database, a database that contains requirements documents describing systems and components for Navy systems development. Having an effective and efficient search application for the database can contribute to avoiding expensive and risky double developments of components and subcomponents.

The SHARE database "provides a capability for discovering, accessing, sharing, managing, and sustaining reusable assets for the Navy Surface Domain's programs" (Johnson and Blais, 2008). SHARE consists of two physically separated parts, an asset library and a card catalog. The asset library collects combat systems software and supporting artifacts. Many of the documents in the asset library are therefore classified. The card catalog is a Web-based interface with unclassified descriptions of the assets that allows asset search. Besides search, it provides such functions as account registry, asset submission assistance, and asset retrieval request. For the scope of this thesis only the search feature is of interest.

Documents in the card catalog have two properties that suggest improved performance of semantic search over keyword search. First, the documents are human-generated free text. Second, the documents come from a specialized domain with non-standard terminology (Martell *et al.*, 2008).

1.2 Methodology

The final goal of this thesis was to implement a prototype of a search engine that augments keyword search by a probabilistic topic model. Two probabilistic topic models are evaluated in detail: Latent Dirichlet Allocation (LDA) and Hierarchical Dirichlet Processes (HDP).

In Chapter 2, the respective document models are motivated and formalized. Methods for inference and learning are introduced and derived from the model.

Chapter 3 shows how a document model can be used to augment a standard keyword search and how documents can be ranked according to their relevance for a query.

In Chapter 4, implementation issues are discussed and important points of the implementation are illuminated. Chapter 5 describes the conducted experiments and their results. All experiments are run on standard benchmark corpora that are used frequently in information retrieval. The corpora are CRANFIELD, CISI, MEDLINE, and TIME MAGAZINE[1]. All these corpora include a collection of documents, a collection of queries, and query-document relations. This allows the evaluation of the performance of the implemented prototype directly. The sample documents consist of abstracts and short articles, which makes them comparable to the SHARE card library.

Finally, Chapter 6 discusses the results and points to some directions for future research.

Sample code for inference and learning on selected models is provided in the Appendix A of this thesis. It is entirely written in R (R Development Core Team, 2008), which makes it very readable and easy to follow. Appendix B contains selected algorithms and implementations from the actual prototype.

1.3 Results

The experiments shown in Chapter 5 provide evidence that probabilistic topic models can improve a keyword search significantly. Additionally, it is shown that stemming (reducing word tokens to their respective stem) does not hurt information retrieval performance, while it reduces the storage demands of the computed index.

Retrieval performance is corpus dependent. For CRANFIELD, the maximum mean average precision achieved in experiments was 45%, while on CISI only 24% was achieved. This confirms findings by other authors who conducted retrieval experiments on the same data sets.

Furthermore, the implemented prototype is functional and can be used either standalone or embedded in a modular search engine as developed by Hawkins (2009). An LDA-based index can be computed very fast, in a couple of minutes, on a corpus with more than 1,000 abstracts. An HDP-based model requires more time for the same task. It does, however, not require the supply of as many fine-tuned model parameters as LDA does and had slightly better retrieval performance in the experiments.

[1] All corpora are retrieved from:
`http://ir.dcs.gla.ac.uk/resources/test_collections/`

1.4 Terminology

This thesis uses the standard terminology of document modeling and information retrieval. The basic unit in both fields is the word. This term can have two meanings: first, it can describe an entry in a dictionary or vocabulary. A vocabulary is an indexed list of words, which, in cases of ambiguity, will also refered to as *word types* or simply *types*. Second, a word can denote an observation in a document. In terms of implementation, an observation is often referred to as a *word token* or simply *token*. In terms of document modeling, the terms *word* and *observation* are used interchangeably, as are the terms *word* and *(word) type*. Where the term *word* is used, the context removes possible ambiguities.

1.5 Recommended Literature

A general introduction into the field of natural language processing (NLP) is given by Jurafsky and Martin (2008). The authors emphasize the advantages of statistical models over entirely rule based approaches to NLP. It is recommended to additionally look at the errata page and compare the models in the book with those in the original papers.

Bayesian Inference and empirical Bayesian methods are well described by Carlin and Louis (1996). The authors introduce the concept of Bayesian modeling and inference on a very general level before they show specific applications. Different inference methods, including Markov Chain Monte Carlo methods (MCMC) such as Gibbs sampling, are described and compared. On top of that, the book contains a high-level introduction to parametric and nonparametric Bayesian mixture models (*e.g.,* Dirichlet processes). A broader discussion of MCMC is given by Gamerman (1997). The author introduces Bayesian inference methods based on MCMC and shows several examples of how they are applied in practice.

Finite mixture models are studied in in a non-Bayesian way by McLachlan and Basford (1987). The authors describe the history and development of mixture models as well as many practical applications. Mixtures of normal components are in the focus of this book. Titterington *et al.* (1985) provide a statistical analysis of finite mixture models with components from different parametric families and compare Bayesian and non-Bayesian inference methods. Several tables provide an overview describing the cases in which a particular model should be applied and which inference methods are suitable.

Latent Dirichlet Allocation (LDA) is formally introduced by Blei *et al.* (2003). The authors give an overview of how LDA arises naturally as an extension of finite mixture models and give methods for inference and parameter estimation applied to document modeling. This

basic model has been extended in different ways that will be discussed in Chapter 2.

Dirichlet processes were formulated by Ferguson (1973). These stochastic processes represent a nonparametric Bayesian approach to stochastic modeling. The author proves several properties of Dirichlet processes and also shows how these models can be applied to known nonparametric problems. Antoniak (1974) shows how mixtures of Dirichlet processes can be formalized and applied in practice. Dirichlet processes are the basis for hierarchical Dirichlet processes, which are formally introduced by Teh *et al.* (2006). Section 2.2 shows how these can be applied to document modeling. To get a broader understanding of Dirichlet processes and other nonparametric Bayesian methods, the reader is referred to Ghosh and Ramamoorthi (2003). For a deeper discussion of the related measure-theoretic issues, (Billingsley, 1986) is recommended.

CHAPTER 2:
Document Modeling

Probabilistic topic models attempt to capture latent structure in documents. Each word in a document is assumed to come from a hidden (latent) topic, and probabilistic topic models assign each word to the proper topic. These latent topic assignments produce document models that have a high likelihood to generate a given corpus. For the information retrieval task, however, these document models need to prove that they indeed lead to better retrieval performance. This will be discussed in Chapter 5.

In all research that is presented in this thesis, a topic is considered to be a multinomial distribution over a vocabulary V. This allows the treatment of the problem of topic discovery as a parameter estimation problem. The cognitive notion of a topic is not within the scope of this thesis.

In the following, unless stated otherwise, we assume that words in a document have the exchangeability property (Carlin and Louis, 1996). That is, the document "do not panic" is produced with exactly the same probability as "not do panic" or "panic not do." More formally:

Definition 2.0.1. An infinite sequence of random variables $X_1, \ldots, X_n, \ldots$ is said to be exchangeable, if for all $n > 1$: $P(X_1, \ldots, X_n) = P(X_{\pi(1)}, \ldots, X_{\pi(n)}),\ \forall \pi \in S(n)$ in which $S(n)$ is the group of permutations of $1, \ldots, n$.

Note. If $X_1, \ldots, X_n, \ldots$ are i.i.d, they are also exchangeable, while the converse is not true in general.

A simplistic approach to document modeling is provided by Nigam *et al.* (2000), in which every document is represented as a mixture of unigrams. That is, every document covers exactly one topic, whereas different documents can share the same topic. This model will not be discussed in further detail.

2.1 Finite Mixture Models

A mixture of unigrams model has the shortcoming that the probability of a word occurring in a document is not well explained by a single parametric distribution. A mixture model attempts to fit to the document a model that consists of a mixture of probability distributions, which are conditioned on a latent variable space.

A very early topic model was introduced by Deerwester *et al.* (1990) and called "Latent Semantic Indexing." For topic discovery, Deerwester *et al.* use singular value decomposition (SVD) on the word-document co-occurrence matrix. Although this model has been applied successfully to information retrieval and language modeling tasks, it does not have a statistical foundation and can therefore not be seen as a probabilistic topic model. For a closer discussion of this model and its shortcomings, see Hofmann (1999).

In finite mixture models, each observation x is thought to be distributed with density $f(x|\boldsymbol{\phi}, \boldsymbol{\theta}) = \sum_{k=1}^{K} \theta_k f_k(x|\boldsymbol{\phi}^{(k)})$ in which k is the index of a mixture component, $\boldsymbol{\phi}^{(k)}$ is its parameter set, $f_k(\cdot|\boldsymbol{\phi}^{(k)})$ its density function determining the distribution, and $\boldsymbol{\theta}$ a vector of mixing probabilities that determines the proportions of densities in the mixture density. This requires $\boldsymbol{\theta}$ to add up to unity, $\sum_{k=1}^{K} \theta_k = 1$. $\boldsymbol{\theta}$ is therefore the parameter of a multinomial distribution (the mixing distribution) (Carlin and Louis, 1996). A general introduction to mixture models is given by McLachlan and Basford (1987).

Unfortunately, it is not trivial to estimate the number of mixture components K in a mixture model. In the case of creating topic models, too many components would result in overfitting, whereas too few components would lead into few very general topics that cannot be used for applications, such as search, in a reasonable way. Furthermore, the additional variability introduced by adding a mixture component can also be achieved by increasing the variance in one of the components. Griffiths and Steyvers (2004) suggest a greedy hill-climbing algorithm to maximize $P(corpus|K)$. This only works if the variability for each component is known in advance or at least assumed. Furthermore, if the number of components K is fixed in advance or parameters are shared among all components, this leads to problems in computing the posterior and predictive distributions (Carlin and Louis, 1996).

2.1.1 Probabilistic Latent Semantic Analysis

For the special case of a finite mixture model, in which $f_k = f$ is the probability mass function of a multinomial distribution, with parameters $\boldsymbol{\phi}^{(z)}$ and topic index z, this model represents "Probabilistic Latent Semantic Indexing" (pLSI), a probabilistic topic model introduced by Hofmann (1999).

The probability of an observation in pLSI is then

$$P(w_i) = \sum_{j=1}^{T} P(w_i|z_i = j)P(z_i = j),$$

in which z_i denotes the topic of the *i*th word token (w_i) and T the number of topics. The

model generating process in the pLSI model looks as follows:

- pick a document d from the corpus with probability $P(d)$
- draw a topic z from the distribution $\boldsymbol{\theta}^{(d)}$, the distribution over topics in document d
- draw a word w from the distribution $\boldsymbol{\phi}^{(z)}$, the distribution over words given topic z.

The joint probability model for words and documents is then $P(w, d) = P(w|d)P(d)$ in which $P(w|d) = \sum_{j=1}^{T} P(w|z_j)P(z_j|d) = \sum_{j=1}^{T} \phi_w^{(z_j)} \theta_{z_j}^{(d)}$, where z_j is the latent variable (Hofmann, 1999). However, pLSI does not specify how the mixture distributions $\boldsymbol{\theta}^{(d)}$ are generated. Therefore, it cannot be seen as a generative model for new documents.

2.1.2 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is another variant of a finite mixture model and a Bayesian extension of pLSI. LDA was first described by Blei *et al.* (2003) and is based on pLSI.

The advantage of LDA versus pLSI is that it models how the mixing proportions $\boldsymbol{\theta}^{(d)}$ for each document d are generated. The main idea is to treat these as random draws from a Dirichlet distribution. That is, $\boldsymbol{\theta}^{(d)} \sim \text{Dirichlet}(\alpha_1, \ldots, \alpha_T)$, in which α_j is the concentration parameter for the *j*th topic. In Bayesian statistic, the Dirichlet distribution represents the conjugate prior distribution for the multinomial distribution. The document generating process changes therefore in the following way:

1. Choose number N to be the number of slots in the document. Each slot will be filled with exactly one word.
2. Choose $\boldsymbol{\theta} \sim \text{Dirichlet}(\alpha)$, the distribution over topics for document d.
3. For each slot,
   (a) choose a topic $z_n \sim \text{Multinomial}(\boldsymbol{\theta}^{(d)})$,
   (b) choose a word w_n by sampling from $p(w_n|z_n)$, which is the z_nth column in the matrix $\boldsymbol{\phi}$.

Figure 2.1 shows the model in plate notation (Blei *et al.*, 2003). The parameters have the following meanings:

- α is the parameter for the Dirichlet distribution used as a prior for the topic distributions
- $\theta \sim \text{Dirichlet}(\alpha)$ is the probability distribution over topics for a given document (also called a multinomial distribution)

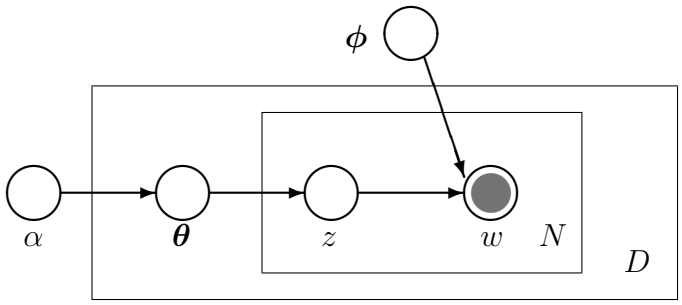

Figure 2.1: The basic LDA model in plate notation (after: Blei *et al.*, 2003)

- M represents the number of documents in the corpus.
- $N \sim \text{Poisson}(\xi)$ is a random number representing the number of words in a given document.
- $z \sim \text{Multinomial}(\theta)$ represents the topic of a particular slot in the document.
- $\boldsymbol{\phi}$ is the set of distributions over words, with one distribution for each topic.
- w is the word chosen for a particular slot in a particular document, determined by z and $\boldsymbol{\phi}$.

Obviously, documents are not generated in this manner. Modeling the document generating process this way, however, allows us the use of Bayesian inference methods to find groups of words that form a topic.

Formalization of the LDA model

The density of a K-dimensional Dirichlet random variable $\boldsymbol{\theta}$ with $\sum_{k=1}^{K} \theta_k = 1$ and for all $k = 1, \ldots, K: \ \theta_k \geq 0$ is defined as:

$$f(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1} \tag{2.1}$$

in which $\alpha_k > 0, \ \forall k = 1, \ldots, K$, and $\Gamma(\cdot)$ is the standard gamma function (Ferguson, 1973).

Assuming that the parameters α and $\boldsymbol{\phi}$ from Figure 2.1 are known, the joint distribution of a topic mixture $\boldsymbol{\theta}$, a sequence of topic labels $\mathbf{z}$, and a sequence of N words $\mathbf{w}$ is given

by

$$f(\boldsymbol{\theta}, \mathbf{z}, \mathbf{w}|\alpha, \boldsymbol{\phi}) = f(\boldsymbol{\theta}|\alpha) \prod_{n=1}^{N} p(z|\boldsymbol{\theta}) p(w_n|z, \boldsymbol{\phi}) = f(\boldsymbol{\theta}|\alpha) \prod_{v=1}^{V} \left(p(z|\boldsymbol{\theta}) p(w_v|z, \boldsymbol{\phi})\right)^{n_{w_v}}, \tag{2.2}$$

in which $p(z_n = i|\boldsymbol{\theta}) = \theta_i$ and n_{w_v} is the number of times word w_v appears in the document. The parameter α is not written in bold letters, because we assume it to be constant, that is, $\alpha_1 = \ldots = \alpha_T = \alpha$. Doing this leads to a symmetric Dirichlet distribution and an exchangeable (see: Definition 2.0.1) stochastic process. The marginal distribution of a document is then obtained by integrating over $\boldsymbol{\theta}$ and summing over all z (Blei *et al.*, 2003):

$$p(\mathbf{w}|\alpha, \boldsymbol{\phi}) = \int f(\boldsymbol{\theta}|\alpha) \prod_{n=1}^{N} \sum_{z} p(z|\boldsymbol{\theta}) p(w_n|z, \boldsymbol{\phi}) d\boldsymbol{\theta} \tag{2.3}$$

$$= \int f(\boldsymbol{\theta}|\alpha) \prod_{v=1}^{V} \left(\sum_{z} p(z|\boldsymbol{\theta}) p(w_v|z, \boldsymbol{\phi}) \right)^{n_{w_v}} d\boldsymbol{\theta}. \tag{2.4}$$

This allows us to express the marginal distribution of a corpus D as the product over the marginal distributions of all documents, since all $\boldsymbol{\theta}^{(d)}$ are independent draws from the same Dirichlet prior:

$$\begin{aligned} p(D|\alpha, \boldsymbol{\phi}) &= \prod_{d=1}^{M} \int f(\boldsymbol{\theta}|\alpha) \prod_{n=1}^{N_d} \sum_{z} p(z|\boldsymbol{\theta}) p(w_n|z, \boldsymbol{\phi}) d\boldsymbol{\theta} \\ &= \prod_{d=1}^{M} \int f(\boldsymbol{\theta}|\alpha) \prod_{v=1}^{V} \left(\sum_{z} p(z|\boldsymbol{\theta}) p(w_v|z, \boldsymbol{\phi}) \right)^{n_{d,w_v}} d\boldsymbol{\theta}. \end{aligned}$$

Inference

The goal is to compute the posterior distribution of the hidden variables $\boldsymbol{\theta}$ and $\mathbf{z}$ given a document and the model parameters (Blei *et al.*, 2003):

$$p(\boldsymbol{\theta}, \mathbf{z}|\mathbf{w}, \alpha, \boldsymbol{\phi}) = \frac{p(\boldsymbol{\theta}, \mathbf{z}, \mathbf{w}|\alpha, \boldsymbol{\phi})}{p(\mathbf{w}|\alpha, \boldsymbol{\phi})}$$

Blei *et al.* (2003) show that the computation of this distribution is intractable. Therefore they use a method that is called variational inference to get around this issue. For our experiments, we used a stochastic simulation method called Gibbs sampling, which is explained in Sec-

tion 2.1.3. Thus, the variational inference method will not be explained in any detail. For further discussion and an implementation in the LDA context, see Blei *et al.* (2003).

Another approach to inference in the LDA model is Expectation Propagation (Minka and Lafferty, 2002). This algorithm approximates integrals over functions that factor into simple terms with the general form $\int p(\boldsymbol{\theta}) \prod_{i=1}^{N} t_w(\boldsymbol{\theta})^{n_w} d\boldsymbol{\theta}$. Equation 2.4 satisfies this condition with $t_w(\boldsymbol{\theta}) = \sum_z p(z|\boldsymbol{\theta}) p(w|z, \boldsymbol{\phi})$. In order to apply Expectation Propagation, the terms t_w have to be approximated by terms with product form $\tilde{t}_w = s_w \prod_z \theta_z^{\beta_{w,z}}$. This expression resembles a Dirichlet distribution with parameters $\beta_{w,z}$ as Minka and Lafferty (2002) point out. An approximation to the posterior in Equation 2.4 is therefore given by

$$q(\boldsymbol{\theta}) \propto f(\boldsymbol{\theta}|\alpha) \prod_w \tilde{t}_w(\boldsymbol{\theta})^{n_w} = f(\boldsymbol{\theta}|\boldsymbol{\gamma}),$$

in which $\gamma_z = \alpha_z + \sum_w n_w \beta_{w,z}$ and $f(\cdot|\cdot)$ denotes the Dirichlet density. The Expectation Propagation algorithm then performs an iterative optimization of the auxiliary parameters to compute the best approximation to the true posterior distribution function. A sample implementation of the algorithm using the R (R Development Core Team, 2008) environment is provided in Appendix A.1.1.

2.1.3 LDA with Random Word Distribution

Griffiths and Steyvers (2004, 2006) introduce an extension of the basic LDA model, which is also discussed by Blei *et al.* (2003). In this model, the distribution over words specified by a topic is not known a priori but thought as random with a Dirichlet density. Thus, they pursue a fuller Bayesian approach, which is also fully generative. Figure 2.2 shows the model in plate notation. Instead of having a multinomial distribution $\boldsymbol{\phi}^{(z)}$ for each topic as a model parameter, this distribution is now a random variable following a symmetric Dirichlet distribution with parameter β (Griffiths and Steyvers, 2006).

Formalization of the Extended Model

Under these assumptions and by setting $\beta_1 = \ldots = \beta_V = \beta$, in which V is the size of the vocabulary, and again $\alpha_1 = \ldots = \alpha_T = \alpha$, the updated joint density function is:

$$f(\boldsymbol{\theta}, \boldsymbol{\phi}^{(z)}, \mathbf{w}, \mathbf{z}|\alpha, \beta) = f(\boldsymbol{\theta}|\alpha) \prod_{n=1}^{N} p(z_n|\boldsymbol{\theta}) p(w_n|z_n, \boldsymbol{\phi}^{(z_n)}) f(\boldsymbol{\phi}^{(z_n)}|\beta).$$

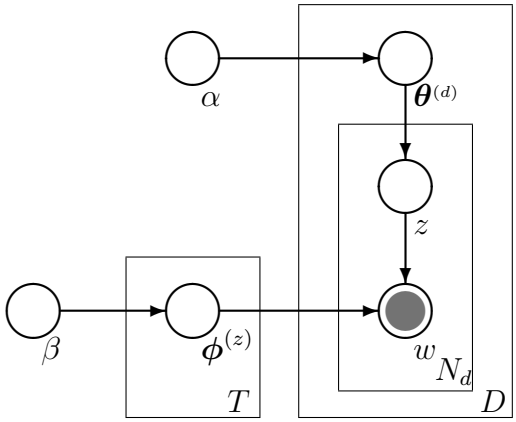

Figure 2.2: The fully generative model in plate notation (after: Griffiths and Steyvers, 2006)

Integrating out $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ separately yields

$$\begin{aligned}
p(\mathbf{w}|\mathbf{z},\beta) &= \prod_{i=1}^{I} p(w_i|z_i,\beta) = \prod_{j=1}^{T}\prod_{i=1}^{I} p(w_i|z_i=j,\beta) \\
&= \prod_{j=1}^{T}\int_{\Phi}\prod_{i=1}^{I} p(w_i|z_i=j,\phi) f(\phi|\beta)d\phi \\
&= \prod_{j=1}^{T}\int_{\Phi}\prod_{i=1}^{I} \phi_{w_i}^{\delta(z_i=j)}\frac{\Gamma(V\beta)}{\Gamma(\beta)^V}\prod_{v=1}^{V}\phi_v^{\beta-1}d\phi \\
&= \prod_{j=1}^{T}\int_{\Phi}\prod_{v=1}^{V} \phi_v^{n_j^{(v)}}\frac{\Gamma(V\beta)}{\Gamma(\beta)^V}\prod_{v=1}^{V}\phi_v^{\beta-1}d\phi \\
&= \left(\frac{\Gamma(V\beta)}{\Gamma(\beta)^V}\right)^T \prod_{j=1}^{T}\int_{\Phi}\prod_{v=1}^{V}\phi_v^{\beta-1+n_j^{(v)}}d\phi.
\end{aligned}$$

The expression inside the integral is proportional to a Dirichlet distribution with parameters $\beta-1+n_j^{(v)}$. The normalizing constant is thus $\frac{\Gamma(n_j^{(\cdot)}+V\beta)}{\prod_{v=1}^{V}\Gamma(n_j^{(v)}+\beta)}$ and therefore the final expression can be simplified to

$$p(\mathbf{w}|\mathbf{z},\beta) = \left(\frac{\Gamma(V\beta)}{\Gamma(\beta)^V}\right)^T \prod_{j=1}^{T}\frac{\prod_{v=1}^{V}\Gamma(n_j^{(v)}+\beta)}{\Gamma(n_j^{(\cdot)}+V\beta)},$$

in which $n_j^{(v)}$ is the number of times topic j was assigned to word v and $n_j^{(\cdot)}$ is the number of times topic j was assigned in total. The total probability of a topic sequence $\mathbf{z}$ is then

$$\begin{aligned} p(\mathbf{z}|\alpha) &= \prod_{d=1}^{D} \int_{\Theta} p(\mathbf{z}^{(d)}|\boldsymbol{\theta}, \alpha) f(\boldsymbol{\theta}, \alpha) d\boldsymbol{\theta} \\ &= \prod_{d=1}^{D} \int_{\Theta} \prod_{j=1}^{T} \theta_j^{n_j^{(d)}} \frac{\Gamma(T\alpha)}{\Gamma(\alpha)^T} \prod_{j=1}^{T} \theta_j^{\alpha-1} d\boldsymbol{\theta} \\ &= \left(\frac{\Gamma(T\alpha)}{\Gamma(\alpha)^T}\right)^D \prod_{d=1}^{D} \int_{\Theta} \prod_{j=1}^{T} \theta_j^{\alpha-1+n_j^{(d)}} d\boldsymbol{\theta} \\ &= \left(\frac{\Gamma(T\alpha)}{\Gamma(\alpha)^T}\right)^D \prod_{d=1}^{D} \frac{\prod_j \Gamma(n_j^{(d)} + \alpha)}{\Gamma(n_\cdot^{(d)} + T\alpha)}, \end{aligned}$$

in which $n_j^{(d)}$ is the number of times topic j was assigned to a word in document d and $n_\cdot^{(d)}$ is the number of words in document d (Griffiths and Steyvers, 2004).

Inference

The posterior probability of topics given a corpus is computed as

$$p(\mathbf{z}|\mathbf{w}, \alpha, \beta) = \frac{p(\mathbf{w}, \mathbf{z}|\alpha, \beta)}{\sum_{\mathbf{z}} p(\mathbf{w}, \mathbf{z}|\alpha, \beta)}, \tag{2.5}$$

which is unfortunately intractable (Blei *et al.*, 2003). As Griffiths and Steyvers (2004) show experimentally, the best way to estimate $p(\mathbf{z}|\mathbf{w})$ is to use Gibbs sampling, a Markov Chain Monte Carlo Method (MCMC). These methods are discussed in Gamerman (1997) and Carlin and Louis (1996) and formally introduced by Geman and Geman (1990).

Gibbs sampling does not directly compute the posterior probability distribution but returns samples from the true posterior distribution after convergence. From these samples the actual distribution can then be estimated. After randomly assigning a topic to each word in the corpus (resulting in a vector $\mathbf{z}^{(0)}$), the algorithm works as follows:

$$\begin{aligned} &\text{Draw } z_1^{(1)} \sim p(z_1 = j|z_2^{(0)}, \ldots, z_I^{(0)}), \\ &\text{draw } z_2^{(1)} \sim p(z_2 = j|z_1^{(1)}, z_3^{(0)}, \ldots, z_I^{(0)}), \\ &\qquad \vdots \\ &\text{draw } z_I^{(1)} \sim p(z_I = j|z_1^{(1)}, \ldots, z_{I-1}^{(1)}), \end{aligned}$$

until convergence of the algorithm. The sampling distribution p has the following form (Griffiths and Steyvers, 2004):

$$p(z_i = j|\mathbf{z}_{-i}, \mathbf{w}, \alpha, \beta) \propto \frac{n_{-i,j}^{(w_i)} + \beta}{n_{-i,j}^{(\cdot)} + V\beta} \frac{n_{-i,j}^{(d_i)} + \alpha}{n_{-i,\cdot}^{(d_i)} + T\alpha}, \tag{2.6}$$

in which $n_{-i,j}^{(\cdot)}$ is the number of times topic j was assigned to a word not including the current assignment. Later, a derivation for this full conditional probability mass function is provided. Appendix A.1.2 shows an implementation of a Gibbs sampling algorithm. It is very useful for showing the concept of Gibbs sampling to a reader who knows to read R source code. However, for practical purposes the execution is too slow.

The model parameters $\boldsymbol{\theta}^{(d)}$ and $\boldsymbol{\phi}^{(z)}$ can be obtained by the transformations

$$\hat{\phi}_j^{(w)} = \frac{n_j^{(w)} + \beta}{n_j^{(\cdot)} + V\beta} \quad \text{and} \quad \hat{\theta}_j^{(d)} = \frac{n_j^{(d)} + \alpha}{n_\cdot^{(d)} + T\alpha}. \tag{2.7}$$

This is Laplace smoothing (Zhai and Lafferty, 2004) on the samples. Since exchangeability (see: Definition 2.0.1) not only applies to the observations in a document, but also to the obtained topic distributions, model averaging cannot be applied and each estimate $\hat{\boldsymbol{\phi}}$ and $\hat{\boldsymbol{\theta}}$ is unique for the sample.

2.2 Infinite Mixture Models

Using heuristics or greedy algorithms to estimate the number of mixture components is unsatisfying, because it causes additional computational effort. It makes sense to treat the number of components as a function of the number of observations. This leads to a non-parametric view, in which the number of parameters grows with the data. The question becomes whether the LDA model can be extended in a non-parametric way. This requires a prior distribution different than the Dirichlet distribution, which is fixed in its dimensionality. Furthermore, this prior distribution should not come from a parametric family, but be a random measure on the space of all probability distributions on the word space. Additionally, it should be possible to apply inference methods on the posterior distribution.

2.2.1 Dirichlet Processes

For the purposes of topic modeling, it is sufficient to obtain discrete distributions. A model that provides the desired properties is the Dirichlet process. It was proposed by Fer-

guson (1973). Measures drawn from a Dirichlet process are discrete with probability 1; the process therefore defines a non-parametric prior distribution on the space of discrete distributions. The Dirichlet process can be defined in several ways. For the purpose of document modeling, three are of interest. First, the Dirichlet process arises as the model described by a stick-breaking construction. Second, it can be defined as a distribution over partitions of a measurable space by the Chinese Restaurant Process. And third, it turns out to be the limiting distribution as the number of mixture components of a finite mixture model increases and approaches infinity. All three views of the model are provided in Teh *et al.* (2006) and will be restated now.

The stick-breaking construction is a metaphor that describes how a draw from a Dirichlet process can be obtained. Let G_0 denote an arbitrary, not necessarily discrete, random measure and $\alpha_0 > 0$ a real number. Define independent sequences of i.i.d. random variables $(\pi'_k)_{k=1}^{\infty}$ and $(\phi_k)_{k=1}^{\infty}$ such that $\pi'_k|\alpha_0, G_0 \sim Beta(1, \alpha_0)$ and $\phi_k|\alpha_0, G_0 \sim G_0$. Further define $\pi_k = \pi'_k \prod_{l=1}^{k-1}(1 - \pi'_l)$. The sequence $\boldsymbol{\pi} = (\pi_k)_{k=1}^{\infty}$ adds up to 1 with probability 1. It thus defines a random probability measure on the natural numbers. The random measure G is then obtained as $G = \sum_{k=1}^{\infty} \pi_k \delta_{\phi_k}$, in which δ_x is an atomic measure giving mass 1 to the point x. It can be shown that $G \sim DP(G_0, \alpha_0)$ (Teh *et al.*, 2006).

The Chinese Restaurant Process (CRP), derived by Pitman (2006), represents another view of the Dirichlet process. Imagine a restaurant with infinitely many tables. When the first customer arrives, he will be assigned to the first table and choose the dish for that table from a menu. In terms of the Dirichlet process, that means that a sample $\phi_1 \sim G_0$ is drawn from the base distribution and assigned as the parameter for the first observation. The second customer to arrive will sit at the first table with probability $\frac{1}{1+\alpha_0}$ or take a new table with probability $\frac{\alpha_0}{1+\alpha_0}$. If he joins the first customer, he will have the same dish, meaning getting the same parameter assigned. If he gets a new table, he will generate a new dish (ϕ_2). In general, the probability of sitting at an already populated table is $p(\theta_n = \phi_i|\theta_1, \ldots, \theta_{n-1}) = \frac{n_i}{n-1+\alpha_0}$, the ratio of the number of customers sitting on that table and the number of customers in the restaurant. The concentration parameter α_0 influences how often a new table is chosen and therefore serves as an innovation parameter. The "tables" form a partition of the sample space and the described process is equivalent to a Dirichlet process with base measure G_0 and concentration parameter α_0. The discreteness of the Chinese Restaurant Process follows from the countability of the tables.

The last view of the Dirichlet process discussed here is that of the infinite limit of a finite mixture model. Consider the LDA model described earlier. If one increases the number

of mixture components and defines the Dirichlet parameter as $\alpha = \frac{\alpha_0}{K}$ then, as $K \to \infty$, LDA approaches a Dirichlet process. This is shown in Teh *et al.* (2006).

2.2.2 Hierarchical Dirichlet Processes

For the task of document modeling, each document is thought to be generated by a Dirichlet process. In this model, G_0 is the base measure on the word simplex. For convenience, this should be a Dirichlet distribution, resulting in a unimodal distribution over the space of multinomial distributions over the vocabulary. This model is very similar to the LDA model. The Dirichlet process G then represents the base measure.

Dirichlet Process Mixtures

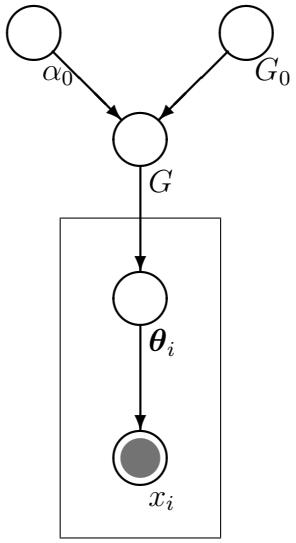

Figure 2.3: The Dirichlet process mixture model in plate notation (after: Teh *et al.*, 2006)

Dirichlet Process mixture models were introduced formally by Antoniak (1974). Figure 2.3 shows a Dirichlet Process mixture model in plate notation. G_0 is the base measure, e.g., a Dirichlet distribution with parameter λ, α_0 the concentration parameter, and G the Dirichlet process prior. $\theta_i \sim G_0$ is then the multinomial distribution over words that belongs to observation x_i. Because of the discreteness of the Dirichlet process, it is very likely that many observations are generated by the same multinomial distribution, which is interpreted as belonging to the same topic. However, the Dirichlet process mixture only allows the generation of a single document. It now seems appealing to allow each document to be generated

by a draw from the Dirichlet Process itself, that is, instead of having a single Dirichlet Process prior G, one has $G_1, \ldots, G_M$ for the M documents in the corpus, which are independent given G_0.

Hierarchical Approach

Having a Dirichlet Process prior for each document under the Dirichlet Process mixture model directly leads to a problem. Since G_0 is a continuous measure, G_i and G_j for $i \neq j$ have no atoms in common with probability one (Teh *et al.*, 2006). This means that topics are generated at the document level and not shared among the documents as desired.

One solution to this issue is to force G_0 to be a discrete measure, but that would be too restrictive. On the other hand, it is a well known fact that a Dirichlet Process generates discrete measures with probability one (Ferguson, 1973). Therefore, the proposed (Teh *et al.*, 2006) procedure is that G_0 is generated by a Dirichlet Process itself. G_0 is then a discrete and nonparametric measure on the multinomial distributions over the word simplex and there is positive probability for each topic (distributed according to G_0) to appear in any of the documents. The resulting model is presented in Figure 2.4. Teh *et al.* (2006) refer to this setting as the Chinese Restaurant Franchise. The metaphor is as follows: There is a number of restaurants; each has an infinite number of tables. All restaurants serve dishes from a global menu. When the first customer arrives, he will occupy the first table and the first dish is generated. Once it is generated, it will be on the global menu and therefore be available in all restaurants. The second customer in the same restaurant joins the first with probability $\frac{1}{1+\alpha_0}$ having the same dish or sits at a new table with probability $\frac{\alpha_0}{1+\alpha_0}$. If he sits at a new table, he will have the same dish as customer 1 with probability $\frac{1}{1+\gamma}$ and a new dish will be created with probability $\frac{\gamma}{1+\gamma}$. In general, if a new table is occupied, it will have an already existing dish assigned with probability proportional to the number of tables serving the same dish and a new dish with probability proportional to the concentration parameter γ.

For the document modeling task, dishes are associated with topics, which are drawn from the base measure H and shared among all documents. This nonparametric setting allows for a potentially infinite number of topics; the actual number can be learned from the data directly. Since the number of topics now is determined by a stochastic process, it makes sense to derive a probability distribution.

This can be done in two steps. First, it is necessary to obtain a probability distribution for the number of occupied tables (used topics) in a restaurant (document). Let $K(n)$ denote a Bernoulli random variable that takes the value of 1 if the nth word in a document generates a new topic in the document, or, correspondingly, if the nth customer occupies a new table

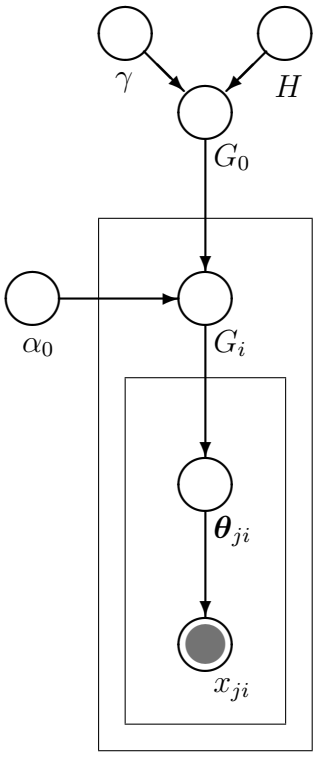

Figure 2.4: The HDP model in plate notation (after: Teh *et al.*, 2006)

in the restaurant. That is, $P(K(n) = 1) = \frac{\alpha_0}{n-1+\alpha_0}$. Let $X(N)$ denote the number of topics used in a document with N words. Then $X(N) = \sum_{n=1}^{N} K(n)$. Now derive an expression for $P(X(N) = k)$. For the case $N = 1$ this is trivial. For $N = 2$, the probability $P(X(2) = 1) = 1 \cdot \frac{1}{1+\alpha_0}$ and $P(X(2) = 2) = 1 \cdot \frac{\alpha_0}{1+\alpha_0}$. For $N = 3$, the expressions become a little bit more complex: $P(X(3) = 1) = 1 \cdot \frac{1}{1+\alpha_0} \cdot \frac{2}{2+\alpha_0}$, $P(X(3) = 2) = \frac{3\alpha_0}{(1+\alpha_0)(2+\alpha_0)}$ and $P(X(3) = 3) = 1 \cdot \frac{\alpha_0}{1+\alpha_0} \cdot \frac{\alpha_0}{2+\alpha_0} = \frac{\alpha_0^2}{(1+\alpha_0)(2+\alpha_0)}$. For arbitrary N and k, $P(X(N) = k) = s(N, k)\alpha_0^k \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_0+N)}$, in which $s(\cdot, \cdot)$ denote the unsigned Stirling numbers of first kind (Teh *et al.*, 2006). This expression, however, cannot be computed for large N, because both, the Stirling numbers and the evaluated Gamma function exceed machine precision. Therefore, a recursive formula is preferred:

$$P(X(N) = k) = \frac{1}{N-1+\alpha_0}(\alpha_0 P(X(N-1) = k-1) + (N-1)P(X(N-1) = k)) \quad (2.8)$$

In Equation 2.8, the relation to the Stirling numbers of the first kind can be seen easily using the recurrence equation $s(N,k) = s(N-1,k-1) + (N-1)s(N-1,k)$ with $s(1,1) = 1$ for $N,k > 0$ and $N \geq k$, otherwise $s(N,k) = 0$. This recurrence relation generates a triangular matrix that looks very similar to the Pascal triangle. Appendix B.1 presents an implementation of this algorithm for direct use in Matlab or Octave, which computes the probability mass function efficiently.

Since $K(n)$ is a Bernoulli random variable, $E[K(n)] = \frac{\alpha_0}{n-1+\alpha_0}$ and $V[K(n)] = \frac{\alpha_0(n-1)}{(n-1+\alpha_0)^2}$ (Billingsley, 1986). Because $K(n),\ K(m),\ m \neq n$ are independent, the expected value and variance for $X(N)$ can then be computed easily:

$$\mu = E[X(N)] = E\left[\sum_{n=1}^{N} K(n)\right] = \sum_{n=1}^{N} E[K(n)] = \sum_{n=1}^{N} \frac{\alpha_0}{n-1+\alpha_0} \tag{2.9}$$

and

$$\sigma^2 = V[X(N)] = V\left[\sum_{n=1}^{N} K(n)\right] = \sum_{n=1}^{N} V[K(n)] = \sum_{n=1}^{N} \frac{\alpha_0(n-1)}{(n-1+\alpha_0)^2} \tag{2.10}$$

For large numbers of words per document and large concentration parameters, a normal approximation can be found via moment matching. This gives an approximate distribution for the total number of tables in the Chinese Restaurant Franchise as the sum of occupied tables over all restaurants as the sum of M normal random variables. A better fit that also works for smaller number of words and/or concentration parameters is provided by the Gamma distribution. The parameters are estimated by moment matching as well, using $\beta = \frac{\sigma^2}{\mu}$ and $\alpha = \frac{\mu}{\beta}$. Figure 2.5 shows an example of the true distribution with overlayed approximating continuous densities. It was obtained using a document size of 20 words and $\alpha_0 = 1$, which is the setting for a long query or an abstract. One can see from the figure that the gamma distribution provides a better fit than the normal distribution, because it captures the skewness of the true distribution. The total number of draws from the base measure (total number of occupied tables) in case of a Gamma approximation is the sum of M independent Gamma random variables. This sum does not follow a Gamma distribution, but computation of the probability density function is still tractable (Moschopoulos, 1985). If the total number of occupied tables in the franchise is known, the distribution of the number of dishes (topics) is easily computed by Equation 2.8. If the total number of used topics is unknown, the probability distribution cannot be obtained in an easy way. The normal or gamma approximations,

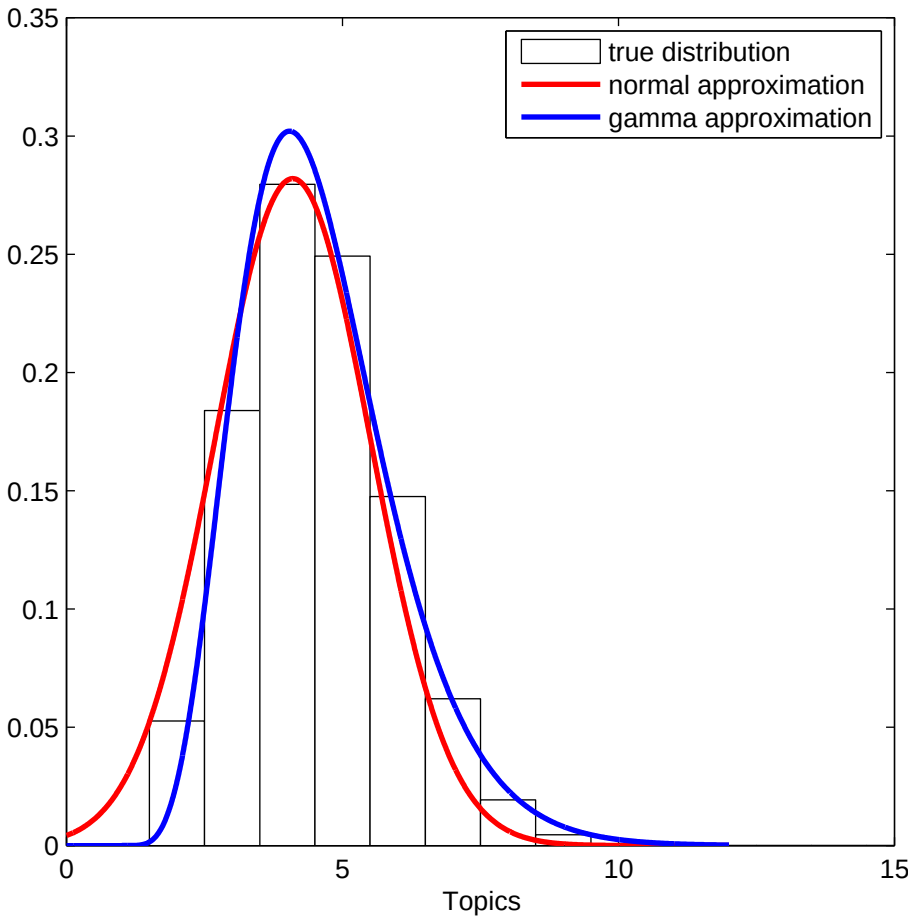

Figure 2.5: Approximation of the true probability $P(T = t)$ by continuous distributions

however, allow the expected value of the total number of topics in a corpus to be expressed as a function of a sum of random variables, which is tractable.

Inference and Learning on Hierarchical Dirichlet Processes

Teh *et al.* (2006) suggest that a Gibbs sampling scheme (see Section 2.1.3) with direct assignment performs best for inferring the posterior distribution in an HDP setup. This suggestion is also supported by experimental results provided by Teh *et al.* (2008).

For the direct assignment sampling scheme, tables in a restaurant are only represented as m_{jk}, the number of tables in restaurant j serving dish k. In the document modeling context that means the number of groups of words in document j sharing topic k. Further, let $r_{v,k}$ denote the number of times the word with vocabulary index v was assigned to topic k. Let $h(\cdot|\lambda)$ denote the prior density of the mixture components, in this case the density of a sym-

metric Dirichlet distribution with parameter λ. The conditional density of a word x_{ji} in topic k given all other assignments of topics to all words in the corpus and excluding the current assignment can be derived as

$$\begin{aligned}
f_k^{-x_{ji}}(x_{ji}) &= \frac{\int \phi_{x_{ji}}^{(k)} \prod_{j'i' \neq ji, z_{j'i'}=k} \phi_{x_{j'i'}}^{(k)} h(\boldsymbol{\phi}^{(k)}|\lambda) d\boldsymbol{\phi}^{(k)}}{\int \prod_{j'i' \neq ji, z_{j'i'}=k} \phi_{x_{j'i'}}^{(k)} h(\boldsymbol{\phi}^{(k)}|\lambda) d\boldsymbol{\phi}^{(k)}} \\
&= \frac{\int \phi_{x_{ji}}^{(k)} \prod_{v=1}^{V} \left(\phi_v^{(k)}\right)^{r_{v,k}^{-ji}} \frac{\Gamma(V\lambda)}{\Gamma(\lambda)^V} \prod_{v=1}^{V} \left(\phi_v^{(k)}\right)^{\lambda-1} d\boldsymbol{\phi}^{(k)}}{\int \prod_{v=1}^{V} \left(\phi_v^{(k)}\right)^{r_{v,k}^{-ji}} \frac{\Gamma(V\lambda)}{\Gamma(\lambda)^V} \prod_{v=1}^{V} \left(\phi_v^{(k)}\right)^{\lambda-1} d\boldsymbol{\phi}^{(k)}} \\
&= \frac{\frac{\prod_{v=1}^{V} \Gamma(r_{v,k}^{-ji} + \delta(x_{ji}, v) + \lambda)}{\Gamma\left(1 + \sum_{v=1}^{V} (r_{v,k}^{-ji} + \lambda)\right)}}{\frac{\prod_{v=1}^{V} \Gamma(r_{v,k}^{-ji} + \lambda)}{\Gamma\left(\sum_{v=1}^{V} (r_{v,k}^{-ji} + \lambda)\right)}} \\
&= \frac{\prod_{v=1}^{V} \Gamma(r_{v,k}^{-ji} + \delta(x_{ji}, v) + \lambda)}{\left(\sum_{v=1}^{V} (r_{v,k}^{-ji} + \lambda)\right) \prod_{v=1}^{V} \Gamma(r_{v,k}^{-ji} + \lambda)} \\
&= \frac{r_{x_{ji},k}^{-ji} + \lambda}{\sum_{v=1}^{V} r_{v,k}^{-ji} + V\lambda},
\end{aligned}$$

which also resembles the first factor of the Gibbs update in Equation 2.6. For the case of $k = k^{new}$, the density just becomes $f_{k^{new}}^{-ji}(x_{ji}) = \frac{1}{V}$. Let now β_k denote the overall popularity of topic k compared to all others. That is $0 \leq \beta_k \leq 1$ and $\sum_{k=1}^{K} \beta_k + \beta_u = 1$ in which u is the index of the next unseen topic. It can be seen directly that β_u will become smaller as the number of observations grows. In order to sample $\boldsymbol{\beta}$, all m_{jk} have to be known. In a Gibbs sampling environment, this is done by sampling. Antoniak (1974) derived a full conditional probability mass function for m_{jk} ignoring the assignment jk:

$$p(m_{jk} = m|\boldsymbol{\beta}) = \frac{\Gamma(\alpha_0\beta_k)}{\Gamma(\alpha_0\beta_k + n_{j\cdot k})} s(n_{j\cdot k}, m)(\alpha_0\beta_k)^m.$$

Here, $s(\cdot, \cdot)$ are the unsigned Stirling numbers of first kind and $n_{j\cdot k}$ is the number of words in document j sharing topic k or, following the metaphor, the number of customers in restaurant j having dish k. Again, a computationally more appealing recursive representation is

provided:

$$p(m_{jk} = m|\boldsymbol{\beta}, n_{j\cdot k}) = \frac{1}{n_{j\cdot k} - 1 + \alpha_0\beta_k}(\alpha_0\beta_k p(m_{jk} = m - 1|\boldsymbol{\beta}, n_{j\cdot k} - 1) + (n_{j\cdot k} - 1)p(m_{jk} = m|\boldsymbol{\beta}, n_{j\cdot k} - 1)).$$

Teh *et al.* (2006) show that $\boldsymbol{\beta} \sim \text{Dirichlet}(m_{\cdot,1}, \ldots, m_{\cdot,K}, \gamma)$ directly depends on the number of groups across all documents that have topic k assigned. Referring to the restaurant franchise metaphor this is the number of tables across all restaurants that serve dish k. Having all the pieces together, the actual sampling for topics can be conducted. Each word x_{ji} is now associated directly with a topic using an indicator variable z_{ji}. This indicator variable is then estimated using a Gibbs sampling scheme with the following update:

$$p(z_{ji} = k|\mathbf{z}^{-ji}, \mathbf{m}, \boldsymbol{\beta}) \propto \begin{cases} (n_{j\cdot k}^{-ji} + \alpha_0\beta_k) f_k^{-x_{ji}}(x_{ji}) & \text{if } k \text{ previously used,} \\ \alpha_0\beta_u f_{k^{new}}^{-x_{ji}}(x_{ji}) & \text{if } k = k^{new} \end{cases} \tag{2.11}$$

This concludes the inference task. An implementation in R is given in Appendix A.2.1. For the learning task, a sampling scheme called auxiliary variable sampling is employed. Teh *et al.* (2006) introduce binary variables s_j and continuous variables $w_j \in [0, 1]$ for each Dirichlet process (each document). Assuming that the concentration parameter α_0 has a gamma distribution with parameters a and b as prior leads to the following expression for the posterior distribution:

$$q(\alpha_0|\mathbf{w}, \mathbf{s}) \propto \alpha_0^{a-1+m_{\cdot\cdot}-\sum_{j=1}^{J} s_j} e^{-\alpha_0(b-\sum_{j=1}^{J}\log w_j)}. \tag{2.12}$$

This resembles a gamma distribution with parameters $a + m_{\cdot\cdot} - \sum_{j=1}^{J} s_j$ and $b - \sum_{j=1}^{J} \log w_j$. Since w_j and s_j are conditionally independent, given α_0, they can be sampled independently with distributions

$$q(w_j|\alpha_0) \propto w_j^{\alpha_0}(1 - w_j)^{n_{j\cdot\cdot}-1}, \tag{2.13}$$

which resembles a beta distribution with parameters $\alpha_0 + 1$ and $n_{j\cdot\cdot}$, and

$$q(s_j|\alpha_0) \propto \left(\frac{n_{j\cdot\cdot}}{\alpha_0}\right)^{s_j}, \tag{2.14}$$

which is proportional to the probability mass function of a Bernoulli random variable. The hyperparameter γ can be obtained in the same way, using K instead of $m_{j\cdot}$ and $m_{\cdot\cdot}$ instead of

$n_{j\cdot\cdot}$. Given the last steps, it is an easy task to implement a system that models the documents in a corpus and learns the parameters α_0 and γ. That the gamma distribution is a natural choice as a prior for α_0 and γ has been shown by Blei and Jordan (2006). Appendix A.2.2 shows how the auxiliary sampling scheme can be implemented.

CHAPTER 3:
Application to Search

This chapter shows how a document model can be used in information retrieval tasks. It starts out by producing robust point estimates from probabilistic topic models. Then an alternative, purely keyword-based, model is introduced and "mixed" with the topic model. This mixture is driven by a mixing weight that determines how much influence the topic model in the combined model has.

The keyword-based model is used because related research (Wei and Croft, 2006) suggests that a probabilistic topic model itself is too coarse to obtain good information retrieval performance. A topic model, however, adds another aspect to keyword search by not just assessing whether a certain keyword is contained in the document but also evaluating the correlation between words. Thus, a relevant document can be returned by the search application, even if it does not share any terms with the query.

The chapter finishes by introducing how document relevance is determined and documents are ranked.

3.1 Combining Estimates from Different Markov Chains

If a Gibbs sampler (see Chapter 2) is used for inference, each individual run represents a Markov chain that, after convergence, produces a point estimate for the type-topic distribution (ϕ) and the topic document distribution (θ). To get a more robust estimate for the distributions, several runs of a Gibbs sampler with different random seeds should be combined.

It is tempting to define $\hat{\boldsymbol{\theta}} = \frac{1}{n}\sum_{k=1}^{n}\hat{\boldsymbol{\theta}}_k$, in which $\hat{\boldsymbol{\theta}}_k$ is the point estimate from each particular run of the Gibbs sampler. The point estimate $\hat{\phi}$ would be obtained similarly. This is, however, not feasible because of the exchangeability of topics (see Definition 2.0.1). That means, the order of topics is not predetermined in advance and there is no way of controlling that order during the Gibbs sampling runs. The attempt to reorder the topics after the runs imposes a matching problem, which is not trivially solved.

To use the computed distributions for search, it is sufficient to have a point estimate for the probability of a word being contained in a document: $\hat{p}_{topic}(w|d)$. In LDA- and HDP-

based models, this probability is computed as

$$p_{topic}(w|d) = \sum_{z=1}^{T} p(w|z)p(z|d). \tag{3.1}$$

Because of the commutativity of the addition, the order of the topics is not relevant at all. To compute the actual matrix of multinomial distributions for the documents, Equation 3.1 represents a simple matrix multiplication for the point estimates of each particular Gibbs sampler run.

The point estimate for the word document distributions ($\hat{\boldsymbol{\zeta}}$) is therefore the combination of the matrix products from each run of the Gibbs sampler:

$$\hat{\boldsymbol{\zeta}} = \frac{1}{n} \sum_{k=1}^{n} \hat{\boldsymbol{\phi}}_k \hat{\boldsymbol{\theta}}_k. \tag{3.2}$$

3.2 A Keyword-based Language Model

In addition to the index generated by the probabilistic topic model, either LDA- or HDP-based, it is necessary to compute a point estimate for the probability of a word being contained in a document based on the documents directly. The method that is used for this is called Bayesian Smoothing using Dirichlet Priors as described by Zhai and Lafferty (2004). This model is also used in Wei and Croft (2006) and is computed as follows:

$$p_{Dirichlet}(w|d) = \frac{c(w,d) + \mu \sum_{d'} \frac{c(w,d')}{|d'|}}{|d| + \mu}. \tag{3.3}$$

Here, μ is the smoothing parameter, $c(w,d)$ is a function that counts how often a token of type w appears in document d, and $|d|$ denotes the number of words in the document. The method is called Dirichlet smoothing, because $p_{Dirichlet}(w|d)$ is the maximum a posterior (MAP) estimate of a Dirichlet-Multinomial model with prior parameter $\mu \sum_{d'} \frac{c(w,d')}{|d'|}$ and the document d as evidence.

3.3 Ranking Documents

A search application evaluates the relevance of every document in the corpus and then returns an ordered list of documents or pointers to documents. This process is called ranking. This section introduces different methods that can be used to rank documents in a

probabilistic topic model. For the implemented prototype, all methods were implemented. Predictive likelihood, however, performs best by far.

3.3.1 Ranking Documents Based on Predictive Likelihood

The ranking of the documents in return to a user query is determined by the predictive likelihood, which roughly can be seen as the probability that the query Q was generated by the model of the document d. More formally:

$$p(Q|d) = \prod_{q \in Q} p(q|d) = \prod_{q \in Q} \left(\lambda p_{Dirichlet}(q|d) + (1-\lambda) p_{topic}(q|d)\right), \tag{3.4}$$

in which λ determines the weighting between the keyword-based model and the topic-based model. A document that has a high predictive likelihood will get assigned a high rank. At this point it becomes apparent that all probabilities need to be strictly positive; otherwise the product will be zero and even a very relevant document can end up with the lowest possible ranking.

3.3.2 Ranking Based on Topic Distributions

For a query that is supplied by a user, the most likely topic distribution under the probabilistic topic model can easily be determined. In case of an LDA model, Variational methods (Blei *et al.*, 2003), Expectation Propagation (Minka and Lafferty, 2002), or Gibbs sampling are suitable. For the HDP-based model, Variational methods (Teh *et al.*, 2008) or Gibbs sampling (Teh *et al.*, 2006) can be used. In either method, the result will be a multinomial distribution over the topics, which looks very similar to the multinomial distribution over topics that is inferred for each document in the corpus.

The ranking of the documents now can be determined based on the similarity of the topic distribution of the query and the topic distribution of a document. In the experiments, however, all similarity measures on the topic distribution (angle and divergence measures) were outperformed by predictive likelihood.

Cosine Similarity

A multinomial distribution over topics can be interpreted as a vector in Euclidean space, in which the number of different topics determines the number of dimensions. The angle between the query vector and the document vector can be used as a similarity measure. An angle of zero would mean the query and the document have exactly the same topics in the same proportions, thus giving the document the highest possible rank, whereas an angle of

90 degrees means that query and the document do not share any topics.

Instead of computing the angle, it makes sense to consider the cosine of the angle, which maps orthogonal vectors to zero and vectors with the same direction to one. The cosine similarity of a query q and a document d is then defined as:

$$sim(q, d) = \frac{\langle \theta_q, \theta_d \rangle}{\|\theta_q\| \|\theta_d\|},$$

in which $\langle \cdot, \cdot \rangle$ denotes the cross product and $\|\cdot\|$ the Euclidean norm.

Kullback-Leibler Divergence

Since multinomial distributions over topics are probability measures, their distance for a query Q and a document D can be determined using the Kullback-Leibler (KL) divergence. It is defined as:

$$D_{KL}\left(p(z|Q) \| p(z|D)\right) = \sum_{z=1}^{T} p(z|Q) \log \frac{p(z|Q)}{p(z|D)}$$

From the definition, it is obvious that neither $p(z|Q)$ nor $p(z|D)$ can be zero for any topic z. Further, it is clear that $D_{KL}\left(p(z|Q) \| p(z|D)\right) \geq 0$ with equality if $p(z|Q) = p(z|D)$. This is different from the cosine similarity, in which the value is larger, if the vectors are more similar. In KL divergence, large numbers express a high distance between the measures.

The definition of KL divergence shows that it is not a symmetric measure. To get around this issue, the Jensen-Shannon (JS) divergence was defined as the arithmetic mean of the two possible KL divergences:

$$D_{JS}\left(p(z|Q) \| p(z|D)\right) = \frac{1}{2}\left(D_{KL}\left(p(z|Q) \| p(z|D)\right) + D_{KL}\left(p(z|D) \| p(z|Q)\right)\right).$$

3.4 Steps Towards Implementation

Computing the keyword-based model and averaging topic models can easily be implemented as matrix operations. Most general purpose languages provide efficient data structures and algorithms for basic matrix algebra. Predictive likelihood is very efficient if the query is represented by a sparse vector structure. The ranking then reduces to a few table lookups and multiplications per document followed by a sorting algorithm on the relevance scores, which is $O(n \log n)$. Similarly, cosine similarity and divergence measures have efficient implementations, although they did not perform well in the preliminary experiments.

Altogether this allows for practical implementations for online search and ad hoc retrieval.

CHAPTER 4: Implementation

This chapter will illuminate some of the implementation steps that were necessary for building a search engine based on probabilistic topic models. For the implementation, MALLET (McCallum, 2002) was used. MALLET is an open source toolkit, written in Java. It provides most of the functionality that is needed for document modeling, clustering, and classification tasks.

Matrix computations are implemented using COLT, a matrix API for Java, built by the CERN institute (Hoschek, 2004). It performs standard matrix operations like addition and multiplication for large matrices with double precision numbers very efficiently. In addition, it provides a very fast and memory-efficient implementation of sparse matrices, which are used frequently in natural language processing tasks.

The prototype implementation is built into the framework of Hawkins' (2009) search application and can be used as a search module in this modular framework. Java is used as the implementation language, allowing the use of well-developed APIs for text processing, matrix algebra, and stochastic processes.

Examples in this chapter will mostly refer to the CRANFIELD benchmark corpus. For a discussion of the improvements of the implemented prototype on other corpora see Chapter 5.

The source code documentation provides a more detailed description of methods and fields that are used by the application.

4.1 Preprocessing

It is important that queries and documents are preprocessed in the same way to ensure that string comparison leads to correct results. That is, the vocabulary for a document in the corpus has to be exactly the same as for a query.

In MALLET there is a special class, the Pipe-class, which fullfills the preprocessing task. Pipe is an abstract class, which is extended by several classes, each of them carrying out a particular preprocessing step. Every document is then "piped" through a list of these Pipe-objects and the end result is a preprocessed document that can be used for index building.

The idea of using a pipe makes MALLET very flexible and attractive for text pro-

cessing tasks. It allows us to arrange processing modules in a list and send the raw text data through all necessary steps from creating a character sequence, then a token sequence, remove stopwords and so on.

4.1.1 Tokenizing

The first step in preprocessing raw documents is to tokenize them. This step splits a character sequence into a token sequence, typically on white spaces and special characters like commas or periods. With MALLET, after reading the character stream, the sequence is then processed through a pipe, in which a regular expression pattern is applied on the character stream to remove special characters and digits.

4.1.2 Stemming

Stemming maps inflections of a word to a common stem. That is, "running," "ran," and "run" are all mapped to "run." One of the most common stemming algorithms is the Porter Stemmer (Porter, 1980). A Porter stemmer applies a small number of rules of the form "(condition) S1 $\rightarrow$ S2" on every token that meets the condition and has form S1 and changes it to S2. Stems found by a Porter stemming algorithm do not necessarily agree with the linguistic stem of a word. For example, "happy" will be stemmed to "happi," which would not be called a stem by a linguist. However, stemming reduces the number of distinct words in a vocabulary, thus reducing noise. Unfortunately, this comes at the cost of introducing additional ambiguities into the documents.

For a probabilistic topic model like LDA- or the HDP-based model, these additional ambiguities result in stronger correlation between similar documents, thus improving the recall in the information retrieval task. For the example of the CRANFIELD data set, a Porter stemmer leaves 1838 types total. An additional advantage of applying a stemming algorithm is the reduced storage requirement for the final index. Without stemming, the CRANFIELD index requires 79 MBytes on the hard disk; stemming reduces its size to 53 MBytes.

For the experiments in this thesis, a Porter stemming algorithm was implemented as a subclass of a MALLET pipe, such that it could be easily integrated as preprocessing step.

4.1.3 Stopword Removal

Stopword removal allows the omission of very frequent terms that are shared in almost all documents with high probability. These words (*e.g.,* "and," "the," "a") do not carry meaning and are therefore not useful for information retrieval tasks. If they are left in the document, they consume computation time and add noise to the model. Therefore, it is standard

procedure to remove them in advance. The English stop word list that was used throughout the experiments contains the 571 most common words in English documents. MALLET provides a pipe implementation that removes stopwords as part of the preprocessing.

The CRANFIELD corpus consists of 7045 unique words; after removing stopwords, 6639 remain. This was measured without stemming first. For the actual implementation, however, it is important to stem first and then apply stopword and rare type removal.

4.1.4 Removing Rare Types

Rare types impose a different issue. Since they show up in the corpus in only one or two documents, it is practically impossible to compute the correlation with other words. Additionally, removing rare types reduces the chance of having missspelled words in the index after preprocessing.

Of the 6639 types that remain in the Cranfield corpus after stopword removal, 4112 are used fewer than 5 times throughout the corpus. Removing these leaves an index with 2527 types total. Having a smaller number of types results in a smaller index, which can be stored in memory directly.

4.1.5 Building Termvectors

For the Expectation Propagation inference algorithm, each document and query needs to be represented as a term vector. A term vector is a vector of length V, the size of the vocabulary, whose entries are the number of tokens of type w in the document. MALLET uses for this purpose an additional pipe, the "FeatureSequence2FeatureVector" class. However, for Gibbs sampling, a term vector representation is not suitable; a term sequence has to be used instead and this preprocessing step must be skipped.

4.2 Building the Index

For the purposes of this thesis, the index is just a matrix $V \times D$ in which V is the number of types and D the number of documents in the corpus. Each column of this matrix represents a multinomial distribution over types, which means that each document is represented as a vector on the V–1-simplex. In order to be able to use the predictive probability for determining the relevancy of a document for a particular query, this matrix has to be dense, that is, it cannot have any zero values.

4.2.1 Building the LDA Index

MALLET implements a very robust and efficient version of an LDA document model, which uses a parallel Gibbs sampling algorithm. This allows the use of several Markov chains to estimate the true topic distribution for each document. For the search engine implementation, the number of chains is a parameter that can be supplied by the user.

After convergence, an estimate of the document topic distribution and the type topic distribution is computed using Equation 2.7. Since topics are exchangeable, it is not possible to average $\hat{\theta}$ or $\hat{\phi}$ from different samples or even different chains. However, the final probability distributions $\hat{p}(w|d) = \sum_z \hat{\phi}_w^{(z)} \hat{\theta}_z^{(d)}$, the probability of a word occuring in a particular document under the LDA model, can be averaged.

The implementation estimates $\hat{p}(w|d)$ for each Markov chain and averages these estimates. This results in a $V \times D$ matrix, which holds the aggregated probabilistic topic model of the corpus.

4.2.2 Building the HDP Index

In order to estimate the topic distribution under an HDP-based model, the implementation by Teh *et al.* (2006) was used. The code runs in Matlab, and Octave was used as an intermediate step to convert the binary Matlab format into a csv file, which then was imported by a Java class.

The type-topic distribution estimate ϕ is then computed in exactly the same way as in Equation 2.7. For the topic-document distribution, there is no equivalent to the hyperparameter α in the LDA model. Therefore, smoothing was ignored. The final distribution, however, cannot have any zero values because the type-topic distribution will not contain any zero values and every row in the topic document distribution will have at least one value greater than zero.

As with the LDA model, the estimates from several chains are averaged to obtain a more robust estimate of the probability distribution $\hat{p}(w|d)$.

4.2.3 Building the Smoothed Language Model

For the keyword-based search, it is necessary to implement a language model that is based on word counts only. In order to use predictive probability for ranking documents, this language model needs to be smoothed. The formula for estimating the language model, Equation 3.3 is used.

Computing this probability distribution can be done in a single pass over the term

vectors and results in a $V \times D$ matrix, in which every column represents the probability distribution over word types for a particular document. It therefore has the same structure as the probability distributions computed by the probabilistic topic models.

4.2.4 Building the Final Index

The final index then is simply the weighted average between a probabilistic topic model and the smoothed language model:

$$p(w|d) = \lambda p_{Dirichlet}(w|d) + (1 - \lambda) p_{LDA/HDP}(w|d), \tag{4.1}$$

in which $0 \leq \lambda \leq 1$ is the weighting parameter. This is implemented by a simple matrix addition, which produces a dense matrix of size $V \times D$.

The index also stores the averaged matrices from the probabilistic topic model and the smoothed language model. This allows changing the values for λ and μ later, after the index has been built, and thus prevents having to retrain the whole model.

4.3 Implementation of the Search and Ranking Algorithm

4.3.1 Preprocessing the Query

The query is provided by the user of the search application as a simple string. It needs to be tokenized, stopwords and rare types need to be removed, and the words need to be stemmed, before the query is finally turned into a term vector.

For this task, a separate pipe, the "query pipe," is used in the implementation. It is important that the query pipe use exactly the same preprocessing steps as are used on documents. It is also necessary that the query pipe have the same word dictionary (in MALLET called the alphabet) as the pipe for the documents had, because the terms in the term vector need to have the same index. That is, if in the document corpus the term "experiment" has the index 5, this needs to be true for the query as well. Otherwise the matrix lookup will return the wrong result.

However, the original word set must not grow, if the query contains a word that is not contained in the corpus. Therefore, the alphabet must not be supplied by reference to the original object. In the implementation, the alphabet is cloned first and the reference of the clone is passed to the query pipe.

4.3.2 Computing the Ranking Score

After computing the combined index as a weighted average between the probabilistic topic model and the smoothed language model and turning the user query into a term vector, computing the ranking score is straightforward by applying Equation 3.4. However, multiplying a sequence of small probabilities can lead to numerical instabilities. Therefore, the natural logarithm is used instead:

$$score_Q(D) = \sum_{w \in Q} c(w, Q) \log p(w|D),$$

in which $c(w, Q)$ counts how often the term w appears in the query Q. This count will typically be one, since words are rarely repeated in a query. If, however, the query is a full paragraph or a question, it might happen that terms are repeated. For the similarity measure, this means that documents that share that particular word or that have many words that are correlated with this word, will get a higher ranking. Repeating terms in the query thus has a boosting effect for that particular term.

MALLET's term vector implementation is very useful for this computation because it is very memory efficient by storing only two arrays of integers. One holds the vocabulary indices of all non-zero count terms, the other holds the actual counts.

After computing the scores for all documents, the documents are ordered accordingly and returned to the user.

4.4 Maintaining the Index

This section describes how the special cases of adding and deleting documents are dealt with. Both cases only allow for marginal changes on the corpus. If many documents are added or removed over time, it is best to retrain the indexer. In case of the LDA-based model, this takes only a few minutes, whereas for the HDP-based model this requires some hours and is more involved.

4.4.1 Adding Documents

As with a query, the new document has to be preprocessed. After this step, a topic distribution is inferred. For the LDA implementation, MALLET's TopicInferencer is used. For HDP, the document is treated as a test document versus the rest of the corpus. In both cases, a Gibbs sampler generates the necessary topic distribution. This new distribution over

topics is now multiplied by the already existing word-topic distribution, which results in a vector of length V, the size of the vocabulary.

Additionally, the smoothed language model for the document is computed, again using the known base distribution. Finally, the two obtained vectors are averaged according to the defined weighting scheme and the final result is attached to the index.

4.4.2 Removing Documents

Removing a document from the corpus is implemented in the easiest possible way. First, the document is removed from the feature seqence list. Then the corresponding columns in the topic model, language model, and combined index are removed. While this ensures that the removed document is not accidently returned by the search application, it leaves the base distributions from the probabilistic topic model and the smoothed language model untouched. For a single document, this is not critical, because a single document does not have a huge influence on the base distributions. If, however, many documents or a very big document are removed, the index should be generated again with the new corpus.

4.5 Evaluation

This section describes how benchmark queries, for which the relevant documents are known, are implemented in a way such that the retrieval performance of the prototype can be computed. Additionally, it describes, which methods allow the computation of performance measures.

4.5.1 Query Representation

Each query is wrapped into an instance of class "Query." Every instance holds an internal identifier and a data set identifier. This is necessary because the benchmark data sets do not provide a consecutive numbering of the queries. The third field that is maintained in the class is a single string value, which represents the actual query. Since the query object can be used in any search module that fits in the search framework, this string is not preprocessed.

In order to allow performance assessment, each query holds a list of document identifiers, which contains all the documents that are labeled relevant to the query.

4.5.2 Query Set Representation

An instance of class "QuerySet" is a container for "Query" instances.

The class provides a static method that reads text files with queries and query-document relevance pairs to generate the query set. As an instance method, "trimToSize()" shall be men-

tioned here. It removes irrelevant queries from the query set. A query is called irrelevant if it does not have relevant documents assigned to it.

4.5.3 Computing Performance Measures

The main class for computing performance measures is the "Evaluator" class. An instance of this class is generated with an object of class "ModularSearchEngine", a "ModuleMixer" object (Hawkins, 2009), a "QuerySet" instance, and an integer indicating the maximim number of documents that the user wants to be retrieved.

An "Evaluator" instance provides methods to compute average precision for each query ("computeAveragePrecision()") and the mean average precision over all queries ("computeMeanAveragePrecision()"). Additionally, it computes a confusion matrix for each of the queries and at every possible number n, $1 \leq n \leq N$, in which N is the maximum number of documents retrieved.

CHAPTER 5:
Evaluation

5.1 Document Modeling Experiments

This section describes exeriments that were used in the document modeling context. It shows how topics emerge and can be interpreted and how hyperparameters and the number of topics can be estimated to obtain a document model with high likelihood.

5.1.1 Topic Detection

Griffiths and Steyvers (2004) provided a Matlab toolbox that uses Gibbs sampling to obtain samples from the posterior distribution of the latent topics. This toolbox was the basis for first experiments on topic detection and parameter estimation. For a smaller corpus with artificial data, the LDA Gibbs sampler was implemented in R (see Appendix A.1.2) and used for small scale experiments which are not discussed.

The first experiments we ran on a stratified sample of 1,000 document from the Wikipedia[1] collection. Table 5.1 shows the 10 most likely words in five sample topics based on a single Markov chain of the Gibbs sampler and 200 topics. The smoothing constant was set to $\beta = 0.01$. The documents were preprocessed as described in Section 4.1 with the exception of stemming. For this experiment, it was important to get topics with readable words, which is not possible if stemming is applied.

5.1.2 Estimating the Number of Mixture Components

Griffiths and Steyvers (2004) suggest a hill climbing method to estimate the number of topics in a given corpus. This requires the hyperparameters α and β to be known and fixed. The idea is to compute the posterior probability of a corpus given the number of topics, $P(\mathbf{w}|T)$. Unfortunately, this is intractable, since it requires the computation of $P(w|T)$ for any conceivable topic distribution. The topic distribution itself is a draw from a continuous Dirichlet distribution and therefore the number of possible topic distributions is uncountable. Griffiths and Steyvers suggest running a Gibbs sampler on the model with different Markov chains and estimating the resulting posterior probability $P(\mathbf{w}|\mathbf{z})$ from the samples of each

[1] Wikimedia Foundation Inc. Wikipedia: The Free Encyclopedia. `http://download.wikimedia.org/enwiki/latest/`. Online, last accessed 28 April 2008

Topic 2	Topic 3	Topic 13	Topic 28	Topic 51
ball	treatment	software	god	greek
play	medical	computer	christian	zeus
team	acupuncture	hardware	chruch	mythology
player	disease	video	jesus	gods
football	pain	disk	christianity	god
line	studies	computers	believe	son
offensive	evidence	memory	book	aeneas
defensive	effects	bit	christ	myth
pass	found	operating	holy	goddess
field	patients	screen	faith	temple

Table 5.1: Sample topics in Wikipedia

chain. These probabilities are then averaged by applying the harmonic mean: $\hat{P}(\mathbf{w}|T) = \frac{K}{\sum_{k=1}^{K} \frac{1}{P(\mathbf{w}|\mathbf{z}_k)}}$, in which K denotes the number of samples taken. The number of topics, that maximizes $\hat{P}(\mathbf{w}|T)$ will then be accepted. This procedure was applied in our research with 8 Markov chains and 10 samples for each chain, giving $K = 80$ on the Wikipedia corpus with $\beta = 0.01$.

An alternative approach uses averages over the natural logarithm of the likelihood and has better numerical stability. Figure 5.1 shows how this method works on the CRANFIELD data set based on 10 samples per number of topics. The concavity of the likelihood function becomes even more obvious with more samples taken per topic value. The gap between two design points should be at least 10 because there is almost no difference in likelihood between two models with a specified number of topics that only differs by one or two. For all practical purposes, it is sufficient to determine the number of topics rounded to the closest multiple of ten.

5.1.3 Number of Iterations for the Gibbs Sampling Algorithm

Determining the optimal number of iterations for burn-in and lag between samples for a Gibbs sampler is not trivial and still an open field of research. Furthermore, it is a hard task to determine if the Gibbs sampler converged to the target distribution (Brooks and Roberts, 1998). The major problem is that a Gibbs sampler may spend many iterations in a local optimum before it finally converges to the right solution. Since it is a stochastic algorithm, this number of iterations cannot be predicted precisely. Results to determine bounds on the number of iterations exist for a few special cases, which do not include LDA- or HDP-based

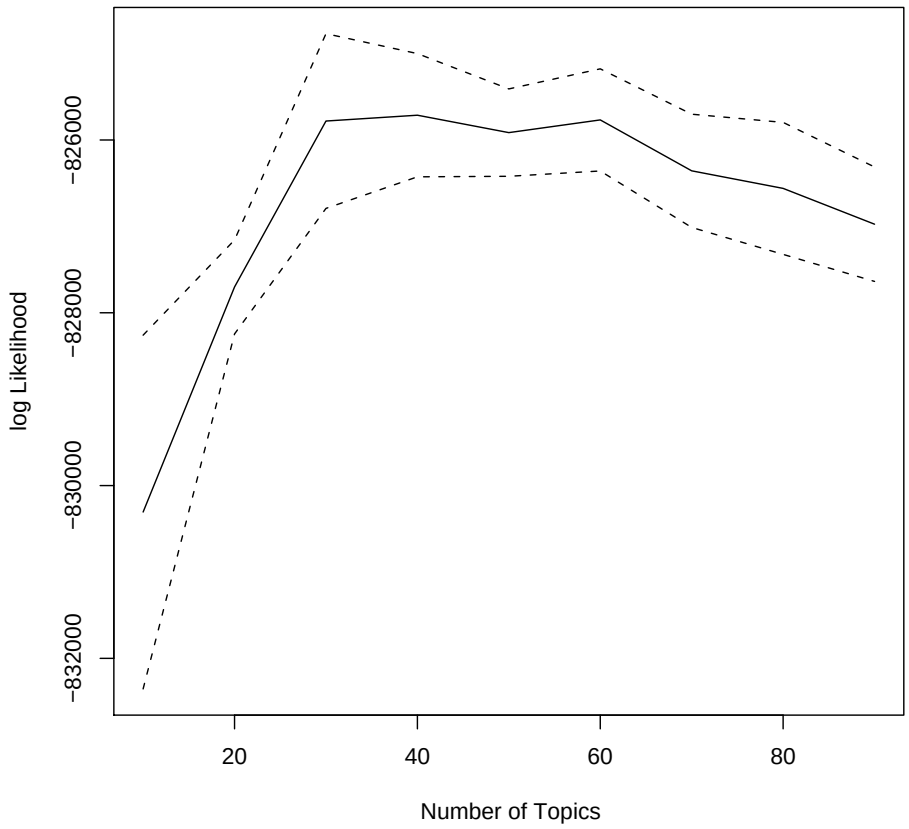

Figure 5.1: Optimal number of topics for the CRANFIELD data set. The solid line represents the likelihood function, while the dashed lines represent the bounds of a 95%-confidence interval. For every number of topics, 10 independent Gibbs sampling runs were used. The optimal number of topics will be determined as 40, given a smoothing parameter of $\beta = 0.007$.

models. Raftery and Lewis (1992) suggest that this number should be less than 5,000. In the LDA and HDP literature, typical numbers are between 1,000 and 2,000.

Figure 5.2 shows the typical behavior of the corpus likelihood $P(D)$ during a run of a single Markov chain. At about 150 iterations, a local maximum can be observed, from which the likelihood first drops, before it climbs to a pretty much stable value after 1,200 iterations. Figure 5.2 was produced by an HDP model based on the CRANFIELD data set. The suggestion is to use at least 1,000 iterations as burn-in time and then at least 20 samples with a lag of at least 100 iterations. These settings regularly produced good results in the experiments, whether LDA-based or HDP-based.

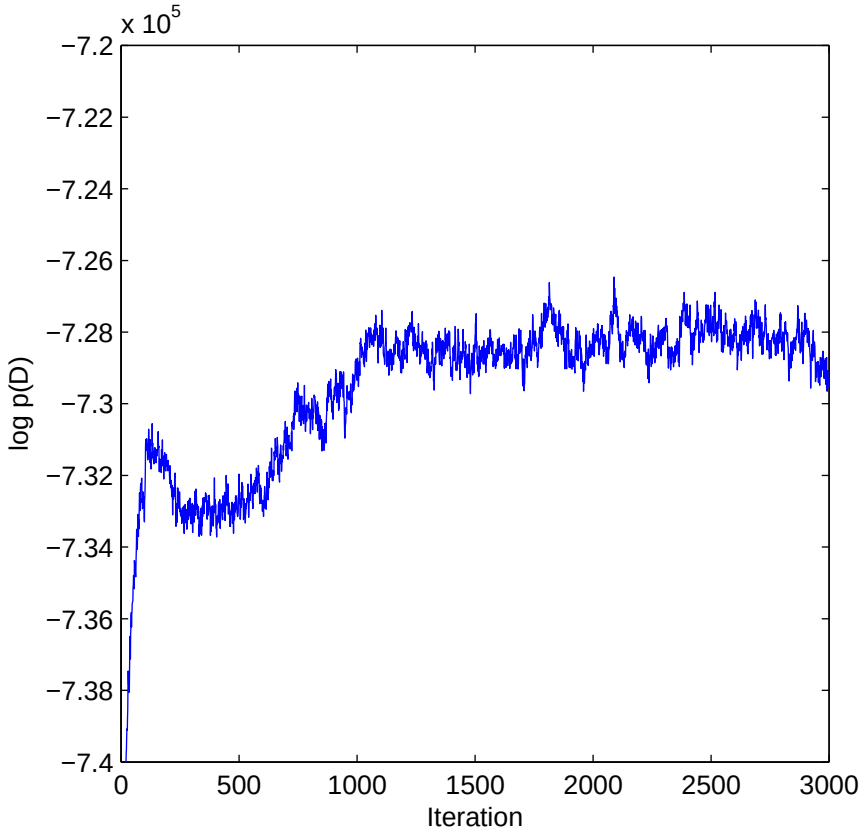

Figure 5.2: Typical Likelihood behavior of a corpus D for a Gibbs sampler in the HDP setting

5.2 Information Retrieval Experiments

Table 5.2 shows how a probabilistic topic model can improve retrieval results. As an example, query 153 from the CRANFIELD benchmark corpus is used and for each method the top ten documents were considered. The table only contains relevant documents. The document 1083 does not share keywords with the query and therefore it will not be returned by any method that bases on keyword search alone. A probabilistic topic model alone however would return too many documents that are not relevant to the query. In Table 5.2, the fourth column shows how a combination of both methods, keyword search and a probabilistic topic model, can improve retrieval performance. Though this is just a single sample, it shows in which way a probabilistic topic model and a keyword-based search augment each other.

Query	pure LDA	pure VSM	mix with $\lambda = 0.8$
153	1078	1081	1078
	1082	1082	1081
	1083	1085	1082
	1085		1083
			1085

Table 5.2: Comparison of pure LDA, pure VSM, and a combination of both with $\lambda = 0.8$ based on query 153 of the CRANFIELD benchmark corpus.

5.2.1 Evaluation Metric

As the metric to compare the retrieval performances of different models and/or different parameter sets, the mean average precision (Robertson, 2008) is used. The average precision for a single query is defined as

$$AP = \frac{1}{R} \sum_{k=1}^{D} AP_n,$$

in which R is the number of total relevant documents and D denotes the total number of documents in the corpus. The contribution of document d_n to the average precision AP_n is defined as

$$AP_n = \frac{1}{n} \sum_{m=1}^{n} \delta_{m,n},$$

in which $\delta_{m,n} = 1$, if the documents d_n and d_m are both relevant to the query and $\delta_{m,n} = 0$ otherwise. The mean average precision is then the mean of the average precision values over all queries.

Mean average precision was chosen because it is the metric used in related research on the same benchmark data. A Java implementation of the algorithm is presented in Appendix B.3.1.

5.2.2 Baselines

For each benchmark corpus, a baseline was defined. This baseline comes from published Information Retrieval papers (Roussinov and Fan, 2006) or, in case of the TIME MAGAZINE data set, from the best experiment that only uses keyword search and that applied stemming, stop word and rare type removal before training the index. Table 5.3 shows the corpus statistics and the used baselines.

Corpus	Number of documents	Number of relevant queries	Mean average precision baseline
CISI	1460	76	0.200
CRANFIELD	1398	225	0.392
MEDLINE	1033	30	0.518
TIME MAGAZINE	423	83	0.526

Table 5.3: Corpus statistics and baselines for the information retrieval experiments.

5.2.3 Effects of Removing Rare Types

Removing types that occure infrequently hurts the average precision in the retrieval task. Table 5.4 shows the differences in retrieval performance with rare type removal and without. While the retrieval performance with rare types is slightly higher, the storage requirements for the index grow drastically.

An analysis on the aggregated values for mean average precision leads to overlapping confidence intervals and therefore no statistical significance. If the analysis is conducted on the difference in average precision for each query, the result changes. The fourth column in Table 5.4 shows the p-values computed by a Wilcoxon signed rank test based on the differences in average precision per query. The alternative hypothesis is that with rare types in the corpus, the average precision is greater than without. A Wilcoxon signed rank test conducted on all queries regardless of the corpus results in a p-value of 0.005, which shows significance even at the one percent level. Of course, this result cannot be extrapolated to an unseen corpus.

Experiments for the values in Table 5.4 are generated with smoothing factor $\mu = 700$, 10 Markov chains, 345 topics, 800 iterations for the Gibbs sampler and 8 parallel threads for the topic model estimation. The prior parameter for LDA was fixed at $\beta = 0.01$ and the mixing proportion was $\lambda = 0.7$.

Corpus	with rare types	without rare types	p-value	storage difference
CISI	22.80%	22.05 %	<0.01	71 MBytes
CRANFIELD	44.02%	42.30%	0.03867	42 MBytes
MEDLINE	59.28%	59.91%	0.5803	108 MBytes
TIME MAGAZINE	59.28%	54.82%	0.037	90 MBytes

Table 5.4: Effects of removing rare types applied to the different corpora. The metric is mean average precision. The p-value is computed per corpus based on direct comparison of the query results

5.2.4 Influence of Stemming

The influence of stemming was studied on an LDA based model mixed with the smoothed language model at a weighting coefficient of $\lambda = 0.7$. The prior parameter on the type topic distribution was $\beta = 0.01$ and the smoothing constant $\mu = 700$. It was trained with $K = 345$ topics and 10 Markov chains. Table 5.5 shows the results in mean average precision and storage reduction for the index after applying stemming. The analysis is the same as described in Section 5.2.3.

For CISI, CRANFIELD, and MEDLINE, the test shows significant performance increase at the 10 percent level, but not at the five percent level. For TIME MAGAZINE, the test does not show significance. Applied to the set of all queries, the Wilcoxon signed rank test returns a p-value of 0.002, which shows significance at the one percent level. Again, it is likely that this result does not apply to an unseen corpus.

Corpus	with Stemming	without Stemming	p-value	Storage Reduction
CISI	22.05%	20.03 %	<0.01	17 MBytes
CRANFIELD	43.29%	42.30%	0.075	26 MBytes
MEDLINE	59.91%	58.14%	0.057	10 MBytes
TIME MAGAZINE	54.82%	52.53%	0.222	12 MBytes

Table 5.5: Effects of stemming applied to the different corpora. The metric is mean average precision.

5.2.5 Influence of the Number of Markov Chains

The number of independent Gibbs sampler runs heavily influences the quality of the probabilistic topic model whether it is LDA- or HDP-based. Since every run results in a point estimate generated by a stochastic process, it makes sense to obtain several independent estimates and produce a more robust estimate for the model distributions.

For all corpora, there is a significant increase in retrieval performance if the index combines the estimates from several Markov chains rather than a single chain. All models were trained with $\beta = 0.006$, a value that consistently lead to good results on all corpora, and 700 topics. The smoothed language model was not used for this test, which means that the weighting parameter in Equation 4.1 was set to $\lambda = 0$.

Thus, in general, the more Markov chains are evaluated, the closer the estimate approaches the correct distribution, which improves the retrieval performance. This comes at the cost of computational effort. In our experiments, 10 Markov chains was always a reasonable number that also agrees with related research (Wei and Croft, 2006).

Corpus	1 chain	3 chains	10 chains	p-value 1 - 10
CISI	14.47%	16.93%	18.25%	<0.01
CRANFIELD	34.84%	37.97%	41.19	<0.01
MEDLINE	51.02%	51.20%	54.97	0.021
TIME MAGAZINE	48.10%	49.75%	53.99%	<0.01

Table 5.6: Effects of combining several Markov chains. The metric is mean average precision on a pure LDA model. The p-value is the comparison between results from one and ten chains.

5.2.6 Influence of the Number of Topics for LDA

As Griffiths and Steyvers (2004) show, the number of mixture components or topics has siginificant influence on the perplexity of a document model. It is therefore natural to assume that the same result holds for the task of information retrieval. Too small a number of topics would result in a few very general topics and the model became a smoothed language model, whereas too large a number of topics leads to a mixture of unigrams model that does not share topics among documents. Thus the assumption is that there is a specific number of topics that maximizes the retrieval performance.

In the experiments, this assumption could not be rejected. Related research (Azzopardi *et al.*, 2003), however, shows that the best document model for a corpus is not necessarily the best information retrieval model. This can be seen directly by comparing the plot in Figure 5.1 with Figure 5.3. The document model for the CRANFIELD data set maximizes its likelihood at 40 topics, whereas the information retrieval model performs best at 1050 topics.

Corpus	number of topics	mean average precision
CISI	700	17.48%
CRANFIELD	1050	42.43%
MEDLINE	500	56.32%
TIME MAGAZINE	900	58.85%

Table 5.7: Optimal number of topics per corpus and achieved mean average precision. The smoothing parameter for all corpora is $\beta = 0.007$

Table 5.7 shows the optimal number of topics for the four benchmarck corpora according to the best retrieval result obtained with this setting. This number was determined by a greedy heuristic on a restricted domain (300 to 1000 topics) submitted to a high performance cluster, which allowed to estimate several hundreds of parameter combinations at once. All models were trained with 10 independent Gibbs sampling estimates.

Figure 5.3 shows the typical behavior of the mean average precision as a function of the number of topics. The figure is based on the CRANFIELD dataset with $\beta = 0.007$, 10 Markov chains and $\lambda = 0$. It cannot be rejected from the plot that the function is concave. In all experiments, hillclimbing over the number of topics gave good results in retrieval performance.

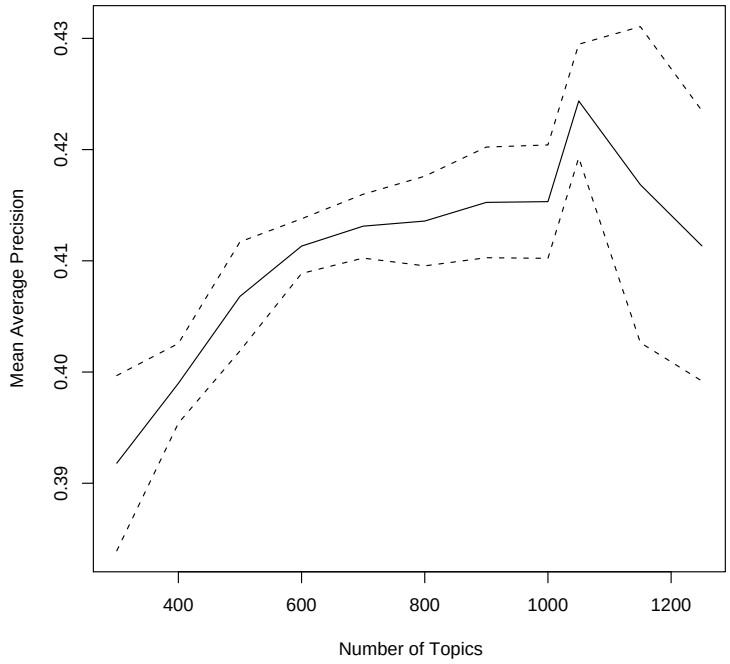

Figure 5.3: The mean average precision versus the number of topics on the CRANFIELD dataset based on 10 samples for each design point. Other parameters were $\beta = 0.007$, $\lambda = 0$, and 10 independent Gibbs sampling runs. The concavity of the function cannot be rejected from this plot. The optimal number of topics for the CRANFIELD data set is determined as 1050.

5.2.7 Influence of the Weighting between the Topic Model and the Language Model

Figure 5.4 shows how the weighting between the probabilistic topic model and the smoothed language model influences retrieval performance. Recall is the proportion of relevant documents that were returned by the application. Precision is the ratio of the number of relevant documents and the total number of returned documents. Ideally, a precision-recall plot starts almost horizontally at recall zero and precision close to one and stays that way until it drops to precision zero at recall one. In practice, this is rarely the case considering the

random nature of documents and queries.

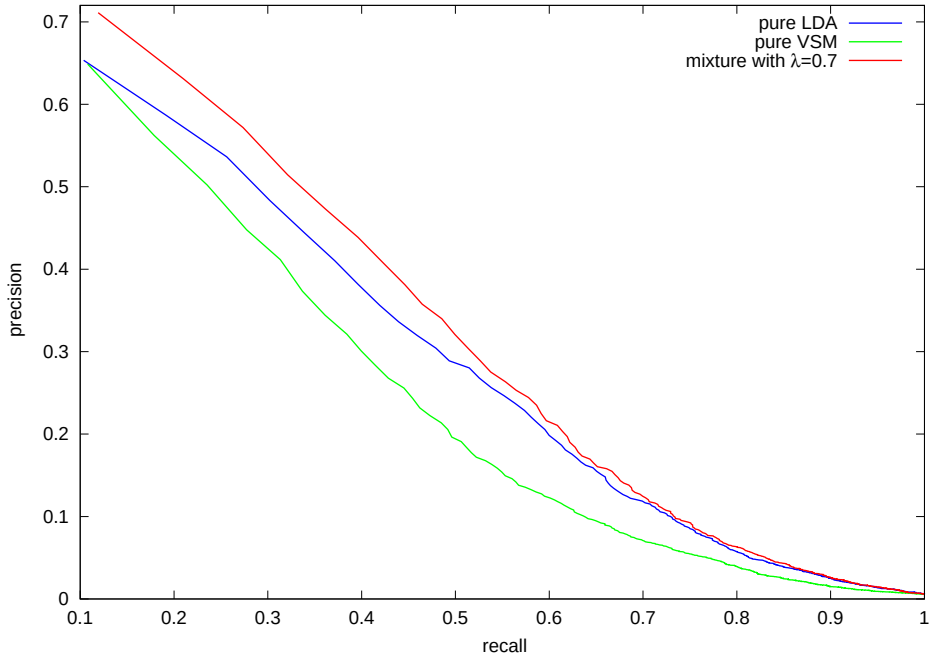

Figure 5.4: Precision and recall for different values of λ on the CRANFIELD dataset.

Clearly, the choice of λ influences the retrieval performance. Values between $\lambda = 0.7$ and $\lambda = 0.8$ worked best for all corpora in all experiments. The plot in Figure 5.4 also shows how the keyword-based search is augmented by the probabilistic topic model: both of the search methods perform worse than their combination.

5.2.8 Difference between LDA and HDP

Table 5.8 shows results of a direct comparison of an LDA- and an HDP-based model with weigthing parameter $\lambda = 0$. For both models, a symmetric Dirichlet distribution with smoothing constant $\beta = 0.007$ was used for the prior base distribution. The preprocessing for both models and all corpora were the same. The computation of the p-value in Table 5.8 was done by a Wilcoxon signed rank test based on paired observations for each query. The second and third column show the respective mean average precision.

The results show significant improvement for HDP over LDA on the CISI and the MEDLINE dataset. For CRANFIELD and TIME MAGAZINE, there is not enough evidence to favor the alternative hypothesis that HDP performs better than LDA. If all queries are compared pairwise ignoring the corpus factor, the resulting p-value is 0.076, which leads

to rejection of the null hypothesis at the 10 percent level of significance, but not at the five percent level.

Corpus	pure LDA	pure HDP	p-value
CISI	18.25%	19.62%	0.025
CRANFIELD	41.19%	41.47%	0.307
MEDLINE	54.97%	59.10%	0.016
TIME MAGAZINE	53.99%	53.20%	0.591

Table 5.8: Comparison of a pure LDA-based model versus a pure HDP-based model with smoothing constant $\beta = 0.007$. P-value is based on a Wilcoxon signed rank test with alternative hypothesis that LDA performs worse than HDP.

5.2.9 Improvements over the Baseline

Finally, it is interesting to determine if the prototype leads to improvements over the baseline. The results are presented in Table 5.9. For all corpora, improvements in information retrieval were achieved based on mean average precision. These improvements range between three percent for CISI up to more than 10 percent for MEDLINE.

For the baseline values, it is not possible to state the statistical significance based on per query comparison, because this is not available in the literature. The only statistical tests that are valid in this case are a sign test and a Wilcoxon signed rank test based on the aggregated values. Both tests result in a p-value of $\frac{1}{16} = 0.0625$. Here, the null hypothesis is that a probabilistic topic model does not lead to significantly different retrieval performance, while the alternative hypothesis is that a probabilistic topic improves retrieval performance. With a p-value of 0.06, the null hypothesis can be rejected in favor of the alternative hypothesis at the 10 percent level of significance, but not at the five percent level.

Corpus	Baseline	LDA-based	HDP-based
CISI	20.0%	23.28%	23.10%
CRANFIELD	39.2%	44.55%	45.41%
MEDLINE	51.8%	61.57%	62.34%
TIME MAGAZINE	52.6%	58.76%	55.88%

Table 5.9: Improvements over the baseline for all corpora. Shown is the result of the respective best parameter configuration.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 6:
Conclusions and Recommendations

This chapter discusses the experiment results from Chapter 5 and possible implications for the SHARE corpus. Further, it will give recommendations for future research on information retrieval based on probabilistic topic models.

6.1 Discussion of Experimental Results

Chapter 5 shows that probabilistic topic models can lead to significant improvements in information retrieval. A probabilistic topic model, however, cannot be used as a single model for information retrieval purposes successfully. In general, such a model will have a feature space that is too coarse to effectively discriminate documents given a user query. The smoothed language model, which is entirely keyword-based, can be trained independently of the probabilistic topic model.

It is desirable to have an efficient implementation for topic detection based on HDP, ideally as part of the MALLET (McCallum, 2002) package. The current detour of using Octave, Matlab, and Java to produce a trained document model for information retrieval is not efficiently usable for a standalone prototype.

6.1.1 Evaluation for SHARE

All experiments were done on abstracts from special domains. These are comparable with descriptions in SHARE's card library, which will be available for search to the user. These experiments, however, cannot replace experiments on the target corpus.

Once a sufficient number of documents from the SHARE corpus is available, it is therefore recommended to produce benchmark queries to adjust parameter settings. In addition, a final implementation of the search engine should collect user feedback and submitted queries to grow the number of benchmark queries. This allows the adjustment of model parameters during the whole life cycle of the application.

6.1.2 Preprocessing

Stemming leads to significant improvement in retrieval performance and storage requirements. Therefore, it is recommended that stemming be applied to the documents in SHARE as well.

Removing rare types has been shown as possibly disadvantegeous for information retrieval. Although the statistical significance has been shown for three of four corpora, the practical significance needs to be assessed separately. Removing rare types has, however, the advantage of reducing storage and computation time, and should therefore be considered.

6.1.3 Parameter Settings

For the prior parameter setting, $\beta = 0.007$ is recommended. This value performed well for all sample corpora. The burn-in time for LDA should be 1,000 iterations or more; a lag of 100 iterations between samples from each chain and at least 10 samples are necessary for a robust point estimate from a single chain. Ten independent Gibbs sampler runs result in consistently good results in information retrieval. The number of topics is hard to predict for an unseen corpus. A number between 500 and 900 can be expected to perform well, but user feedback should be used to adjust that. In general, the number of topics can be expected to become larger as the number of word tokens in the corpus grows. Typically that means that the number of documents has to grow in this case as well. There is, however, no way of estimating this number in advance or as a function of the corpus size. For the HDP-based model, the number of topics does not need to be specified as it is computed by the inference algorithm.

For the keyword-based model, a smoothing constant of $\mu = 1100$ showed best results on all corpora except the TIME MAGAZINE corpus. For TIME MAGAZINE, $\mu = 1000$ performed slightly better.

In all cases, the weighting of $\lambda = 0.7$ between the keyword-based and the topic-based model performed best. This result holds regardless if the topic model is LDA- or HDP-based.

6.2 Hidden Markov Models for Topic Detection

So far, all research presented in this thesis had the underlying assumption that observations are exchangeable (see Section 2.0.1). Though the evaluated models performed significantly better in the information retrieval task compared to the baseline, the question remains if further improvement is possible by also acknowledging the word order, which is not random in reality.

A refinement of LDA in that direction was presented by Griffiths *et al.* (2005). In this research, the authors defined a model in which each word is tagged not only by a topic index, but also by a syntactic class index (*e.g.,* noun, verb, adjective). The document generating process therefore gets an additional step: after determining the topic for word i, its syntactic state

is defined and then the word is drawn from a multinomial distribution that is conditioned not only on the topic, but also on the syntactic state. The sequence of these syntax labels is determined by a Hidden Markov Model (HMM), defining an order on the words in a document. The authors show that document models based on that model result in higher likelihood for a corpus than plain LDA. The question is now whether a document model that incorporates syntactic structure can improve information retrieval performance.

Hidden Topic Markov Models (HTMM), introduced by Gruber *et al.* (2007), represent another attempt to relax LDA's modeling assumptions. Instead of seeing a document as a bag of words, the authors treat it as a bag of sentences. Topic changes are allowed only at the beginning of a sentence and all words inside the sentence share the same topic. Assuming that words inside the same sentence are generated by the same topic is very intuitive. The experimental results presented in Gruber *et al.* (2007) show a higher corpus likelihood than LDA.

Another possible direction for improving topic models based on HMM is derived from HDP. Using HDP, it is possible to define an HMM in a nonparametric way. That is, the state space grows with the number of observations as the number of mixture components does in the HDP model presented in Section 2.2. A topic model based on such an HMM would combine the advantages of a nonparametric model and a model that does not rely on a bag of words assumption. The HDP-HMM is derived in Teh *et al.* (2006) and an inference algorithm is described.

All three presented refinements should be considered in future research, because they can possibly lead to document models that improve information retrieval performance. A query, however, needs its own model in a word-order-based information retrieval system. This follows because a user, who submits a query to the application, is typically not concerned about the ordering of the keywords, nor does he provide a complete sentence from which syntactic states can be inferred.

6.3 Empirical Priors for HDP

So far, all refinements of the document models considered changes of the model. As noted in Section 2.2, HDP can have any probability measure as the base measure. Rather than using a symmetric Dirichlet distribution as base measure, the distribution should be learned from the data directly.

As a step in this direction, a simple experiment was conducted using an asymmetric Dirichlet distribtuion as prior. For the distribution parameter, the smoothed language model

was used (see Chapter 4). The experimental results showed improvement in retrieval performance over the symmetric distribution. These results are not presented here, because an asymmetric base measure does not necessarily lead to an exchangeable process (Hansen and Pitman, 1998) and the inference scheme in Section 2.2.2 is not guaranteed to converge to the right distribution.

McAuliffe *et al.* (2006) present an algorithm to compute an empirical base measure using Gibbs sampling scheme kernel methods. In their example, the authors use a kernel based on the normal distribution. For document modeling, the kernel should be based on the Dirichlet distribution or a Polya Tree (Ghosh and Ramamoorthi, 2003). Kernels based on the Dirichlet distribution have been used by Hinneburg *et al.* (2007) and Draeger *et al.* (2009).

APPENDIX A:
Inference and Learning Algorithms

A.1 Inference for LDA

A.1.1 Expectation Propagation algorithm in R

```
exp_propagation_LDA
   <- function(document, alpha, pwa, numIter=100){

   # Initialize variables
   W <- length(document)
   T <- length(alpha)
   gamma_ <- alpha
   beta_ <- matrix(rep(0,W*T),nrow=W)
   beta_new <- beta_
   s <- rep(1,W)
   s_new <- s

   # Repeat until convergence
   for (i in 1:numIter){
      # Loop through all words
      for (w in 1:W){

      # Start with deletion
         gamma_w <- gamma_ - beta_[w,]
         # if one of the gamma_w's is negative, skip this word
         if (sum(gamma_w<0)==0 & document[w]>0){
            #
            # Moment matching
            dp_pwa_gamma <- pwa[w,]%*%gamma_w
            sum_gamma <- sum(gamma_w)
```

```
            zw <- dp_pwa_gamma/sum_gamma
            prefactor <- 1/zw * gamma_w/sum_gamma
            m  <- prefactor *(pwa[w,]+ dp_pwa_gamma)
                            /(1+sum_gamma)
            m2 <- prefactor *(gamma_w +1)/(1+ sum_gamma)
                            *(2*pwa[w,]+ dp_pwa_gamma)
                            /(2+sum_gamma)
            gamma_prime <- (m-m2)/(m2-m^2)*m
            #
            # update
            # Define the step size
            mu <- 1/document[w]
            # update variables tentatively
            beta_new[w,]<-mu*(gamma_prime - gamma_w)
                            +(1-mu)*beta_[w,]
            s_new[w]<-zw*gamma(sum(gamma_prime))
                      /prod(gamma(gamma_prime))
                      *prod(gamma(gamma_w))
                      /gamma(sum_gamma)
            #
            # inclusion
            gamma_new <- gamma_+document[w]
                         *(beta_new[w,]-beta_[w,])
            if (sum(gamma_new<0)==0){
               gamma_<- gamma_new
               beta_[w,] <- beta_new[w,]
               s[w]<-s_new[w]
            }
         }
      }
   }
   return (list(gamma=gamma_,beta=beta_,s=s))
}
```

A.1.2 Gibbs Sampler for LDA in R

```
function(WS,DS,T,NN,ALPHA,BETA,Z=NA){
   #
   # WS is a vector of observations
   # DS has the same length as WS and specifies the document,
   # observation i came from
   # NN is the number of iterations
   # ALPHA is the prior parameter for the topic distribution
   # BETA is the prior parameter for the word distribution
   # given a topic
   #
   # e1071 provides useful function for discrete distributions
   require (e1071)
   #
   # Create the return values
   #
   WP<-matrix(0,nrow=max(WS),ncol=T)
   DP<-matrix(0,nrow=max(DS),ncol=T)
   ztot<-matrix(0,nrow=T,ncol=1)
   #
   # Create local and temp variables
   #
   topic<-0
   wbeta <- max(WS)*BETA
   #
   # Initialize the states
   #
   if (is.na(Z)){
      Z<-matrix(0,nrow=length(WS),ncol=1)
      for(i in 1:length(WS)){
         wi <-WS[i]
         di <-DS[i]
         topic<-rdiscrete(1,rep(1/T,each=T),1:T)
```

```
      Z[i]<-topic
      WP[wi,topic]<-WP[wi,topic]+1
      DP[di,topic]<-DP[di,topic]+1
      ztot[topic]<-ztot[topic]+1
   }
}
else { # Start from previously saved state
   for (i in 1:length(WS)){
      wi <-WS[i]
      di <-DS[i]
      topic<-Z[i]
      WP[wi,topic]<-WP[wi,topic]+1
      DP[di,topic]<-DP[di,topic]+1
      ztot[topic]<-ztot[topic]+1
   }
}
#
# Finally, start sampling
#
for (iter in 1:NN){
   # permutate the order of observations
   order<-sample(1:length(WS))
   for (ii in 1:length(WS)){
      i<-order[ii]
      wi <-WS[i]
      di <-DS[i]
      topic <-Z[i]
      ztot[topic]<-ztot[topic]-1
      WP[wi,topic]<-WP[wi,topic]-1
      DP[di,topic]<-DP[di,topic]-1
      # Compute probabilities p(w_i|z)
      probs<-(WP[wi,]+BETA)/(ztot+wbeta)*(DP[di,]+ALPHA)
      # Sample from this discrete distribution
      topic <- rdiscrete(1,probs,1:T)
```

```
            #Update the topic counts
            Z[i]<-topic
            WP[wi,topic]<-WP[wi,topic]+1
            DP[di,topic]<-DP[di,topic]+1
            ztot[topic]<-ztot[topic]+1
        }
    }
    return (list(WP=WP,DP=DP,Z=Z,ZTOT=ztot))
}
```

A.2 Inference and learning for HDP

A.2.1 Gibbs Sampling for HDP

```
function (hdp, alpha0, gamma, lambda, numiter, vocab)
{
   # parameters
   # hdp is a list of numeric vector with word indices
   # alpha0 is the concentration parameter for each DP
   # gamma is the concentration parameter for the HDP
   # numiter is the number of iterations
   # vocab is the vocabulary

   # create some local and intermediate variables
   V<-length(vocab)

   #use internal representation as Griffiths & Steyvers (2006)
   WS<-numeric()
   DS<-numeric()
   for (i in 1:length(hdp)){
      WS <- c(WS, hdp[[i]])
      DS<-c(DS,rep(i,length(hdp[[i]])))
   }
   K<-1 # number of assigned clusters
   beta <- rdirichlet(1,c(length(hdp),gamma))

   # create the output matrices (these have to grow ...)
   # wt vocab x topic, dt document x topic
   wt <- matrix(0, V, 1) # one class at first
   dt <- matrix(0, length(hdp),1)

   # Initialize the relevant vectors
   zvec<-rep (1,length(WS)) # First try, assign all to one
   # mvec holds the number of tables for each
```

```
# restaurant serving dish k
mvec <- matrix(rep(0,length(hdp)),length(hdp),K)

for (l in 1:numiter){
   if (l %% 10 ==0)
      print (paste("iteration",l, sep=": "))
   # slot is a temp variable that iterates
   # over all observations
   slot <- 1
   # now the Gibbs updates
   for (j in 1:length(hdp)){
      for (i in 1:length(hdp[[j]])){
         pzji <- numeric(K+1)
         # Take out the current observation
         wt[WS[slot],zvec[slot]]
                  <- max(0,wt[WS[slot], zvec[slot]] -1)
         dt[j,zvec[slot]] <- max(0, dt[j,zvec[slot]] -1)

         # always sample for one more topic
         for (k in 1:(K+1)) {
            if (k < K+1){
               # number of customers in restaurant j
               # having dish k
               njk <- dt[j,k]

               pzji[k] <- (njk+alpha0*beta[k])
                          *(wt[WS[slot],k]+lambda)
                          /(sum(wt[,k])+V*lambda)
            }
            else {
               # if k=k_new
               pzji[k] <- alpha0*beta[k]/V
            }
         }
```

```
zvec[slot]<-sample(x=1:(K+1),size=1, prob=pzji)
# here we have to increase the number of classes
if (zvec[slot]>K){
   K <- K+1
   wt<-matrix(wt,V,K)
   wt[,K]<-rep(0,V)
   dt<-matrix(dt,length(hdp),K)
   dt[,K]<-rep(0, length(hdp))
   mvec <- matrix(mvec,length(hdp), K)
   mvec[,K] <- rep(0,length(hdp))
}

# Now, put everything back in order
wt[WS[slot], zvec[slot]]
           <- wt[WS[slot], zvec[slot]] +1
dt[DS[slot], zvec[slot]]
           <- dt[DS[slot], zvec[slot]] +1

# sample the number of tables
for (k in 1:K)
   mvec[j,k]<-sample_tables
              (alpha0, beta[k],dt[j,k])
# sample beta
beta<- rdirichlet(1,c(apply(mvec,2,sum),gamma))

# Delete empty classes
wtGzero <- apply (wt, 2, sum)>0
wt<-matrix(wt[,wtGzero],V)
dt<-matrix(dt[,wtGzero], length(hdp))
mvec<-matrix(mvec[,wtGzero], length(hdp))
K<-sum(wtGzero)
# fix the z vector
for (i in 1:length(wtGzero)){
   # this was an empty class before
```

```
            if (!wtGzero[i])
               zvec[zvec>i]<-zvec[zvec>i]-1
         }
         slot <- slot+1
      }
   }
   dimnames(wt)[1]<-list(words=vocab)
   return (list(WT=wt, DT=dt, Z=zvec))
}
```

A.2.2 Auxiliary Variable Sampling

```
function (prior, numTables, numDraws, numIter) {
   # a and b are the parameters for the gamma prior
   a<- prior[1]
   b<- prior[2]

   # J is the number of restaurants
   J <- length(numDraws)
   # Initialize auxiliary variables
   s <- sample(c(0,1),size=J, replace=TRUE)
   w <- runif(J)
   # Declare the concentration parameter
   alpha0 <- 0

   for (i in 1:numIter){
      #First, sample alpha0
      alpha0 <- rgamma(1,a+numTables-sum(s),
                       scale=b-sum(log(w)))
      # then resample s
      p <- numDraws/alpha0/(numDraws/alpha0 +1)
      s <- apply(as.matrix(p),1, function(x) rbinom(1,1,x))
      # Finally, resample w
```

```
      w <- apply (as.matrix (numDraws), 1,
                  function(x) rbeta(1, alpha0+1, x))
    }

    return (list(alpha0=alpha0, s=s, w=w))
}
```

APPENDIX B:
Implementation Examples

B.1 A MEX Function to Compute the PMF for the Number of Mixture Components in a CRP

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "mex.h"

void mixpmf(double *p, int n, double g){
  int i,j;
  double *pp, *tmp;
  pp=mxCalloc(n,sizeof(double));

  p[0]=1;
  for (i=2; i<n; i++){
    pp[0]=(i-1)*p[0]/(i-1+g);
    for (j=1; j<i; j++) pp[j]=((i-1)*p[j]+g*p[j-1])/(i-1+g);
    tmp=p;
    p=pp;
    pp=tmp;
  }
}

void mexFunction(int nlhs, mxArray *plhs[],
                 int nrhs, const mxArray *prhs[]){
  double g, *p;
  int n;

  n=(int)(mxGetScalar(prhs[0]));
```

```
  g=(double)(mxGetScalar(prhs[1]));

  plhs[0]=mxCreateDoubleMatrix(1,n,mxREAL);
  p=mxGetPr(plhs[0]);
  mixpmf(p,n,g);
}
```

B.2 Matlab Code to Train HDP Model

```
% create the prior distribution parameters
pML = zeros(vocabSize,1);
numTokens=0;
numDoc =size (restaurants,2);

for (i=1:numDoc)
    numTokens=numTokens+size(restaurants{i},2);
end
for (i=1:numDoc)
    for (j=1:size(restaurants{i},2))
        pML(restaurants{i}(j)) = pML(restaurants{i}(j)) +1;
    end
end

hh=pML/numTokens*200;

% create the matrix for the sum of the models
pTotal = zeros(vocabSize,size(restaurants,2));

% create the prior parameters on the hyperparameters
% (\gamma~Gamma(1,0.1)
% and \alpha_0~Gamma(1,1))
alphaa=[ 1 1];
```

```
alphab=[ .1 1];

% train the different Markov chains

for (c=1:10)
    % train the model
    [hdp, sample, lik, predlik] = hdp2Multinomial_run(
                          hh,alphaa, alphab,200,
                          restaurants, restaurants,1000,
                          20, 100, 1, 1, 15, 1, 1);
    % obtain the topics as smoothed
    % distributions over words
    p_w_z=hdp.base.classqq(:,1:hdp.base.numclass);
    for (i = 1:hdp.base.numclass)
        p_w_z(:,i)=(p_w_z(:,i)+hh)
                  ./(sum(p_w_z(:,i))+sum(hh));
    end
    % obtain the documents as distributions over topics
    p_z_d=zeros(hdp.base.numclass,size(restaurants,2));
    for (i = 2:(size(restaurants,2)+1))
        p_z_d(:,i-1)=hdp.dp{i}.classnd(1:hdp.base.numclass);
    end
    % normalize column wise
    p_z_d=p_z_d*diag(1./sum(p_z_d,1));
    % generate the final model
    p_w_d=p_w_z*p_z_d;
    fprintf('RUN %d finished\n',c)
    pTotal = pTotal + p_w_d;
end

% compute the average
p_w_d = pTotal/10;
```

B.3 Java Methods

B.3.1 Average Precision Computation

```
/**
* @return the average precision for the
*         retrieval task based on the
*         benchmark corpus.
*/
public double[] computeAveragePrecision() {
   int numQ = queries.size();
   double[] aP = new double[numQ];
   for (int i = 0; i < numQ; i++) {
      double pN = 0;
      // each Query object "knows" its
      // relevant documents
      ArrayList<Integer> relDocs =
         queries.get(i).getRelevantDocs();
      int numRel = relDocs.size();
      int numFound = 0;
      // qres is an ordered list of documents
      SearchResults qRes = results.get(i);
      int lastRelRank = 1;
      int r = 1;
      double sumPn = 0;
      for (DocScore d : qRes) {
         if (relDocs.contains(d.id())) {
            numFound++;
            pN = (pN * lastRelRank + 1) / r;
            sumPn += pN;
            lastRelRank = r;
         }
         // if all relevant documents are found, stop
         if (numFound == numRel)
```

```
            break;
         n++;
      }
      aP[i] = sumPn / numRel;
   }
   // return the mean average precision
   return aP;
}
```

B.3.2 Computation of Mean Average Precision

```
public double computeMeanAveragePrecision() {
   double[] aP = computeAveragePrecision();
   double mAP = 0;
   for (int i = 0; i < aP.length; i++)
      mAP += aP[i];
   return mAP / aP.length;
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

Antoniak, C. E. 1974. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The Annals of Statistics*, **2**, 1152–1174.

Azzopardi, L., Girolami, M. and van Risjbergen, K. Investigating the relationship between language model perplexity and IR precision-recall measures. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 369–370, New York, NY, USA, 2003. ACM.

Billingsley, P. *Probability and Measure*. Wiley Series in probability and mathematical statistics. John Wiley & Sons Ltd., 2^{nd} edition, 1986.

Blei, D. M. and Jordan, M. I. 2006. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, **1**, 121–144.

Blei, D. M., Ng, A. and Jordan, M. I. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, **3**, 993–1022.

Brooks, S. P. and Roberts, G. O. 1998. Diagnosing convergence of markov chain monte carlo algorithms. *Statistics and Computing*, **8**.

Carlin, B. P. and Louis, T. A. *Bayes and Empirical Bayes Methods for Data Analysis*. Chapman & Hall, 1996.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. and Harshman, R. September 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, **41**, 391–407.

Draeger, M., Squire, K. and Martell, C. Density Estimation for Multinomial Distributions using Dirichlet Distribution Kernels. unpublished, February 2009.

Ferguson, T. 1973. A bayesian analysis of some nonparametric problems. *The Annals of Statistics*, **1**, 209–230.

Gamerman, D. *Markov Chain Monte Carlo - Stochastic Simulation for Bayesian Inference*. Chapman & Hall, 1997.

Geman, S. and Geman, D. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. In *Readings in uncertain reasoning*, pages 452–472. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

Ghosh, J. K. and Ramamoorthi, R. V. *Bayesian Nonparametrics*. Springer Series in Statistics. Springer-Verlag, 2003.

Griffiths, T. L. and Steyvers, M. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, **101**, 5228–5235.

Griffiths, T. L. and Steyvers, M. Probabilistic topic models. In Landauer, T., McNamara, D., Dennis, S. and Kintsch, W., editors, *Latent Semantic Analysis: A Road to Meaning*. Laurence Erlbaum, 2006.

Griffiths, T. L., Steyvers, M., Blei, D. M. and Tenenbaum, J. B. Integrating topics and syntax. In *In Advances in Neural Information Processing Systems 17*, pages 537–544. MIT Press, 2005.

Gruber, A., Rosen-Zvi, M. and Weiss, Y. March 2007. Hidden Topic Markov Models. *Artificial Intelligence and Statistics (AISTATS)*.

Hansen, B. and Pitman, J. Prediction Rules for Exchangeable Sequences Related to Species Sampling. In *in Processor Design. Master's Thesis. LM Ericsson 2000*, 1998.

Hawkins, B. M. Developing a modular framework for implementing a semantic search engine. Master's thesis, Naval Postgraduate School, Monterey, CA, 2009.

Hinneburg, A., Gabriel, H.-H. and Gohr, A. Bayesian folding-in with Dirichlet kernels for PLSI. In *Seventh IEEE International Conference on Data Mining*, pages 499–504. IEEE Computer Society, 2007.

Hofmann, T. Probabilistic latent semantic indexing. In *Proceedings of the 22^{nd} Annual International SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, 1999.

Hoschek, W. *Colt*. CERN, http://dsd.lbl.gov/~hoschek/colt/, 2004.

Johnson, J. and Blais, C. SHARE Repository Framework: Component Specification and Ontology. In *Proceedings of the 5^{th} annual acquisition research symposium*, pages 194–212, 2008.

Jurafsky, D. and Martin, J. H. *Speech and Language Processing : An Introduction to Natural Language Processing*. Pearson Prentice Hall, Upper Saddle River; New Jersey 07458, 2^{nd} edition, 2008.

Martell, C., Adams, P. H., Anand, P., Gera, R., Grant, G., Draeger, M. and Squire, K. Research: A requirements search engine. In *Proceedings of the 5^{th} annual acquisition research symposium*, pages 213–229, May 2008.

McAuliffe, J., Blei, D. M. and Jordan, M. I. March 2006. Nonparametric empirical Bayes for the Dirichlet process mixture model. *Statistics and Computing*, **16**, 5–14.

McCallum, A. K. *MALLET: A Machine Learning for Language Toolkit*, 2002. http://mallet.cs.umass.edu.

McLachlan, G. J. and Basford, K. E. *Mixture Models - Inference and Applications to Clustering*, volume 84 of *STATISTICS: Textbooks and Monographs*. Marcel Dekker, Inc., 270 Madison Avenue, New York, New York 10016, 1987.

Minka, T. and Lafferty, J. Expectation-propagation for the generative aspect model. In *In Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 352–359. Morgan Kaufmann Publishers Inc., 2002.

Moschopoulos, P. G. 1985. The distribution of the sum of independent gamma random variables. *Annals of the Institute of Statistical Mathematics*, **37**, 541–544.

Nigam, K., McCallum, A. K., Thrun, S. and Mitchell, T. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, **39**, 103–134.

Pitman, J. *Combinatorial stochastic processes*, volume 1875 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 2006.

Porter, M. F. 1980. An algorithm for suffix stripping. *Program*, **14**, 130–137.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008.

Raftery, A. E. and Lewis, S. How many iterations for the Gibbs sampler. In *In Bayesian Statistics 4*, pages 763–773. Oxford University Press, 1992.

Robertson, S. A new interpretation of average precision. In *SIGIR '08: SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 689–690, New York, NY, USA, 2008. ACM.

Roussinov, D. and Fan, W. Learning ranking vs. modeling relevance. In *Proceedings of the 39^{th} Hawaii International Conference on System Sciences*. IEEE, 2006.

Teh, Y. W., Kurihara, K. and Welling, M. Collapsed variational inference for HDP. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

Teh, Y. W., Jordan, M. I., Beal, M. J. and Blei, D. M. December 2006. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, **101**, 1566–1581.

Titterington, D. M., Smith, A. F. M. and Makov, U. E. *Statistical Analysis of Finite Mixture Models*. Wiley Series in probability and mathematical statistics. John Wiley & Sons Ltd., 1985.

Wei, X. and Croft, W. B. 2006. LDA-based document models for ad-hoc retrieval. *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185.

Zhai, C. and Lafferty, J. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, **22**, 179–214.

Referenced Authors

Adams, Paige H.
Anand, Pranav
Antoniak, Charles E.
Azzopardi, Leif

Basford, Kaye E.
Beal, Matthew J.
Billingsley, Patrick
Blais, Curtis
Blei, David M.
Brooks, Stephen P.

Carlin, Bradley P.
Croft, W. B.

Deerwester, Scott
Draeger, Marco
Dumais, Susan T.

Fan, Weiguo
Ferguson, Thomas
Furnas, George W.

Gabriel, Hans-Henning
Gamerman, Dani
Geman, Donald
Geman, Stuart
Gera, Ralucca
Ghosh, J. K.
Girolami, Mark
Gohr, Andrè
Grant, Gehrke
Griffiths, Thomas L.
Gruber, Amit

Hansen, Ben
Harshman, Richard
Hawkins, Brian M.
Hinneburg, Alexander
Hofmann, Thomas
Hoschek, W.

Johnson, Jean
Jordan, Michael I.
Jurafsky, Daniel

Kurihara, K.

Lafferty, John
Landauer, Thomas K.
Lewis, Steven
Louis, Thomas A.

Makov, U. E.
Martell, Craig
Martin, James H.
McAuliffe, Jon
McCallum, Andrew Kachites
McLachlan, Geoffrey J.
Minka, Thomas
Mitchell, Tom
Moschopoulos, P. G.
Ng, Andrew
Nigam, Kamal

Pitman, Jim
Porter, Martin F.

R Development Core Team
Raftery, Adrian E.
Ramamoorthi, R. V.
Roberts, Gareth O.
Robertson, Stephen
Rosen-Zvi, Michal
Roussinov, Dmitri

Smith, A. F. M.
Squire, Kevin
Steyvers, Mark

Teh, Y. W.
Teh, Yee Whye
Tenenbaum, Joshua B.
Thrun, Sebastian
Titterington, D. M.

van Risjbergen, Keith

Wei, X.
Weiss, Yair
Welling, M.

Zhai, Chengxiang

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudly Knox Library
   Naval Postgraduate School
   Monterey, California

3. Department of Operations Research
   Naval Postgraduate School
   Monterey, California

4. Department of Computer Science
   Naval Postgraduate School
   Monterey, California

5. Marco Draeger
   Ransbach-Baumbach, Germany