Online Multi-label Text Classification using Topic Models

A thesis submitted for the degree of

Doktor der Naturwissenschaften

at the Department of Physics, Mathematics and Computer Science
at the *Johannes Gutenberg-Universität*
in Mainz

Sophie Burkhardt
born in Filderstadt

Mainz, 4.7.2018

Abstract

Every day, an enormous amount of text data is produced. Sources of text data include news, social media, emails, text messages, medical reports, scientific publications and fiction. Due to these increasing amounts of data, the need for scalable and interpretable models, that help analyze this data, is growing. Such models may generally be divided into supervised and unsupervised models. In the case of supervised methods, the problem is to classify the data given an existing set of labels. In the case of unlabeled data, unsupervised models may be learned, that cluster the data to reveal hidden similarities and regularities.

For both of these tasks, existing methods often lack either scalability or interpretability. Scalability is especially important as dataset sizes grow and it is often necessary to be able to process streaming data online without being able to store it. Interpretability helps to understand a given result and is of increasing importance if actions have to be taken based on the modeling result. Such actions often need to be justified to customers or other stake-holders based on information extracted from the model. In this thesis, both scalability and interpretability are achieved by focusing on generative Bayesian topic models for text data. These models are applicable in the supervised as well as the unsupervised setting while maintaining interpretability in both cases.

Overall, four novel topic models are proposed in this thesis. These models allow to not only cluster and classify the data but also to assign a semantic interpretation to each cluster that helps to understand its content. This way, it is possible to understand *why* a text document was assigned a certain topic. At the same time, the proposed models are scalable to large datasets and able to handle streams of data.

The first model is trained online and used for multi-label classification of text, meaning that each document may be assigned several labels that possibly exhibit dependencies. The second model is a nonparametric multi-label topic model that utilizes a novel sampling method to make it more efficient. Its nonparametric nature allows it to model different label frequencies. The third model is also nonparametric and trained with a hybrid Variational-Gibbs sampling training algorithm that takes advantage of sparsity. The last model is trained online and tracks changes of topics over time to analyze the German media with respect to the refugee crisis. In conclusion, this thesis demonstrates the manifold possibilities and flexibility of the topic model framework for complex settings such as multi-label classification by exploring different learning and sampling strategies.

Zusammenfassung

Tagtäglich werden ernorme Mengen an Textdaten produziert. Mögliche Quellen von Textdaten sind z.B. Nachrichten, soziale Medien, Emails, Textnachrichten, medizinische Gutachten und Belletristik. Aufgrund der wachsenden Datenmengen, wächst auch der Bedarf an skalierbaren und interpretierbaren Modellen, mit denen diese Daten analysiert werden können. Solche Modelle können generell in überwachte und unüberwachte Verfahren eingeteilt werden. Im Fall von überwachten Verfahren ist das Problem, die Daten anhand einer existierenden Labelmenge zu klassifizieren. Im Fall von nicht klassifizierten Daten können unüberwachte Modelle gelernt werden, die die Daten gruppieren und dadurch versteckte Gemeinsamkeiten und Regularitäten aufdecken.

Methoden für beide Anwendungsfälle zeichnen sich oftmals durch fehlende Skalierbarkeit und Interpretierbarkeit aus. Skalierbarkeit is besonders wichtig aufgrund der wachsenden Datenmengen, die die Notwendigkeit implizieren, den Datenstrom online zu verarbeiten, wobei die Daten nicht gespeichert werden müssen. Interpretierbarkeit hilft dabei, das erzielte Resultat zu verstehen und ist von wachsender Bedeutung, wenn Handlungsentscheidungen aufgrund der erzielten Resultate getroffen werden sollen. Solche Handlungen müssen oft gegenüber Kunden oder anderen Beteiligten gerechtfertigt werden. Dies geschieht aufgrund von Informationen, die aus dem gelernten Modell extrahiert werden. In dieser Arbeit werden sowohl Skalierbarkeit als auch Interpretierbarkeit erreicht, indem generative Bayes'sche Modelle für Textdaten angewandt werden. Diese Modelle sind im überwachten und im unüberwachten Fall anwendbar, wobei in beiden Fällen die Interpretierbarkeit gewährleistet bleibt.

Insgesamt werden in dieser Arbeit vier neuartige Topic Models vorgeschlagen. Diese Modelle ermöglichen nicht nur die Daten zu gruppieren, sondern auch, den Gruppen eine Semantik zu verleihen, die dabei hilft, ihren Inhalt zu verstehen. Dadurch ist es möglich zu verstehen, *warum* ein Textdokument einer bestimmten Gruppe zugewiesen wurde. Gleichzeitig sind die vorgeschlagenen Modelle auf große Datensätze skalierbar und können mit Datenströmen umgehen.

Das erste Modell wird online trainiert und zur Multi-Label-Klassifikation verwendet, was bedeutet, dass jedes Dokument mehreren Klassen zugeordnet werden kann, die möglicherweise voneinander abhängig sind. Das zweite Modell ist ein nichtparametrisches Multi-Label-Topic-Model, das sich eine neuartige Sampling-Methode zunutze macht, um effizienter zu sein. Die Eigenschaft, nichtparametrisch zu sein, erlaubt das Modellieren verschiedener Labelfrequenzen. Das dritte Modell ist ebenfalls nichtparametrisch und wird mit einem hybriden Variational-Gibbs Trainingalgorithmus trainiert, der sich die Dünnbesetztheit von Matrizen zunutze macht. Das letzte Modell wird online trainiert und

verfolgt Änderungen über die Zeit hinweg, um deutsche Medien zum Thema Flüchtlingskrise zu analysieren. Insgesamt zeigt diese Arbeit die vielfältigen Möglichkeiten und die Flexibilität des Topic Model Frameworks für komplexe Anwendungsfälle wie Multi-Label-Klassifikation, indem verschiedene Lern- und Samplingverfahren erforscht werden.

Publications

Parts of this thesis have been published in international conferences and journals:

- Sophie Burkhardt and Stefan Kramer. "Multi-Label Classification using Stacked Hierarchical Dirichlet Processes with Reduced Sampling Complexity". In: *Knowledge and Information Systems* (2018), pp. 1–23. DOI: 10.1007/s10115-018-1204-z
- Sophie Burkhardt and Stefan Kramer. "Online Multi-label Dependency Topic Models for Text Classification". In: *Machine Learning* 107.5 (May 2018), pp. 859–886. DOI: 10.1007/s10994-017-5689-6
- Sophie Burkhardt and Stefan Kramer. "Online Sparse Collapsed Hybrid Variational-Gibbs Algorithm for Hierarchical Dirichlet Process Topic Models". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Michelangelo Ceci et al. Cham: Springer International Publishing, 2017, pp. 189–204. DOI: 10.1007/978-3-319-71246-8_12
- Zahra Ahmadi, Sophie Burkhardt, and Stefan Kramer. "Online Topic Modeling: Keeping Track of News Topics for Social Good". In: *Proceedings of the Second Workshop on Data Science for Social Good at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*. 2017. URL: http://ceur-ws.org/Vol-1960/paper5.pdf
- Sophie Burkhardt and Stefan Kramer. "Multi-Label Classification using Stacked Hierarchical Dirichlet Processes with Reduced Sampling Complexity". In: *ICBK 2017 - International Conference on Big Knowledge*. Hefei, China: IEEE, 2017, pp. 1–8. DOI: 10.1109/ICBK.2017.27
- Sophie Burkhardt and Stefan Kramer. "On the Spectrum Between Binary Relevance and Classifier Chains". In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. SAC '15. Salamanca, Spain: ACM, 2015, pp. 885–892. DOI: 10.1145/2695664.2695854

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Contribution . . . 2
- 1.3 Structure . . . 3

2 Background **7**
- 2.1 Topic Models . . . 7
  - 2.1.1 Statistical Background . . . 7
  - 2.1.2 Latent Dirichlet Allocation (LDA) . . . 10
  - 2.1.3 Gibbs Sampling for LDA . . . 11
  - 2.1.4 Variational Bayes for LDA . . . 14
  - 2.1.5 Batch Variational Bayes . . . 17
  - 2.1.6 Online Variational Bayes . . . 17
  - 2.1.7 Gibbs Sampling vs. Variational Bayes . . . 18
  - 2.1.8 Nonparametric Topic Models . . . 19
  - 2.1.9 Different Topic Models: Overview . . . 26
- 2.2 Multi-label Classification . . . 28
  - 2.2.1 Evaluation Measures . . . 29
  - 2.2.2 Threshold Selection Methods . . . 30
  - 2.2.3 Binary Relevance . . . 32
  - 2.2.4 Classifier Chains . . . 33
  - 2.2.5 Block Classifier Chains . . . 34
  - 2.2.6 Extreme Multi-Label Classifiers . . . 45
- 2.3 Multi-Label Topic Models . . . 45
  - 2.3.1 Labeled LDA . . . 45
  - 2.3.2 Dependency-LDA . . . 47

3 Online Multi-label Topic Model **51**
- 3.1 Online Topic Modeling for Multi-Label Classification . . . 51
- 3.2 Fast-Dep.-LLDA . . . 53
- 3.3 Greedy Layer-Wise Training . . . 55
- 3.4 Online Fast-Dep.-LLDA (SCVB-Dep.) . . . 58
- 3.5 Reversed Fast-Dep.-LLDA . . . 62

3.6 Prediction . . . 64
3.7 Computational Complexity . . . 66
3.8 Experiments . . . 67
3.8.1 Binary Predictions . . . 67
3.8.2 Datasets . . . 67
3.8.3 Experimental Setting . . . 68
3.8.4 Results . . . 70
3.9 Discussion . . . 77
3.10 Conclusion . . . 79

4 Nonparametric Multi-label Topic Model 81
4.1 Introduction . . . 81
4.2 Differences to Existing Methods . . . 83
4.3 Proposed Method . . . 83
4.3.1 Hierarchical Model . . . 83
4.3.2 Alias-sampling for the HDP . . . 86
4.3.3 Supervised Training . . . 90
4.3.4 Prediction . . . 90
4.4 Experiments . . . 91
4.4.1 Algorithms . . . 91
4.4.2 Datasets . . . 92
4.4.3 Parameter Settings . . . 92
4.4.4 Results . . . 95
4.5 Conclusion . . . 98

5 Online Nonparametric Topic Model 101
5.1 Introduction . . . 101
5.2 Proposed Method – Hybrid Variational-Gibbs . . . 103
5.3 Doubly Sparse Updates for Online HDP . . . 105
5.3.1 Further Improving the Sparse Sampler. . . . 105
5.4 Algorithm Description . . . 106
5.5 Datasets . . . 109
5.6 Evaluation . . . 109
5.7 Experiments . . . 110
5.7.1 Mean-field vs. Hybrid Approach. . . . 110
5.7.2 Gibbs Sampling vs. Hybrid Approach. . . . 111
5.7.3 Supervised Variant . . . 112
5.8 Conclusion . . . 116

6 Online Topic Model for Tracking News 119
6.1 Tracking News Related to the Refugee Crisis . . . 119
6.2 Existing Online Topic Models . . . 120
6.3 New Online Topic Model . . . 121
6.4 Experiments . . . 123

6.5 Discussion and Conclusion 123

7 Conclusion **127**
7.1 Online Multi-label Topic Model 127
7.2 Nonparametric Multi-label Topic Model 127
7.3 Hybrid Nonparametric Topic Model 128
7.4 Tracking News Topics 128
7.5 Overview of Key Results 128
7.6 Outlook 129

Chapter 1

Introduction

1.1 Motivation

Every day, an enormous amount of text data is produced. Sources of text data include news, social media, emails, text messages, medical reports, scientific publications and fiction, to name just a few. As dataset sizes grow, so does the need for efficient algorithms to extract information, categorize, and analyze them. Generally, we can distinguish two main tasks: 1. Texts can be assigned to a predefined system of labels, the supervised task, or 2. they can be clustered into groups of similar documents, the unsupervised task. In this work we are concerned with a group of models that is applicable to both tasks, called topic models.

The topic models discussed in this thesis are generative models of text. In contrast to discriminative models, generative models assume that the data was generated from certain priors that are explicitly modeled. The modeling of priors in addition to the data and latent variables enables an important property: Generative models provide an explanation for *how* the data was generated and thus an answer to the question *why* the latent variables are the way they are. This makes generative models highly interpretable and flexible. Interpretable, because we can analyze the generative parameters for the data that is modeled and flexible because the behavior of the model may be influenced by adjusting the priors.

Generative models are of growing importance as dataset sizes grow and methods are increasingly employed in critical application contexts where the interpretability of model outputs is crucial since decisions that are taken based on these outputs have to be justified. As a recent example, the 2018 reform of EU data protection rules now gives all EU citizens the right to have a say when decisions are automated. Article 22 of the general data protection regulation (GDPR) states: "The data subject shall have the right not to be subject to a decision based solely on automated processing...". Examples of such decisions could be giving a loan or a specific insurance policy where

companies are not allowed to blindly follow automated recommendations. In such cases, generative models could provide the means to understand model outputs and the ability to justify decisions correspondingly.

Data is also often diverse and best represented in a hierarchical structure with different levels of granularity. Bayesian models naturally lend themselves to such multi-level hierarchical analyses where data is stored in different levels of abstraction that influence each other using different kinds of priors and probability distributions. Thus, there is an increasing need for scalable Bayesian methods for the analysis of large datasets. This thesis has a focus on text data.

1.2 Contribution

The main focus of the contributions lies in the scalability of the presented models. This scalability is achieved by developing online methods that can be updated continuously and by developing efficient sampling methods that take advantage of sparsity. These two aspects make it possible to present relatively complex Bayesian models that nevertheless may be trained and tested efficiently using large datasets on standard desktop computers.

The individual contributions are given at the beginning of each chapter. In summary, the contributions and the papers that are the basis of the given chapter are as follows:

1. Chapter 3 introduces the first multi-label topic model that can be trained online and considers label dependencies at the same time.

   First published as: Sophie Burkhardt and Stefan Kramer. "Online Multi-label Dependency Topic Models for Text Classification". In: *Machine Learning* 107.5 (May 2018), pp. 859–886

2. Chapter 4 proposes a particularly efficient sampling method for non-parametric topic models. This sampling method enables fast training of relatively complex topic models.

   First published as: Sophie Burkhardt and Stefan Kramer. "Multi-Label Classification using Stacked Hierarchical Dirichlet Processes with Reduced Sampling Complexity". In: *ICBK 2017 - International Conference on Big Knowledge*. Hefei, China: IEEE, 2017, pp. 1–8
   and Sophie Burkhardt and Stefan Kramer. "Multi-Label Classification using Stacked Hierarchical Dirichlet Processes with Reduced Sampling Complexity". In: *Knowledge and Information Systems* (2018), pp. 1–23

3. Chapter 5 introduces the most efficient and effective available online training method for HDP topic models to my knowledge that could

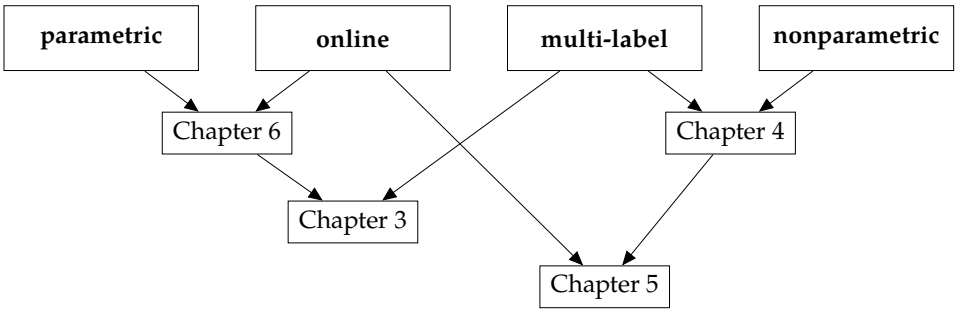


Figure 1.1: Overview of the methods introduced in this thesis. Chapter 6 proposes a parametric online model, Chapter 3 a parametric online multi-label model, Chapter 4 puts forward a nonparametric multi-label model, whereas Chapter 5 introduces a nonparametric online model that can also be made multi-label. Apart from the parametric online model, all models represent novel combinations of these subfields of topic model research.

potentially widen the applicability of such models substantially. Additionally, this method is applied for multi-label classification to show for the first time that nonparametric methods may yield competitive classification results on large-scale datasets.

First published as: Sophie Burkhardt and Stefan Kramer. "Online Sparse Collapsed Hybrid Variational-Gibbs Algorithm for Hierarchical Dirichlet Process Topic Models". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Michelangelo Ceci et al. Cham: Springer International Publishing, 2017, pp. 189–204

4. Chapter 6 introduces a scalable online topic model for the analysis of topics over time. In contrast to previous models, the topics stay consistent over time and each batch of data only has to be processed once.

   First published as: Zahra Ahmadi, Sophie Burkhardt, and Stefan Kramer. "Online Topic Modeling: Keeping Track of News Topics for Social Good". In: *Proceedings of the Second Workshop on Data Science for Social Good at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*. 2017

1.3 Structure

The remainder of this thesis consists of five chapters. Chapter 2 introduces relevant background in generative topic models (Section 2.1) as well as multi-label classification (Section 2.2). Both themes are brought together in discussing existing work on multi-label topic models (Section 2.3). The re-

maining four chapters introduce four different topic models. An overview is given in Figure 1.1 which shows how different subfields of topic model research are brought together in this thesis.Chapter 3 introduces a parametric online *multi-label* model, Chapter 4 proposes a *nonparametric* multi-label model, Chapter 5 combines three different fields to yield the first nonparametric online multi-label topic model and Chapter 6 presents a parametric online model. Combining new fields does not always automatically lead to good results, however, the remainder of this thesis will show the effectiveness of the proposed models.

Despite the advantages of generative models, in classification tasks, discriminative models are usually the preferred choice since in general they provide a higher classification performance. However, there are settings where it is still worth considering generative models for classification. In particular this affects settings with an extremely large number of labels. This is because most discriminative models become very resource intensive with increasing labelset size. We therefore present a multi-label classifier in Chapter 3 which efficiently models thousands of labels as well as dependencies between those labels and may be trained online on potentially infinite data streams. In addition to the competitive classification performance this model maintains interpretability, allowing to analyze why a certain document was assigned to a certain set of labels.

In Chapter 4, a method with more complex priors is presented that may be employed in the batch setting. Due to the complex prior, this method is better able to model different label frequencies. In this model, an emphasis is placed on the hierarchical architecture which is exploited to enable a more efficient training strategy. Thus, this complex model is applicable for datasets with hundreds to thousands of labels, yielding a classification performance competitive with discriminative classifiers, even on the smaller datasets.

The model introduced in Chapter 4 is a classification model that models label dependencies. The hierarchical structure of its priors however, is also exploitable in more basic unsupervised nonparametric topic models. This is shown in Chapter 5 where a fast online nonparametric topic model is proposed which uses a hybrid training strategy consisting of Gibbs sampling for the local counts and variational Bayes for the global parameter updates. The proposed sampling strategy leads to a new state-of-the-art nonparametric topic model in terms of training and testing time performance. Additionally, this model is applied in the supervised setting to show that it has a competitive classification performance on large-scale datasets.

Two important properties of generative topic models, interpretability and flexibility, are exploited in the last model presented in this thesis: "Online Topic Model for Keeping Track of News Topics" (Chapter 6). This model allows to track news topics over time by dividing the data into time slices and detecting topics in each time slice separately. Despite the

separate training on the different time slices, the different parameters are connected through the prior by using the parameters of the previous time slice to adjust the prior for the next time slice. Because of this the topics stay consistent over time and maintain a high quality even when the amount of training data for the time slices varies a lot. This model is very time efficient and may in principle be adjusted by adding or removing words or even whole topics. Thus, the model is flexible and easy to employ in practice.

To sum up, this thesis is concerned with showing the broad applicability of topic models in both unsupervised and supervised settings. The focus lies on text data and large-scale settings in terms of both dataset size and, in the case of classification models, labelset size. In addition to improvements in terms of speed, likelihood and classification performance, this thesis pushes the boundaries for the applicability of topic models by exploring previously untried combinations of different fields.

Chapter 2

Background

The background chapter is divided into three parts. First, the statistical background for topic models is presented. To do this, first, the relevant distributions are introduced in Section 2.1.1. Then latent Dirichlet allocation (LDA) is explained, the most prominent generative topic model, which is the basis for all models in this thesis (see Section 2.1.2). The two main training methods for LDA, sampling (Section 2.1.3) and variational Bayes (Section 2.1.4) are going to be discussed and the nonparametric variant of topic models, hierarchical Dirichlet processes (HDP), are described in Section 2.1.8. Finally, an overview of different existing topic models in the literature is given in Section 2.1.9. In the second part (Section 2.2), the broader theme of multi-label classification is introduced. I present the common evaluation measures used in this thesis (Section 2.2.1) as well as the important topic of threshold selection (Section 2.2.2). Two of the most prominent multi-label classifiers, binary relevance (Section 2.2.3) and classifier chains (Section 2.2.4) are introduced and finally a combination of both, called block classifier chains, is proposed in Section 2.2.5. The two parts on topic models and multi-label classification are then brought together in the last part on multi-label topic models, where the most prominent methods in this field, labeled LDA (LLDA, Section 2.3.1) and Dependency-LDA (Section 2.3.2), are described as well as the nonparametric extension of LLDA in Section 5.7.3.

2.1 Topic Models

2.1.1 Statistical Background

The generative topic models that are the subject of this thesis are based on the Dirichlet distribution and the Dirichlet process. I therefore give a brief introduction to the necessary statistical concepts required to understand the subsequent chapters.

An important part in the definition of the Dirichlet distribution is the

gamma function. It enables a generalization of the factorial function to real and complex numbers. For positive integers n, it is defined by

$$\Gamma(n) = (n-1)!.$$

The definition for complex numbers with a positive real part is given by

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\,\mathrm{d}t.$$

Based on this, the Beta function is defined as

$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}.$$

This leads to the Beta distribution, which has the probability density function

$$f(x|\alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1}.$$

This distribution may be used as a conjugate prior distribution for the Bernoulli or binomial distribution. $\alpha = \beta = 1$ results in a uniform distribution. When both parameters are greater than one, the most probable value is at the center of the distribution (i.e. for $\alpha = \beta$ this would be 0.5) whereas when both parameters are less than one, the distribution favours values at the extreme, i.e. values close to zero or one. The multivariate generalization of the Beta function is given by

$$B(\alpha) = \frac{\prod_i \Gamma(\alpha_i)}{\Gamma(\sum_i \alpha_i)}$$

This multivariate generalization serves as the normalizing constant for the Dirichlet distribution, which has the probability density function

$$f(x|\alpha) = \frac{1}{B(\alpha)} \prod_i x^{\alpha_i - 1}.$$

In the same way that the Beta distribution is conjugate to the Bernoulli and binomial distributions, the Dirichlet distribution is conjugate to the categorical and multinomial distributions.

The Dirichlet distribution has a predefined number of dimensions that correspond to the length of the parameter vector α. In the case where the number of dimensions is not given, a Dirichlet process (DP) may be used, which is the infinite-dimensional generalization of the Dirichlet distribution. Marginalizing over a finite partition of a DP results in a Dirichlet distribution. It can be used as a prior for a multinomial with a potentially unbounded number of topics. This means that drawing different multinomials from a DP results in multinomials of different sizes.

A DP may generally be constructed in three different ways. To provide a comprehensive overview, I briefly discuss all three paradigms.

1. **Stick-breaking process:** In the stick-breaking representation of the DP, we explicitly represent the probability distribution $G \sim DP(\alpha, H)$ as follows. Imagine a stick of length one. Now we break off a part of length β'_1. The remaining part has length $1 - \beta'_1$. Breaking off another part of the remaining stick, we are left with a stick of length $(1-\beta'_1)(1-\beta'_2)$. After breaking off $k-1$ parts of a stick, the length of the remainder is $\prod_{i=1}^{k-1}(1-\beta'_i)$. The probability G_k is given by the length of stick k, which is accordingly given by

$$G_k = \beta'_k \sum_{i=1}^{k-1}(1-\beta'_i).$$

   The variables β'_k are independently drawn from a univariate Beta distribution with parameter α, $B(\alpha, 1)$. The smaller parameter α, the more extreme are the values drawn from the Beta distribution and therefore smaller parts of the stick are left, which leads to more concentrated distributions.

2. **Pólya's urn process:** In Pólya's urn process, the distribution $G \sim DP(\alpha, H)$ is not represented explicitly, but through a series of draws θ_i from a distribution G which is distributed according to a DP with concentration parameter α and base distribution H. To do this, we start by drawing a color θ_0 from the base distribution H and add a ball of that color into the urn. In subsequent draws, we draw a ball from the urn with probability proportional to $n-1$, where n is the index of the current draw and a ball from the base distribution with probability proportional to α. If we draw a ball from the urn, we record its color θ_i and return the ball to the urn with a second ball of the same color.

   By the law of large numbers, as the number of balls approaches infinity, the distribution of the colors in the urn converges to a discrete distribution G. By just viewing the draws from the urn, the distribution G is integrated out or marginalized.

3. **Chinese restaurant process (CRP):** In the Chinese restaurant process representation, similar to the Pólya's urn process, we do not represent the distribution explicitly, but construct a series of samples from the distribution. We imagine a restaurant with an infinite number of tables. When a customer enters the restaurant, he chooses a table with a probability proportional to the number of customers already sitting at that table or sits at a new table with a probability proportional to α. Comparing this to Pólya's urn process, each customer corresponds to a ball and each table corresponds to a color. The difference is that the assignment of parameters to a partition is a separate step in the CRP.

Each table may be assigned to a different group or to the same group, making the CRP independent from the base distribution. In Pólya's urn process the partition and the assignment of the partition to one group is one step since each partition is associated with a color from the base distribution. This means that when the same color is chosen twice from the base distribution, it is not possible anymore to distinguish between balls belonging to the first draw and balls belonging to the second draw. Therefore the CRP is more flexible and used more often, especially in settings with hierarchies of distributions.

The methods introduced in this thesis are based on the CRP. However, some comparison methods are based on the stick-breaking process.

2.1.2 Latent Dirichlet Allocation (LDA)

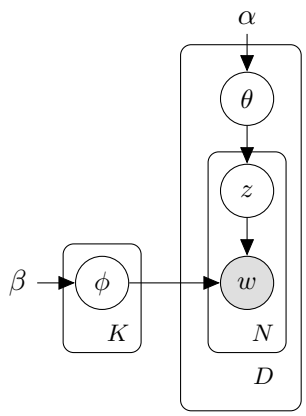


Figure 2.1: The graphical model of LDA.

After having introduced the mathematical background, this section describes how the Dirichlet and multinomial[1] distributions are used to model collections of text documents. Latent Dirichlet allocation (LDA) [8] is a generative model of document collections where each document is modeled as a mixture of latent topics (see Equation 2.1). LDA is built on the assumption that words as well as documents are exchangeable, which means that the order in which words or documents are viewed plays no role in the training process. With respect to words, this assumption is often called the "bag-of-words" assumption, meaning that each document is viewed as a bag of

[1]The term multinomial is used ambiguously in the literature on topic models. We use it to refer to the discrete distribution which models only one draw instead of a series of draws.

words where the actual sequence of words makes no difference.

$$\begin{aligned} w_{di}|z_{di}, \phi_{z_{di}} &\sim Multinomial(\phi_{z_{di}}) \\ \phi &\sim Dirichlet(\beta) \\ z_{di}|\theta_d &\sim Multinomial(\theta_d) \\ \theta &\sim Dirichlet(\alpha) \end{aligned} \tag{2.1}$$

The generative process is given as follows:

- For each topic $k \in 1, \ldots, K$
  - draw $\phi_k \sim Dirichlet(\beta)$
- For each document $d \in D$
  - draw $\theta_d \sim Dirichlet(\alpha)$
  - For each word token with indices $i = 1, \ldots, N_d$ in document d (N_d is the number of words in document d)
    - draw topic indicator $z_{di} \sim \theta_d$
    - draw word $w_{di} \sim \phi_{z_{di}}$

Each topic $k \in 1, \ldots, K$ is represented by a multinomial distribution ϕ_k over words that is assumed to be drawn from a Dirichlet distribution with parameter β. Document d is generated by drawing a distribution over topics from a Dirichlet $\theta_d \sim Dirichlet(\alpha)$, and for the ith word token in the document, first drawing a topic indicator $z_{di} \sim \theta_d$ and finally drawing a word $w_{di} \sim \phi_{z_{di}}$.

To learn a model over an observed document collection D, we need to compute the posterior distribution over the latent variables z, θ, and ϕ which is in general intractable to compute directly.

$$p(\phi, \theta, z|D, \alpha, \beta) = \prod_{k=1}^{K} p(\phi_k|\beta) \prod_{d=1}^{D} p(\theta_d|\alpha) \prod_{i=1}^{N_d} p(z_{di}|\theta_d) p(w_{di}|\phi_{z_{di}}) \tag{2.2}$$

Therefore, the posterior distribution needs to be estimated. There are two main methods that are commonly used to do this: Gibbs sampling and variational Bayes. I now describe each of the two methods.

2.1.3 Gibbs Sampling for LDA

Gibbs sampling is a special case of Markov chain Monte Carlo sampling (MCMC). Hereby, each variable is sampled conditioned on all other variables, which are held fixed. Since the Dirichlet distribution is conjugate to

the multinomial distribution, it is possible to integrate/collapse out the latent variables ϕ and θ from the joint distribution $p(w, z, \phi, \theta)$, where w and z are the word and topic variables for all tokens i and documents d:

$$
\begin{aligned}
&p(w, z) = p(w|z)p(z) = \\
&\int\int \prod_k p(\phi_k) \prod_d p(\theta_d) \prod_i p(w_{di}|\phi_{z_{di}}) p(z_{di}|\theta_d) \,\mathrm{d}\phi \,\mathrm{d}\theta = \\
&\int \prod_d \prod_i p(w_{di}|\phi_{z_{di}}) \prod_k p(\phi_k) \,\mathrm{d}\phi \int \prod_d \prod_i p(z_{di}|\theta_d) \prod_d p(\theta_d) \,\mathrm{d}\theta
\end{aligned}
$$

This enables efficient model training.

The two integrals can be performed separately. Assuming that the ith word is given as w, the first term is calculated as follows:

$$
\begin{aligned}
p(w_{di} = w|z_{-di}) &= \int \prod_d \prod_i p(w_{di}|\phi_{z_{di}}) \prod_k p(\phi_k|z_{-i}, \beta) \mathrm{d}\phi && (2.3)\\
&= \prod_k \int \phi_{kw}^{n_{wk}} \frac{\Gamma(\sum_{v \in V} \beta_v)}{\prod_{v \in V} \Gamma(\beta_v)} \phi_{kw}^{\beta_w - 1} \,\mathrm{d}\phi && (2.4)\\
&= \prod_k \frac{\Gamma\left(\sum_{v \in V} \beta_v\right)}{\prod_{v \in V} \Gamma\left(\beta_v\right)} \int \phi_{kw}^{n_{wk}} \phi_{kw}^{\beta_w - 1} \,\mathrm{d}\phi && (2.5)\\
&= \prod_k \frac{\Gamma\left(\sum_{v \in V} \beta_v\right)}{\prod_{v \in V} \Gamma\left(\beta_v\right)} \int \phi_{kw}^{n_{wk} + \beta_w - 1} \,\mathrm{d}\phi && (2.6)\\
&= \prod_k \frac{\Gamma\left(\sum_{v \in V} \beta_v\right)}{\prod_{v \in V} \Gamma\left(\beta_v\right)} \frac{\Gamma\left(n_{wk} + \beta_w\right)}{\Gamma\left(\sum_{v \in V} \left(n_{vk} + \beta_v\right)\right)} && (2.7)
\end{aligned}
$$

where z_{-di} are all topic indicators except the one for the ith token in document d, n_{wk} denotes the number of times word w and topic k occur together, whereas n_{-wk} denotes the number of times word w and topic k occur together if the ith token is excluded from the count. The definitions of the multinomial and Dirichlet densities are inserted in Equation 2.4, the exponents of ϕ are combined in Equation 2.5 and 2.6, the integral is removed by adding the normalizing constant of the Dirichlet distribution so that the integral evaluates to one in Equation 2.7.

Similarly, for the second term by focusing on a specific document d we

get:

$$p(z_{di} = k|z_{-di}) = \int p(\theta_d) \prod_d \prod_i p(z_{di}|\theta_d) \, \mathrm{d}\theta_d = \tag{2.8}$$

$$\int \prod_k \theta_{dk}^{n_{dk}} \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_k \theta_{dk}^{\alpha_k - 1} \, \mathrm{d}\theta_d \tag{2.9}$$

$$= \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \int \prod_k \theta_{dk}^{n_{dk}} \prod_k \theta_{dk}^{\alpha_k - 1} \, \mathrm{d}\theta_d \tag{2.10}$$

$$= \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \int \prod_k \theta_{dk}^{n_{dk} + \alpha_k - 1} \, \mathrm{d}\theta_d \tag{2.11}$$

$$= \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \frac{\prod_k \Gamma(n_{dk} + \alpha_k)}{\Gamma(\sum_k (n_{dk} + \alpha_k))} \tag{2.12}$$

where n_{dk} is the number of times topic k occurs in document d.

To see how we arrive at the complete sampling equation, we have to note that $\Gamma(n+1) = n\Gamma(n)$. Denote by n_{-j} the counts excluding the current variable z_i. The next steps are:

$$\begin{aligned} & \prod_{j=1}^{k} \Gamma(n_j + \alpha_j) \\ &= \Gamma(n_k + \alpha_k) \prod_{j \neq k} \Gamma(n_j + \alpha_j) \\ &= \Gamma(n_{-k} + 1 + \alpha_k) \prod_{j \neq k} \Gamma(n_{-j} + \alpha_j) \\ &= (n_{-k} + \alpha_k)\Gamma(n_{-k} + \alpha_k) \prod_{j \neq k} \Gamma(n_{-j} + \alpha_j) \\ &= (n_{-k} + \alpha_k) \prod_{j=1}^{k} \Gamma(n_{-j} + \alpha_j) \propto n_{-k} + \alpha_k \end{aligned} \tag{2.13}$$

Now we want to apply Gibbs sampling, so we need to sample from the posterior for a specific $z_{di} = k$ given all remaining variables denoted by z_{-i} and all words w. Therefore, the derivation in Equation 2.13 is used to further transform Equation 2.7 and 2.12 by focusing only on a particular word w and topic k:

$$\prod_k \frac{\Gamma\left(\sum_{v \in V} \beta_v\right)}{\prod_{v \in V} \Gamma\left(\beta_v\right)} \frac{\Gamma\left(n_{wk} + \beta_w\right)}{\Gamma\left(\sum_{v \in V} \left(n_{vk} + \beta_v\right)\right)} \propto \frac{n_{-wk} + \beta_w}{\sum_v \left(n_{-vk} + \beta_v\right)} \tag{2.14}$$

$$\frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \frac{\prod_k \Gamma(n_{dk} + \alpha_k)}{\Gamma(\sum_k (n_{dk} + \alpha_k))} \propto n_{-dk} + \alpha_k \tag{2.15}$$

Finally, the conditional probabilities for training an LDA topic model are given by [23]:

$$p(z_{di} = k|z_{-di}, w) \propto \frac{n_{wk} + \beta_w}{\sum_{w'} (n_{w'k} + \beta_{w'})} (n_{dk} + \alpha_k), \tag{2.16}$$

where n_{wk} and n_{dk} are the respective counts of topics k with words w or in documents d. α and β are hyperparameters as before. z_{-di} are all topic indicators except the one for token i in document d.

Intuitively, Equation 2.16 consists of two parts, where the first part describes the probability of a word in a certain topic. This part is responsible for words preferentially being assigned to topics where they already occur in, thus exploiting the clustering effect of the Dirichlet distribution. The second part is proportional to the probability of a topic in a certain document. Therefore, while the first part may be seen as ensuring the consistency with the global model and its topics, the second part ensures that each document minimizes the number of topics it exhibits at the local level, ensuring that a topic is more likely if other words in the same document have already been assigned to this topic.

2.1.4 Variational Bayes for LDA

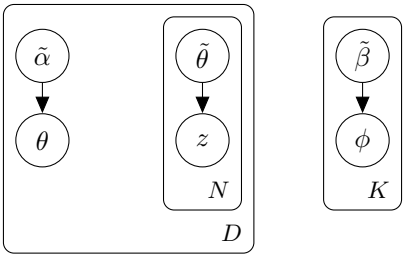


Figure 2.2: The graphical model of the variational distribution used to approximate the posterior of LDA.

In variational Bayesian inference a variational distribution is introduced to approximate the posterior by minimizing the Kullback-Leibler (KL) divergence between the variational distribution q and the true posterior.

$$\begin{aligned} \mathrm{KL}[q(z|D)||p(z|D)] &= \sum_z q(z|D) \log \frac{q(z|D)}{p(z|D)} \\ &= \mathbb{E}_{q(z|D)} \left[\log q(z|D) - \log p(z|D)\right] \end{aligned}$$

Usually, a fully factorized variational distribution is chosen:

$$q(\phi, \theta, z|\tilde{\beta}, \tilde{\alpha}, \tilde{\theta}) = \prod_d^D q(\theta_d|\tilde{\alpha}_d) \prod_i^{N_d} q(z_{di}|\tilde{\theta}_{di}) \prod_k^K q(\phi_k|\tilde{\beta}_k), \tag{2.17}$$

where $\tilde{\beta}, \tilde{\alpha}$ and $\tilde{\theta}$ denote the variational parameters.

The evidence lower bound (ELBO) that is to be maximized is given as follows:

$$\log p(W|\alpha,\beta) \geq \mathcal{L}(\tilde{\beta},\tilde{\alpha},\tilde{\theta}) \triangleq \mathbb{E}_q[\log p(\phi,\theta,z,W)] + \mathcal{H}(q(\phi,\theta,z)) \quad (2.18)$$

$$= \mathbb{E}_q[\log p(\theta|\alpha)] + \mathbb{E}_q[\log p(z|\theta)] + \mathbb{E}_q[\log p(w|z,\phi)] + \mathcal{H}(q(\phi,\theta,z)), \quad (2.19)$$

where $\mathcal{H}$ denotes the entropy and in the first step the log-likelihood is lower bounded using Jensen's inequality.

Equation 2.19 is expanded to yield:

$$\begin{aligned}
&\mathcal{L}(\tilde{\beta},\tilde{\alpha},\tilde{\theta}) = \\
&\log\Gamma\left(\sum_K \alpha_k\right) - \sum_K \log\Gamma(\alpha_k) + \sum_K (\alpha_k - 1)\left(\Psi(\tilde{\alpha_k}) - \Psi\left(\sum_{j=1}^K \tilde{\alpha_j}\right)\right) \\
&+ \sum_N \sum_K \tilde{\theta}_{ki}\left(\Psi(\tilde{\alpha_k}) - \Psi\left(\sum_{j=1}^K \tilde{\alpha_j}\right)\right) \\
&+ \sum_N \sum_K \sum_V \tilde{\theta}_{ki} w_i^v log\phi_{vk} \\
&- \log\Gamma\left(\sum_K \tilde{\alpha_k}\right) + \sum_K \log\Gamma(\tilde{\alpha_k}) - \sum_K (\tilde{\alpha_k} - 1)\left(\Psi(\tilde{\alpha_k}) - \Psi\left(\sum_K \tilde{\alpha_k}\right)\right) \\
&- \sum_N \sum_K \tilde{\theta}_{ki} log\tilde{\theta}_{ki} \\
&- \log\Gamma\left(\sum_V \tilde{\beta_v}\right) + \sum_V \log\Gamma\left(\tilde{\beta_v}\right) - \sum_V \left(\tilde{\beta_v} - 1\right)\left(\Psi\left(\tilde{\beta_v}\right) - \Psi\left(\sum_V \tilde{\beta_v}\right)\right)
\end{aligned}$$

By calculating the gradient of the ELBO with respect to the variational parameters, the parameters can be updated until convergence.

First, all terms containing $\tilde{\alpha}_{dk}$ are isolated:

$$
\begin{aligned}
&\mathcal{L}_{\tilde{\alpha}} = \\
&\sum_K (\alpha_k - 1)\left(\Psi\left(\tilde{\alpha_k}\right) - \Psi\left(\sum_{j=1}^{K}\tilde{\alpha_j}\right)\right) + \sum_N\sum_K \tilde{\theta}_{ki}\left(\Psi\left(\tilde{\alpha_k}\right) - \Psi\left(\sum_{j=1}^{K}\tilde{\alpha_j}\right)\right) \\
&- \log\Gamma\left(\sum_K \tilde{\alpha_k}\right) + \sum_K \log\Gamma\left(\tilde{\alpha_k}\right) - \sum_K \left(\tilde{\alpha_k} - 1\right)\left(\Psi\left(\tilde{\alpha_k}\right) - \Psi\left(\sum_K \tilde{\alpha_k}\right)\right) \\
&= \sum_K \left(\Psi\left(\tilde{\alpha_k}\right) - \Psi\left(\sum_{j=1}^{K}\tilde{\alpha_j}\right)\right)\left(\alpha_k + \sum_N \tilde{\theta}_{ki} - \tilde{\alpha_k}\right) \\
&- \log\Gamma\left(\sum_K \tilde{\alpha_k}\right) + \sum_K \log\Gamma\left(\tilde{\alpha_k}\right)
\end{aligned}
$$

By taking the gradient of this and setting it to zero, we arrive at the local/document-level update equations for variational Bayes [8, 58]:

$$\tilde{\alpha}_{dk} = \alpha + \sum_{i=1}^{N_d} \tilde{\theta}_{dki} \tag{2.20}$$

Updates for the other parameters are derived in the same way.

$$\tilde{\theta}_{dki} \propto \exp\left(\Psi(\tilde{\beta}_{wk}) - \Psi(\sum_v \tilde{\beta}_{vk})\right)\exp\left(\Psi(\tilde{\alpha}_{dk}) - \Psi(\sum_{k'} \tilde{\alpha}_{dk'})\right) \tag{2.21}$$

where we have to note that for the expectation of the log-Dirichlet we have $\mathbb{E}[\log\theta|\alpha] = \Psi(\alpha) - \Psi(\sum_k \alpha_k)$ and Ψ is the digamma function.

However Teh *et al.* [58] proposed a collapsed version of variational Bayes where the parameters are marginalized. By using a Gaussian approximation and a Taylor expansion, that will not be explained in detail here, they arrive at the update equation

$$\tilde{\theta}_{dki} \propto \frac{\tilde{\beta}_{wk} + \beta}{\sum_{w'}(\tilde{\beta}_{w'k} + \beta_{w'})}(\tilde{\alpha}_{dk} + \alpha), \tag{2.22}$$

where α and β are hyperparameters and N_d is the number of words in document d. This equation has a strong similarity with the sampling equation for collapsed Gibbs sampling (Section 2.1.3). It is shown by Teh *et al.* [58] and Asuncion *et al.* [4] that this collapsed version called CVB0 [4, 19] has a better convergence than the uncollapsed one. This is why in this thesis only the collapsed version is used.

Algorithm 1 Batch Variational Bayes

```
1: while not converged do
2:   for each document d do
3:     for each word token do
4:       update local parameters (Equation 2.22)
5:       normalize θ̃_di to sum to one
6:     end for
7:     update local parameters (Equation 2.20)
8:   end for
9:   update global parameters (Equation 2.23)
10: end while
```

Based on the local variational parameters $\tilde{\theta}$, the global parameter $\tilde{\beta}$ can be updated as follows:

$$\tilde{\beta}_{vk} = \beta + \sum_{d=1}^{|D|} \sum_{i=1}^{N_d} \tilde{\theta}_{dki} \mathbb{1}[w_{di} = v], \tag{2.23}$$

where $|D|$ is the number of documents. $\mathbb{1}[w_{di} = v]$ is one if word $w_{di} = v$ and zero otherwise.

2.1.5 Batch Variational Bayes

For the batch variational Bayes algorithm, all local variational parameters $\tilde{\theta}_d$ for all documents d are computed and then the global parameter is updated in one step. The algorithm (see Algorithm 1) may also be described as consisting of an expectation/E-step and a maximization/M-step:

- *E-Step*: For each document, the local variational parameters are optimized (lines 2–8).
- *M-Step*: The lower bound on the log-likelihood is maximized with respect to the global variational parameters (line 9).

2.1.6 Online Variational Bayes

In the batch method, all local parameters have to be kept in memory and have to be updated at once. For large datasets this can lead to a large memory consumption and slow down training especially in the beginning. Therefore an online method based on minibatches was introduced [27] that converges faster and is efficient to train on large datasets.

The methods proposed in this thesis build on the following previous work. There exists an online variant for streaming data based on variational Bayes introduced by Hoffman *et al.* [27, 28]. Teh *et al.* [58] and Asuncion *et*

Algorithm 2 Online Variational Bayes

```
1: while not converged do
2:   draw minibatch M
3:   for each document d ∈ M do
4:     for each word token do
5:       update local parameters (Equation 2.22)
6:       normalize θ̃_di to sum to one
7:     end for
8:     update local parameters (Equation 2.20)
9:   end for
10:  update global parameters (Equation 2.24)
11: end while
```

al. [4] improved this work by collapsing out the latent variables. Foulds *et al.* [19] combine the online part of Hoffman *et al.* and Cappe and Moulines [15], and the collapsing part of Asuncion *et al.*, resulting in an online stochastic collapsed variational Bayes (SCVB) with improved performance. The method by Foulds *et al.* is the basis of the work presented in this thesis.

The online variational Bayes algorithm is summarized in Algorithm 2. It is similar to the batch algorithm except we now iterate over the documents minibatch by minibatch instead of over all documents at once and use a different update equation for the global parameters (line 10). Updating variational parameters $\tilde{\beta}$ for one minibatch M is done as follows, where the counts for one minibatch are scaled by $\frac{|D|}{|M|}$ to arrive at the expectation for the whole corpus and ρ_t is a parameter between zero and one.

$$\tilde{\beta}_{vk} = (1 - \rho_t)\tilde{\beta}_{vk} + \rho_t \left(\beta + \frac{|D|}{|M|} \sum_{d \in M} \sum_{i=1}^{N_d} \tilde{\theta}_{dki} \mathbb{1}[w_{di} = v] \right), \tag{2.24}$$

Given appropriate updates and choice of hyperparameter ρ_t, the online algorithm is guaranteed to converge to the optimal variational solution. Since the global parameters are updated after each minibatch instead of each iteration over the whole dataset, the online algorithm usually converges faster than the batch algorithm, especially in the beginning of training.

2.1.7 Gibbs Sampling vs. Variational Bayes

Convergence of Gibbs sampling can be slow in comparison to variational methods, since updates only involve a sampled topic instead of the full distribution over topics as in variational Bayes. On the other hand, Gibbs sampling is unbiased, meaning it is guaranteed to learn the true posterior after an infinite number of iterations. How to determine if a Gibbs sampler has converged, however, is still an open problem. Another advantage of Gibbs

samplers are the sparse updates. One word-topic assignment is updated at a time so the global counts can be efficiently updated for each document by only decrementing the counts of the previous word-topic assignments and incrementing counts for the new word-topic assignments. Variational Bayes updates are dense in comparison because the whole distribution over topics is kept for each word, making frequent updates inefficient. This is one reason why updates are usually done in minibatches. While variational Bayes converges generally faster than Gibbs sampling, the method is biased and not guaranteed to arrive at the true posterior.

The second main difference between variational Bayes and Gibbs sampling is that variational Bayes topic models may be trained online, one minibatch at a time. Gibbs sampling on the other hand is a batch method. Convergence of Gibbs sampling is only guaranteed if all data is kept available and all topic assignments keep being updated until convergence. While it is theoretically possible to train it online by only sampling topic assignments once, in practice this is only successful in restricted settings where the labels/topics for each document are already known and even then there is no guarantee for convergence. Particle samplers provide a way around this, but are generally not practical and efficient. This is why for streaming settings and models that have to be continuously updated, variational Bayesian methods are the preferred choice. In chapter 5, a hybrid method is introduced that combines advantages from both methods.

2.1.8 Nonparametric Topic Models

Overview

Nonparametric topic models are based on hierarchical Dirichlet processes (HDPs). In HDP topic models [59], the multinomial distribution θ from LDA is drawn from an HDP instead of a Dirichlet distribution:

$$\theta \sim DP(G_0, b_1), G_0 \sim DP(H, b_0).$$

The base distribution G_0 of the first Dirichlet process (DP) is again drawn from a DP with base distribution H. This is why it is called a *hierarchical* DP. Dirichlet processes (DPs) [59] are distributions over probability measures. If a distribution over topics is drawn from a DP, the number of topics is not fixed. This is why such models are called *nonparametric*.

Because the prior is hierarchical, there is a local topic distribution θ for each document and a global topic distribution G_0 which is shared among all documents. The advantage of this global topic distribution is that it allows topics of widely varying frequencies whereas in standard LDA with a symmetric prior α, all topics are expected to have the same frequency. The asymmetric prior of HDP usually leads to a better representation and higher log-likelihood of the dataset [59].

A generalization of the HDP is given by the hierarchical Poisson-Dirichlet process (HPDP), sometimes also called hierarchical Pitman-Yor process. In this stochastic process, an additional parameter a is the so-called discount parameter. For $a = 0$ the process reduces to the normal HDP. In this thesis, the experiments are based on the HDP, but all these models could also be extended to the HPDP by simply letting $a \neq 0$.

Sampling methods for HDPs are mostly based on the Chinese restaurant process metaphor. Each word token is assumed to be a customer entering a restaurant, and sitting down at a certain table where a specific dish is served. Each table is associated with one dish, which corresponds to a topic in a topic model. The probability for a customer to sit down at a certain table is proportional to the number of customers already sitting at that table. This leads to a clustering effect where new customers are most likely to sit at a table that already has attracted a large number of customers. With a certain probability α (see Equation 2.25), the customer sits down at a new table. In this case a topic is sampled from the base distribution. For an HDP topic model, each document corresponds to a restaurant. The topics in each document-restaurant are drawn from a global restaurant. Because all documents share the same base distribution that is discrete, represented by the global restaurant, the topics are shared by the local document restaurants. If a new table is added to a document restaurant, a pseudo customer enters the global restaurant (see Figure 2.3). If a new table is opened in the global restaurant, a new topic is added to the topic model.

$$P(z_n = k|z_1 \dots z_{n-1}) = \begin{cases} \frac{n_k}{n-1+\alpha} & , n_k > 0 \\ \frac{\alpha}{n-1+\alpha} & , new\ table \end{cases} \tag{2.25}$$

In terms of the statistics that need to be kept, in the basic version we need to store for each word not only the sampled topic, but also the table it is associated with. Also, we need to store the corresponding topic for each table.

The generative model for the two-level HDP topic model is given as follows:

$$\theta_0|b_0, H \sim DP(b_0, H), \qquad \theta_d|b_1, \theta_0 \sim DP(b_1, \theta_0)$$
$$z|\theta_d \sim \theta_d, \qquad w|z \sim Mult(z)$$

Each word w is assumed to be drawn from a multinomial distribution associated with a certain topic z. The topic indicator variable z is drawn from a document-specific distribution over topics θ_d, which is in turn drawn from a DP with base distribution θ_0. θ_0 is drawn from another DP with base distribution H. b are hyperparameters.

Three basic sampling methods were introduced by Teh *et al.* [59]. Two methods are directly based on the Chinese restaurant representation, whereas the third is the direct assignment sampler. The first two methods

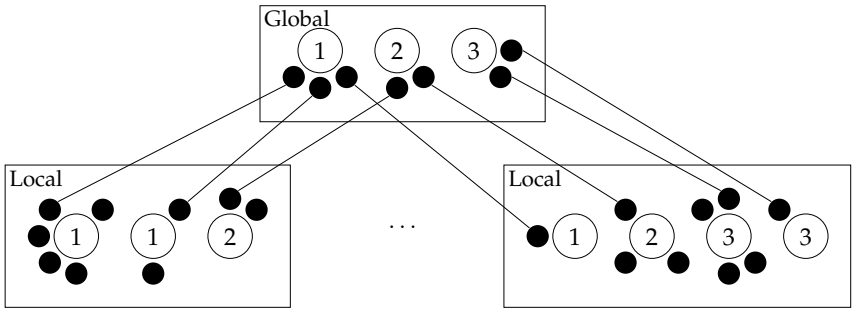


Figure 2.3: Illustration of a hierarchical Chinese restaurant process. For each table at a local restaurant one customer is sent to the global restaurant. The numbers represent different topics.

sample a table for each word and a topic for each table. This can be slow and requires to store separate counts for each table. The direct assignment sampler does not sample an individual table but assigns a topic to each word token directly, and instead of keeping the statistics for each table separately, it simply samples the number of tables that are associated with a certain topic. While this sampler has improved convergence over the other sampling methods, it needs to sample $M_k \in \{1, \ldots, N_k\}$, the number of tables for topic k, which can be inefficient when the number of customers per topic N_k is large. A further improved version was therefore introduced by Chen *et al.* [16]. In this version another auxiliary variable u is introduced which is sampled for each customer and determines whether or not the customer sits down at a new table or an existing one. This is then used to update the table count s_k. u itself does not have to be kept in memory but can be sampled when needed. This way, the memory requirements are similar to Teh's auxiliary variable sampler, the sampling process itself is more efficient, and convergence is improved. For this reason the methods proposed in this thesis are based on the sampling method by Chen *et al.*.

Sampling Methods

After this general overview, I now describe the different sampling methods in more detail. The most basic sampling method is based on the Chinese restaurant metaphor. Integrating out the document-topic distribution θ_j, the probability of sampling table t for the ith customer is given by

$$p(t_i = t|t_{-i}) \propto \begin{cases} n_{jt} \cdot p(w_i|z_i = k) & //\text{existing table} \\ \alpha \cdot p(w_i|t_i = t_{new}) & //\text{new table} \end{cases}, \tag{2.26}$$

where the probability of sitting at an existing table is proportional to the number of customers n_{jt} sitting at table t in restaurant j, whereas the probability for opening a new table in restaurant j is proportional to α. $p(w_i|z_i =$

k) is the prior probability of word w_i given topic k. If a new table is opened, a topic is assigned to this new table according to

$$p(k_t = k|t, k) \propto \begin{cases} m_{\cdot k} \cdot p(w_i|z_i = k) & \text{if } k \text{ exists} \\ \gamma \cdot p(w_i|z_i = k) & \text{if } k \text{ is new} \end{cases}, \tag{2.27}$$

where $m_{\cdot k}$ corresponds to the overall number of tables serving topic k in all restaurants.

For this sampler we need to store the table-topic assignments ψ_l as well as the customer-table assignments t_i, where ψ_l is the topic assignment for the lth table.

An augmented version of this sampler does not integrate out the global distribution G_0 which makes sampling in more complex models easier. In this case, G_0 is instantiated by using another variable called b which is sampled from a Dirichlet distribution based on the table counts m:

$$b_1, \ldots, b_K \sim \text{Dirichlet}(m_{\cdot 1}, \ldots, m_{\cdot K}, \gamma) \tag{2.28}$$

See Teh *et al.* [59] for further details.

The third sampler by Teh *et al.* is the direct assignment sampler. In this sampler, instead of sampling a table for each token, we directly sample the topic.

$$p(z_i = k|z_{-i}, m, \beta) \propto \begin{cases} (n_{jk} + \alpha_0\beta_k) \cdot p(w_i|z_i = k) & \text{if k exists} \\ \alpha_0 b_0 \cdot p(w_i|z_i = k) & \text{if k is new} \end{cases} \tag{2.29}$$

The number of tables per topic needs to be sampled separately in the direct assignment sampler according to:

$$p(m_{jk} = m|z, m_{-jk}, b) = S_{m,0}^{n_{jk}}(\alpha_0 b_k)^m \frac{\Gamma(\alpha_0 b_k)}{\Gamma(\alpha_0 b_k + n_{jk})}, \tag{2.30}$$

where m_{-jk} is the number of tables is restaurant j with topic k excluding the current token. This sampling step can be inefficient if the number of customers n_{jk} for topic k is high since the probability for every table count up to n_{jk} has to be calculated. b is sampled as for the augmented sampler in Equation 2.28.

Table Indicator Sampling

The currently most efficient sampling method for HDPs is by Chen *et al.* [16]. Here, an additional variable, the table indicator u, is introduced, which indicates up to which level a customer has a table contribution. In the case of a two-level HDP, $u = 2$ means the customer sits at an existing table, $u = 1$ means the customer opens a new table at the lowest level and sends

a pseudo customer up to the next level, whereas $u = 0$ means that the customer opens a new table at the lowest level, sends a pseudo customer to the next level, which again opens a new table thereby adding a new topic to the topic model (see Fig. 2.3).

I now briefly explain how to arrive at the table indicator representation. The starting point is the direct assignment sampler by Teh *et al.* [59]. In this sampler each customer is directly assigned a topic and the number of tables per topic M_k is sampled separately. (Note that M_k corresponds to $m_{\cdot k}$ in the previous section. The capital letters N, and M now refer to global counts over all documents, whereas n and m refer to local document counts.)

The probability for the number of tables M_k for topic k given the number of customers per topic N_k is given by (see Buntine and Hutter [12], Lemma 8)

$$p(M_k|N_k, a, b) = \frac{(b|a)_{M_k}}{(b)_{N_k}} S^{N_k}_{M_k,a}, \tag{2.31}$$

where $S^n_{m,a}$ is a generalized Stirling number defined by the recursion $S^{n+1}_{m,a} = S^n_{m-1} + (n - ma)S^n_{m,a}$, for $m \leq n$. It is zero otherwise and $S^0_{0,a} = S^1_{1,a} = 1$.

Note that in this work only the standard Dirichlet process is considered, which is the special case of the Poisson-Dirichlet process (PDP) for $a = 0$. The parameter $b > 0$ is usually estimated and n_m is the number of customers at table m. $(x)_N$ denotes the Pochhammer symbol $x \cdot (x + 1) \cdot \ldots \cdot (x + N - 1) = \frac{\Gamma(x+N)}{\Gamma(x)}$ and $(x|y)_N$ denotes the Pochhammer symbol with increment y, $x \cdot (x + y) \cdot \ldots \cdot (x + (N - 1)y)$, and $(x|0)_N = x^N$.

For $a = 0$, Equation 2.31 becomes (shown by Antoniak [3], compare Teh *et al.* [59], Equation 40):

$$p(M_k|N_k, b) = S^{N_k}_{M_k,0} b^{M_k} \frac{\Gamma(b)}{\Gamma(b + N_k)}.$$

Applying this result to the PDP with base distribution H, the joint probability of the samples z_i and the number of tables $M_1, M_2, \ldots, M_K$ for each topic $k = 1, \ldots, K$ is

$$p(z_1, z_2, \ldots, z_N, M_1, \ldots, M_K) = \frac{(b|a)_{M_k}}{(b)_{N_k}} \prod_{k=1}^{K} (H(k) S^{N_k}_{M_k,a})$$

Now, this representation with the number of tables per topic M_k is converted to the representation with table indicators u_i that are assigned to each token and specify at which levels this token has a table contribution. E.g., if $u_1 = 0$, the first token contributes to the table count at all levels, whereas in the case of two levels and $u_1 = 2$ the token does not contribute to the table count, i.e. the customer sits at an existing table. It is possible to

reconstruct the original representation from the table indicator representation as follows [16]:

$$p(z, M) = \prod_k \frac{N_k!}{M_k!(N_k - M_k)!} p(z, u)$$

since given one table configuration, there are

$$\prod_k \binom{N_k}{M_k} = \prod_k \frac{N_k!}{M_k!(N_k - M_k)!}$$

possible table indicator configurations.

The joint posterior distribution of the hierarchical PDP given base distribution H_0 for the root node is now given by

$$p(z, u|H_0) = \prod_{j \geq 0} \left(\frac{(b|a)_{M_j}}{(b)_{N_j}} \prod_{k=1}^{K} S^{n_{jk}}_{m_{jk},a} \frac{m_{jk}!(n_{jk} - m_{jk})!}{n_{jk}!} \right),$$

where j is the index of the hierarchy level, N_j and M_j are the overall number of customers and tables for restaurant j and n_{jk} and m_{jk} are the respective counts for topic k.

To get the posterior distribution for a specific topic $z_i = k$ and table indicator $u_i = u$, application of the chain rule yields [16]

$$p(z_i = k, u_i = u|z_{-i}, u_{-i}, H) = \frac{p(z, u, H)}{p(z_{-i}, u_{-i}, H)} = \prod_{j \in path} \frac{(b_j + a_j M_j)^{\delta_{M'_j \neq M_j}}}{(b_j + N_j)^{\delta_{N'_j \neq N_j}}}$$
$$\left(\frac{S^{n'_{jk}}_{m'_{jk},a}}{S^{n_{jk}}_{m_{jk},a}} \right)^{\delta_{n'_{jk} \neq n_{jk} || m'_{jk} \neq m_{jk}}} \frac{(m'_{jk})^{\delta_{m'_{jk} \neq m_{jk}}} (n'_{jk} - m'_{jk})^{\delta_{n'_{jk} - m'_{jk} \neq n_{jk} - m_{jk}}}}{(n'_{jk})^{\delta_{n'_{jk} \neq n_{jk}}}}.$$

Here the path consists of the restaurants in the hierarchy where the customer has a table contribution. The subscript $-i$ refers to all variables except the one with index i, N', M', n' and m' are the counts after adding the current token to the counts and N, M, n and m refer to counts before the addition of token i. The ratio of Pochhammer symbols $(x|y)_{N+1}/(x|y)_N$ reduces to $x + Ny$, whereas $(x)_{N+1}/(x)_N = \Gamma(x + N + 1)/\Gamma(x + N) = x + N$. S^n_m are generalized Stirling numbers of the first kind whose ratios can be efficiently precomputed and retrieved in $O(1)$.[2]

[2]See Buntine and Hutter [12] for an efficient way to compute ratios of these numbers. They can be precomputed once and subsequently retrieved in $O(1)$. Note that it may be necessary to store large values sparsely if the number of tokens in a restaurant becomes large.

Finally, the sampling equations of the full 2-level HDP topic model for the joint sampling of topic and table indicator are as follows [16]. $rest$ is an abbreviation referring to all remaining variables.

If the topic is new for the root restaurant (table indicator is zero):

$$P(z_i = k_{new}, u_i = 0|rest) \propto \frac{b_0 b_1}{(M_\cdot + b_0)(N_j + b_1)} P_{wk_{new}} \tag{2.32}$$

If the topic is new for the base restaurant (e.g. a document), but not for the root restaurant (table indicator is one):

$$P(z_i = k, u_i = 1|rest) \propto \frac{b_1 M_k^2}{(M_k + 1)(M_\cdot + b_0)(N_j + b_1)} P_{wk} \tag{2.33}$$

If the topic exists at the base restaurant and an already existing table is chosen (table indicator is two):

$$P(z_i = k, u_i = 2|rest) \propto \frac{S_{m_{jk}}^{n^{jk}+1}}{S_{m_{jk}}^{n_{jk}}} \frac{n_{jk} - m_{jk} + 1}{(n_{jk} + 1)(N_j + b_1)} P_{wk} \tag{2.34}$$

If the topic exists at the base restaurant and a new table is opened (table indicator is one):

$$P(z_i = k, u_i = 1|rest) \propto \frac{b_1}{N_j + b_1} \frac{S_{m_{jk}+1}^{n_{jk}+1}}{S_{m_{jk}}^{n_{jk}}} \frac{m_{jk} + 1}{n_{jk} + 1} \frac{M_k^2}{(M_k + 1)(M_\cdot + b_0)} P_{wk} \tag{2.35}$$

The prior term P_{wk} is calculated in the same way as for the standard LDA model in Section 2.1.3:

$$P_{wk} = \frac{N_{wk} + \beta}{\sum_{w'} (N_{w'k} + \beta)} \tag{2.36}$$

In the above equations, b_0 is the hyperparameter for the root DP, b_1 is the hyperparameter for the lower level DP, M_k is the total number of tables for topic k, $M_\cdot$ is the total number of tables, n_{jk} is the number of customers for topic k in restaurant j, N_{wk} is the total number of tokens for word w and topic k, and m_{jk} is the number of tables for topic k in restaurant j.

Alias-sampling for the HDP

As proposed by Li *et al.* [35], the sampling equations may be rewritten as $\frac{\beta}{\sum (N_{w'k} + \beta)} \cdot X + \frac{N_{wk}}{\sum (N_{w'k} + \beta)} \cdot X$, where X stands for the remaining part of the equation. The first part can be stored and sampled from in $O(1)$, since repeated samples from the same distribution are feasible in $O(1)$, adding a Metropolis-Hastings acceptance step to account for the difference with the updated counts. The second part only has to be computed for the topics that occur with word w. Therefore, the sampling complexity is reduced to amortized $O(K_w)$, where K_w is the number of topics that occur with word w.

2.1.9 Different Topic Models: Overview

In this section I want to briefly discuss a number of topic models that are related to the models in this thesis in different ways. They can be differentiated according to five different features as is shown in Table 2.1. Supervised topic models incorporate a target variable in some way, but are not necessarily multi-label. In multi-label topic models each document may exhibit multiple labels. Online topic models can be trained on streaming data. Dependencies between topics or labels are only modeled by some of the methods, whereas in nonparametric topic models, the number of topics is not fixed and they are in some way based on hierarchical Dirichlet processes as introduced in chapter 2.1.8.

There are three relevant multi-label topic models, two of which are discussed in detail in sections 2.3.1 and 2.3.2, Labeled LDA and Dependency-LDA. Wang *et al.* [70] developed a model called CoL (Correlated Labeling Model). It models each label as a distribution over latent topics. A variational learning method is proposed and the results show that this model achieves a slightly better F-measure on the tested datasets than SVMs. Online collapsed variational Bayes [19] was introduced in chapter 2.1.6. Existing supervised models include Supervised LDA [41], DiscLDA [32] and MedLDA [78]. However, these models are single-label classification or regression models and not usable in a multi-label setting.

There exists a number of methods that model dependencies between topics. Among these is the author-topic model [53] which assigns an author and a topic to each word, such that one document is modeled as a mixture of topics and each author is associated with a topic distribution. In the partially labeled topic model by Ramage *et al.* [48] each label is divided into several topics. Another method that models topic dependencies is the Pachinko allocation model (PAM) [37]. It assigns topics on two different hierarchy levels in such a way that each super-level topic is associated with a distribution over sub-level topics and each document has a distribution over both super- and sub-topics. A nonparametric version of this model was proposed by Li [36]. Nonparametric PAM (nPAM) is based on HDPs that model topic correlations. Another model based on nested DP called cHDP was proposed by Shimosaka *et al.* [56]. This model requires that each document is assigned to exactly one super-topic. The proposed learning procedure is based on variational Bayes. The generative process is defined as follows:

$$G_0 \sim DP(b_0, H), Q \sim DP(\alpha, DP(\beta, G_0)), G_d \sim Q$$

As the second equation shows, here one DP is nested into another DP as described by Rodriguez *et al.* [52]. Another model that allows topic sharing was proposed by Salakhutdinov *et al.* [55]. It is a supervised model, however, it does not allow multiple labels per document. Each document

Table 2.1: This table provides an overview over topic models with different properties as compared to the models proposed in this thesis.

	supervised	*multi-label*	*online*	*dependencies*	*nonparametric*
Models in this thesis					
Chapter 6	no	no	yes	no	no
Chapter 3	yes	yes	yes	yes	no
Chapter 4	yes	yes	no	yes	yes
Chapter 5	no	no	yes	no	yes
Other models					
LabeledLDA [47]	yes	yes	(yes)	no	no
DependencyLDA [54]	yes	yes	no	yes	no
Correlated Labeling Model [70]	yes	yes	no	yes	no
Author-topic model [53]	no	no	no	yes	no
Partially Labeled [48]	no	no	no	yes	no
PAM [37]	no	no	no	yes	no
Correlated TM [33]	no	no	no	yes	no
nPAM [36]	no	no	no	yes	yes
Coupled HDP [56]	no	no	no	yes	yes
Salakhutdinov *et al.* [55]	yes	no	no	yes	yes
Supervised LDA [41]	yes	no	no	no	no
DiscLDA [32]	yes	no	no	no	no
MedLDA [78]	yes	no	no	no	no
Online VB [28]	no	no	yes	no	no
Online Collapsed VB [19]	no	no	yes	no	no
On-line LDA [1]	no	no	yes	no	no

Table 2.2: Notation for multi-label classification.

variable	meaning
L	set of all labels
$\|L\|$	number of overall labels
Y_i	label vector for instance with index i
y	a specific label
X	input feature matrix
D	the dataset: $D = (X, Y)$
d	a specific document $d = (X_i, Y_i)$
V	the features/words

is assigned one label using a nested Chinese restaurant distribution. Then the whole document is sampled according to the document's label. The correlated topic model [33] is unsupervised and models correlations between topics using a logistic normal distribution. However, the model is complicated since the normal distribution is not conjugate to the multinomial distribution.

Overall, while it is by no means complete, Table 2.1 shows that the models proposed in this thesis represent unique combinations of different subfields with the exception of the model in Chapter 6 which is closely related to On-line LDA [1] and Online VB [28].

2.2 Multi-label Classification

The last section was about generative topic models. This section introduces multi-label classification in general while the following section brings both sections together by showing how topic models are used for multi-label classification.

Multi-label classification is the problem where each instance in a dataset is assigned one or several labels. It is to be distinguished from multi-class classification which only assigns one out of multiple classes to each instance. The evaluation measures specific to multi-label classification are introduced in Section 2.2.1. Multi-label classification may also be seen as a special case of multi-label ranking where each instance is associated with a ranking over the possible labels as well as a cut-off point which determines at which point to separate the negative from the positive labels [20]. Section 2.2.2 discusses the determination of this cut-off point. The classifiers binary relevance, classifier chains and block classifier chains are introduced in Sections 2.2.3 to 2.2.5.

2.2.1 Evaluation Measures

As suggested by e.g. Tsoumakas *et al.* [63], multi-label evaluation measures can be divided into example-based and label-based measures. Label-based measures are further divided into micro- and macro-averaged measures. Additionally, rank-based measures are computed based on the ranking of labels instead of the binary predictions. In the following, $|D|$ is the number of instances and $|L|$ is the number of labels. tp_i is the number of true positives, fp_i is the number of false positives, tn_i the number of true negatives and fn_i the number of false negatives for instance i.

Example Based Measures

The example-based F-measure, the harmonic mean of recall and precision, is defined as follows:

$$\text{F-Measure}(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{2\text{tp}_i}{2\text{tp}_i + \text{fn}_i + \text{fp}_i}$$

Recall is the fraction of correctly predicted labels among the actually positive labels:

$$\text{Recall}(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{\text{tp}_i}{\text{tp}_i + \text{fn}_i}$$

Precision is the fraction of correctly predicted labels among the labels that are predicted as positive:

$$\text{Precision}(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{\text{tp}_i}{\text{tp}_i + \text{fp}_i}$$

The example-based accuracy is defined as:

$$\text{Accuracy}(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{\text{tp}_i}{\sum \text{tp}_i + \text{fp}_i + \text{fn}_i}$$

The Hamming loss measure is also commonly used in multi-label classification. It represents the fraction of errors averaged over all instances and labels. Hamming loss is optimized by the BR classifier (see Section 2.2.3) as discussed in Cheng *et al.* [17]. In general, the more labels there are, the lower the Hamming loss measure usually gets because most labels are correctly predicted as negative.

$$\text{Hamming Loss}(D) = \frac{1}{|D| \cdot |L|}\sum_{i=1}^{|D|}\text{fp}_i$$

Label-Based Measures

The label-based measures focus on the classifier performance per label. This means that each label is given the same importance irrespective of its frequency. The micro-averaged F-measure is defined as:

$$\text{Micro-F1} = 2\frac{\text{Micro-Precision} \cdot \text{Micro-Recall}}{\text{Micro-Precision} + \text{Micro-Recall}},$$

where Micro-Precision is defined as:

$$\text{Micro-Precision} = \frac{\sum_{y \in L} \text{tp}_y}{\sum_{y \in L}(\text{tp}_y + \text{fp}_y)},$$

where L is the set of all labels, tp_y are true positives for label y and fp_y are the false positives for label y. Accordingly, Micro-Recall is defined as:

$$\text{Micro-Recall} = \frac{\sum_{y \in L} \text{tp}_y}{\sum_{y \in L}\left(\text{tp}_y + \text{fn}_y\right)}$$

The macro-averaged F-measure on the other hand, is based on Precision_y and Recall_y which are precision and recall for a specific label y:

$$\text{Macro-F1} = \frac{1}{|L|}\sum_{y \in L}\frac{\text{Precision}_y \cdot \text{Recall}_y}{\text{Precision}_y + \text{Recall}_y}$$

Rank-Based Measures

The ranking based measures used in this thesis are micro- and macro-averaged AUC [10] and rank one error. Micro-averaged AUC is indicative of the overall rankings of correct and incorrect labels over all instances and labels, whereas macro-averaged AUC evaluates the rankings of all instances per label and averages over the labels. The rank one error is the error rate just for the highest ranking label.

$$\text{Rank One Error}(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}\mathbb{1}[\tau(1,i) \in Y_i],$$

where $\tau(r,i)$ is the label with rank r for instance i and $\mathbb{1}[\tau(1,i) \in Y_i]$ is one if the highest ranking label is in the true labelset of instance i, Y_i, and otherwise zero.

2.2.2 Threshold Selection Methods

To be able to evaluate multi-label classifiers on the binary evaluation measures presented in the previous sections, we first need to determine the positive labels for a certain classifier. Many classifiers however, do not directly

output a partition into positive and negative labels, but instead provide a ranking or a probability distribution over labels. Therefore, there exist many different methods to transform probability distributions into binary predictions. Many different strategies are available and none has so far emerged as a widely accepted standard. This section only discusses exemplary strategies from relevant related work that are commonly used and briefly discusses the main differences.

Overall, thresholding strategies can be divided into two general groups, instance-based and dataset-based strategies, where the instance-based strategies are further divided into threshold-based and cardinality-based methods:

1. Instance-based: The labels $\hat{Y}_i$ that are predicted for one particular instance i are determined.
   (a) Threshold-based: First, a threshold t is calculated, then all labels that have a score above t are set to positive.
   (b) Cardinality-based: The number of labels $|\hat{Y}_i|$ to predict is calculated directly. The $|\hat{Y}_i|$ labels with the highest prediction score are set to positive for that instance.
2. Dataset-based: It is also possible to predict the number of instances N_y that are assigned a certain label y in the whole testset. The label is predicted for all instances with the highest score for this label.

As an example for the instance-based method, Lewis *et al.* [34] use SCut [74] for thresholding. SCut sets a different threshold for each label and sets a label to positive if its score exceeds this threshold. The threshold is found using a validation set by simply using the optimal threshold on the validation set. Nam *et al.* [43] also use an instance-based threshold predictor that is more complex. They consider the list of scores for one instance and form the training set by selecting the threshold that produces the highest F-measure on the training data. They then train a linear regression with l2-regularization to predict the threshold for a particular instance from its features. Many variations of instance-based threshold prediction methods are possible. In particular, it is possible to predict the number of positive labels instead of a concrete threshold, which enables to use the true number of labels given in the training set for training (see Section 3.8.1 for further details of this method).

In contrast to this, an example for the dataset-based strategy is given by Rubin *et al.* [54], who compare results using three different thresholding methods:

1. Proportional: Use the label frequencies in the training set to determine the expected number of instances N_y in the testset with that label. Set

the label to positive for the N_y instances with the highest score for label y.

2. Calibrated: Use the true number of instances with that label in the testset. This thresholding method is only usable for classifier comparison since it uses information from the testset.

3. Break-Even-Point: Choose N_y such that the F-measure for that label is optimized. This thresholding method also uses information from the testset.

To sum up, there are two main groups of thresholding methods, ones that consider the whole testset at once and instance-based methods. Further we can divide thresholding methods into methods that use information from the testset and ones that work without information from the testset. Which method one chooses depends on the intended goal. For comparison of multiple classifiers that produce real-valued outputs, it is fair to use the same number of positive labels for all of them to make results independent from differing thresholding methods. In this case it makes sense to use information from the testset. To realistically evaluate performance or compare two classifiers that produce binary outputs, it is preferable to choose the threshold without information from the testset. The dataset-based approach is well suited to systematic comparisons given a fixed testset of sufficient size. Instance-based thresholding methods on the other hand are more appropriate in streaming settings where different testsets may have high variance and are not well balanced or where it is to be expected that label frequencies differ in the testset.

2.2.3 Binary Relevance

A simple yet efficient approach for multi-label classification is the binary relevance method (BR, [62, 9]). BR does not take dependencies between labels into account, but instead trains one binary classifier for each label separately. The predictions for the different labels are then combined into one multi-label prediction. This is also often referred to as "one-vs-all", "one-against-all" or "one-vs-rest" training [54, 49, 39], a term which is more fitting in the multi-class setting. Therefore, the expression "binary classifier for each label" is preferred in this thesis. More formally, BR learns $|Y|$ binary functions $f_i : X \to \{0, 1\}$ to predict the ith label from the features X, where $|Y|$ is the number of labels.

More commonly, the BR classifier is divided into two steps, first $|Y|$ functions $f_i : X \to \mathbb{R}$ are learned and in a second step these predictions are transformed into binary decisions. Different strategies for the second step were discussed in Section 2.2.2. These methods are applicable to BR as well as other multi-label methods that produce real-valued outputs. In the case

of BR, special care has to be taken in this transformation step. Since the binary classifiers are trained separately for each label, it may be important to make sure that the real-valued prediction results are comparable between labels and that they are normalized to be in the same range.

BR has many advantages, including its computational efficiency (it is linear in the number of labels), but also its resistance to overfitting. Not modeling the label dependencies is in a certain sense also an advantage because the model cannot overfit on label dependencies. It will not erroneously assume that a previously observed label combination is likely to occur again. Another advantage is that BR can easily be extended by adding further classifiers. Since the different classifiers are completely independent from each other, adding or removing labels is not a problem. Also parallelization is straightforward since it is possible to train and test the classifiers for the different labels separately and in parallel.

The performance of BR is often surprisingly good given that it does not take label dependencies into account [51]. A good performance for BR with SVMs has also been found by Lewis *et al.* [34] for labels with high as well as low frequency in the dataset as compared to k-NN and a prototype classifier which computes a prototype vector for each label as a weighted average of positive and negative training instances.

BR is commonly used as a baseline in multi-label classification since it measures the performance possible without considering label dependencies. The performance of BR also depends on the chosen base classifier. Commonly used base classifiers in the case of text classification are decision trees and linear SVMs. Nonlinear SVMs are also possible, but in the case of text classification there was no advantage of nonlinear kernels found in past work [34]. Using SVMs as base classifiers it is also important to optimize the parameters, especially because SVMs may have problems with highly unbalanced training sets where the number of positive and negative examples are different [34]. This is typically the case in multi-label classification, where the number of negative examples is usually much higher than the number of positive examples. However, this problem is usually alleviated by hyperparameter optimization.

2.2.4 Classifier Chains

An extension of BR that includes label dependencies are classifier chains (CC, [49]), where classifiers are part of a chain and each classifier receives the prediction of the previous chain element as an additional attribute.

This might be viewed as inspired by the chain rule of probabilities [17].

$$P(Y) = P(Y_0) \prod_{i=1}^{|Y|} P(Y_i | Y_0 ... Y_{i-1}) \tag{2.37}$$

A CC learns $|Y|$ functions $f_i : X \times \{0,1\}^{i-1} \rightarrow \{0,1\}$, one function for each label in the chain. CC uses a deterministic version of the chain rule where probabilities are either zero or one, but it can be extended to probabilistic classifier chains (PCC, [17]) that use probabilistic base classifiers so the actual probability of a label being zero or one can be propagated along the chain. PCCs have a much better performance but are only applicable with a small number of labels.

CC are the sparsest multi-label method that models dependencies between *all* labels. There are several methods that redundantly model label dependencies using stacking. The first of those, Meta-Stacking by Godbole *et al.* [21], uses a BR classifier that receives as additional input features the predictions for all labels by another BR classifier. This method is adapted by Tsoumakas *et al.* [61], who prune those labels from the meta level that are uncorrelated to the label that is being predicted according to the ϕ coefficient. The rationale is to avoid additional noise that is introduced from completely uncorrelated labels. BR+ is another approach similar to Meta-Stacking (Alvares-Cherman *et al.* [2]) that uses not all the label information at the meta level, but excludes the label being predicted itself. Additionally, it updates the label information at the meta level in a manner that is similar to CC using different chains. The chains are determined by the confidence in the predictions of the respective classifiers or by the frequency of the respective label in the dataset. Finally, stacking is also explored by Montañes *et al.* [42], who analyze the combination of stacking with the use of the actual labels for training vs. the use of the predicted labels for training. They conclude that stacking is especially effective when the predicted labels are reliable.

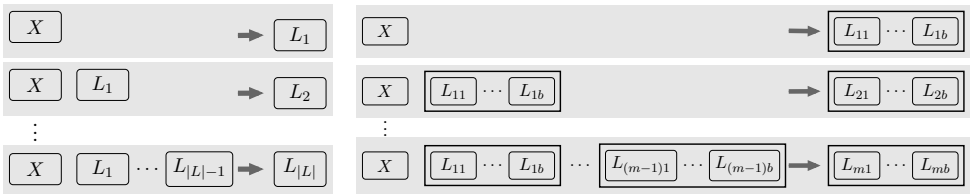
CC is also frequently used in ensembles (ECC). Madjarov *et al.* [40] however, found that ECC cannot improve CC on all performance measures; on some performance measures, CC is even better than ECC. They assume CC to be one of the most stable methods in multi-label classification.

2.2.5 Block Classifier Chains

While BR does not model any dependencies, CC potentially models dependencies among *all* labels. The method proposed in this section, however, (given an appropriate block size) models dependencies between most, though not all, of the labels.

In the original classifier chains, one error at the beginning of the chain possibly has a great influence on the overall result since the error might be propagated along the whole chain. Therefore, if the first classifier predicts the wrong label, and subsequent classifiers depend on this wrong prediction, they are more likely to give wrong predictions as well.

What we need to alleviate this problem is to reduce the probability that errors are propagated along the chain of classifiers. The basic idea of the



(a) CC (b) BCC

Figure 2.4: This figure shows schematic diagrams of CC and BCC. X denotes the input features that are used for the prediction of all labels L_i. In CC, the second classifier predicts L_2 using X and L_1, the third classifier predicts L_3 using X, L_1, and L_2 etc. In BCC, the whole first block is predicted from the input features X by the first b classifiers, the following b labels are predicted using X and the labels from the first block etc. L_{ij} denotes the jth label in the ith block.

presented approach is, instead of using a chain of individual classifiers, to use a chain of *blocks* of classifiers. Hence, Equation 2.37 is approximated by

$$P(Y) \approx \prod_{i=1}^{b-1} P(Y_i) \cdot \prod_{\substack{j=b \\ j+=b}}^{|L|} P(Y_j|Y_1,\ldots,Y_{j-1}),\ldots,P(Y_{j+b-1}|Y_1,\ldots,Y_{j-1}),$$

where b is the block size. Figure 2.4 illustrates the approach: In CC (see Figure 2.4a), classifier one predicts the first label (L_1) using only the features X, classifier two receives the features X plus the output of classifier one etc. In BCC (see Figure 2.4b), all classifiers in the first block predict one label using only the features X, the classifiers in the second block additionally use the outputs of all classifiers in the first block etc.

The idea is that this reduces the probability of propagating errors because information is handed down the chain blockwise, so even if one classifier in a block predicts wrongly, the other classifiers in the block reduce the effects of this error. The algorithms for training and testing a blockwise classifier chain (BCC) are shown in Algorithms 3 and 4. As shown in the pseudo-code, the block sizes are not necessarily equally distributed but for large block sizes, there can be one large block and one smaller one. For example, if there are 10 labels and the block size is 8, there would be one block of size 8 and one of size 2. The first 8 labels would be predicted as in BR, the last 2 would get the predictions of the first 8 labels as an input.

Note that a block size equal to the number of labels corresponds to BR, whereas block size one leads to CC. Between those two extremes, a spectrum of different block sizes is induced.

Algorithm 3 Train Block Classifier Chains

Input: dataset D, block size b, number of labels $|L|$
Output: trained classifier chain

```
for i = 0, ..., |L| − 1 do
    blockIndex ← ⌊i/b⌋
    D' ← {}
    for (x, y) ∈ D do
        x' ← [x_0, ..., x_d, y_0, ..., y_{b·blockIndex−1}]
        D'_j ← D'_j ∪ (x', y_j)
    end for
    h_j : D'_j → {0, 1}
end for
```

Algorithm 4 Test Block Classifier Chains

Input: instance x, classifier chain h
Output: prediction $\hat{y}$

```
y ← [ŷ_0, ..., ŷ_{|L|−1}]
for j = 0, ..., |L| − 1 do
    blockIndex ← ⌊j/b⌋
    x' ← [x_0, ..., x_d, ŷ_0, ..., ŷ_{b·blockIndex−1}]
    ŷ_j ← h_j(x')
end for
```

Second Generalization

Another way to generalize CC is to introduce a different kind of blocks. In this alternative, label dependencies are not modeled between blocks but within blocks. Instead of having one big chain, the chain is divided into several smaller subchains that are independent of each other. In this case, error propagation is also reduced since in shorter chains, the error cannot be propagated as far. However, this also leads to far fewer label dependencies being modeled than in BCC. Every label that is not in the same block as another label cannot make use of the respective label information, whereas in the first approach every label can make use of all the information from previous labels *except* the ones in its own block. This approach is called BCC-IO (Inside Out). The algorithms for BCC can also be applied to BCC-IO, the only difference being a different interpretation of the blocks. Note that for BCC, block size one corresponds to CC, whereas for BCC-IO the largest possible block size ($b = |L|$) corresponds to CC.

Equation 2.37 is in this case approximated by:

Algorithm 5 Find Ensemble of Chains

```
c ← random chain
E ← {c}
p ← true
n ← 0
while |E| < s do
    n ← n + 1
    p ← false
    c ← random chain
    for e ∈ E do
        σ ← Sim(c, e)
        if σ > θ then
            p ← true
        end if
    end for
    if not p then
        E ← E ∪ c
    else if n > m then
        θ ← θ + 0.05
    end if
end while
```

$$P(Y) \approx \prod_{\substack{i=1 \\ i+=b}}^{|L|} \left\{ P(Y_i) \prod_{j=i+1}^{i+b} P(Y_j | Y_i, \ldots, Y_{j-1}) \right\} \tag{2.38}$$

Ensembles

Just as with CC and PCC, BCC can also be used in an ensemble. The difference is that for BCC there are (depending on the block size) many equivalent chains since the order within a block does not make a difference. If we have two blocks of 3 classifiers each, there are not $6! = 720$ different chains but only $\binom{6}{3} = \frac{6!}{3!(6-3)!} = 20$ different possibilities. To ensure that the ensemble does not consist of equivalent chains, a similarity measure is introduced. Let l_i be the ith label, $prev(l, c)$ be the set of labels that are used to predict label l in a chain c, then the similarity of two chains c_1 and c_2 is defined as:

$$Sim(c_1, c_2) = \frac{\sum_{i=0}^{|L|-1} |prev(y_i, c_1) \cap prev(y_i, c_2)|}{\sum_{i=0}^{|L|-1} i} \tag{2.39}$$

When the ensemble is built, random chains are produced, and discarded if they are too similar according to a certain threshold. The threshold is in-

Table 2.3: number of labels, attributes, and instances for the used datasets

number of	*medical*	*birds*	*enron*	*yeast*	*genbase*	*CAL500*
labels	45	19	53	14	27	174
attributes	1449	260	1001	103	1186	68
instances	978	645	1702	2417	662	502

creased if the algorithm finds no chain with less similarity (see Algorithm 5, line 18). A threshold close to one leads to an ensemble of non-equivalent but possibly similar chains, whereas a threshold close to zero leads to an ensemble of dissimilar chains. More involved similarity measures for classifier chains are conceivable.

Experiments

Datasets

Six well-established publicly available multi-label datasets were used[3] that are listed in Table 2.3 together with their respective number of labels, attributes and instances. All those data sets are typical multi-label data sets in that they have few instances and many labels. If a dataset has many instances, multi-label methods are generally less effective because the classifier has enough information to infer the labels from the features, while data sets with few labels are less interesting for the proposed block method since there are limited possibilities of divisions of the label set for small numbers of labels. There is no known influence of the number of attributes on multi-label learning in general, however, it influences the performance of the underlying base classifier. The `medical` dataset is a text dataset where ICD-9-CM codes are assigned to radiology reports. `birds` is a dataset that is used to identify bird sounds in a noisy environment with background noises and multiple birds vocalizing simultaneously. The *enron* dataset includes 1702 hand labeled email messages. *yeast* associates gene functions with protein types. In `CAL500`, music is categorized according to different tags whereas in `genbase` proteins are classified into different structural families.

Results

The implementation and evaluation of all classifiers is based on the MULAN framework (version 1.4 [65]). To compare CC and BCC, both were run for two days on an 8-core 3.50 GHz Intel Core i7. J48, a standard decision tree method, was used as a base classifier. For each run a random 2:1 split of the data and different random chain orderings were used. The block size

[3] http://mulan.sourceforge.net/datasets.html

Table 2.4: The medians for CC and BCC on different multi-label datasets; the average difference between CC and BCC on the same chain: significant differences (0.5% significance level, Wilcoxon signed-rank test) are shown in bold, significant improvements of BCC in italics; the best block size according to the block size with the highest average value of BCC

	medical	birds	enron	yeast	genbase	CAL500
			Micro-averaged AUC			
Med. CC	0.9142	0.7258	0.8025	0.6668	0.9933	0.6545
Med. BCC	0.9178	0.7283	0.8018	0.6732	0.9933	0.6675
Av. Diff.	***0.0038*** ±0.0091	***0.0026*** ±0.0168	0.0003 ±0.0118	***0.0061*** ±0.0176	0.0000 ±0.0002	***0.0131*** ±0.0208
best b.s.	7	13	7	14	5	12
			Example-based F-Measure			
Med. CC	0.7779	0.5936	0.5123	0.5248	0.9879	0.3481
Med. BCC	0.7725	0.5945	0.5057	0.5411	0.9879	0.3452
Av. Diff.	**-0.0053** ±0.0117	***0.0010*** ±0.0116	**-0.0061** ±0.0195	***0.0136*** ±0.0256	0.0000 ±0.0004	***-0.003*** ±0.0166
best b.s.	3	13	12	14	7	6
			Example-based Recall			
Med. CC	0.7958	0.5916	0.4931	0.5496	0.9882	0.3223
Med. BCC	0.7928	0.5932	0.4846	0.5662	0.9882	0.3031
Av. Diff.	**-0.0028** ±0.0112	***0.0015*** ±0.0118	**-0.0082** ±0.0218	***0.0142*** ±0.0314	0.0000 ±0.0004	**-0.0194** ±0.0293
best b.s.	3	13	3	10	7	6
			Example-based Precision			
Med. CC	0.7858	0.6226	0.591	0.555	0.9923	0.4023
Med. BCC	0.7793	0.6231	0.5934	0.5743	0.9923	0.4261
Av. Diff.	**-0.0064** ±0.0125	***0.0006*** ±0.0130	***0.0015*** ±0.0190	***0.0185*** ±0.0291	0.0000 ±0.0004	***0.0230*** ±0.0301
best b.s.	5	3	12	14	7	12
			Example-based Accuracy			
Med. CC	0.7495	0.5626	0.4073	0.4205	0.9827	0.2182
Med. BCC	0.7426	0.563	0.3991	0.4249	0.9827	0.2137
Av. Diff.	**-0.0067** ±0.0127	***0.0004*** ±0.0112	**-0.0085** ±0.019	***0.0032*** ±0.019	0.0000 ±0.0004	**-0.0046** ±0.0131
best b.s.	5	13	12	13	7	6
			Hamming Loss			
Med. CC	0.0107	0.0519	0.0546	0.2713	0.0013	0.1755
Med. BCC	0.0108	0.0519	0.0539	0.2634	0.0013	0.167
Av. Diff.	**0.0001** ±0.0004	***0.0000*** ±0.0016	***-0.0006*** ±0.0014	***-0.0074*** ±0.0132	-0.0000 ±0.0000	***-0.0081*** ±0.0097
best b.s.	5	8	9	14	11	12

Table 2.5: ECC compared to EBCC; an average over 100 runs was computed. Significant results (0.5% significance level, Wilcoxon signed-rank test) are shown in bold. The ensemble size is 20, the block size is chosen randomly for each chain.

	medical	birds	enron	yeast	genbase	CAL500
			Micro-averaged AUC			
ECC	0.964	0.862	0.912	0.836	0.994	**0.799**
EBCC	0.964	0.864	0.912	0.836	0.994	0.797
			Example-based F-Measure			
ECC	0.776	0.606	**0.559**	0.6043	0.985	0.338
EBCC	0.773	0.608	0.5515	**0.607**	0.994	**0.343**
			Example-based Recall			
ECC	0.790	0.581	**0.520**	0.572	0.984	0.225
EBCC	0.792	0.583	0.512	**0.578**	0.984	**0.254**
			Example-based Precision			
ECC	0.787	0.649	0.671	**0.713**	0.990	**0.583**
EBCC	0.785	0.653	0.670	0.709	0.990	0.563
			Example-based Accuracy			
ECC	0.750	0.582	0.449	0.499	0.980	0.208
EBCC	0.745	0.584	0.441	0.500	0.980	0.212
			Hamming Loss			
ECC	0.011	0.043	0.048	0.198	0.002	**0.141**
EBCC	0.011	0.043	**0.047**	0.199	0.002	0.143

was also chosen randomly each time. However, the same chain was used for CC and BCC in each run. This way the effect that the introduction of blocks had on each chain could be analyzed. The average difference between CC and BCC was computed for different measures (see Table 2.4). If the average difference is positive, this means that BCC is better than CC on average. Only for Hamming loss, negative values indicate that BCC was better than CC on average.

As is shown in Table 2.4, BCC is better than CC on AUC, precision and Hamming loss. Across these three measures BCC improves over CC 11 out of 18 times, whereas CC improves over BCC only 3 out of 18 times. Additionally, the median for all measures and datasets is provided, which is the value below which 50% of all data points fall. For AUC, F-measure, recall, precision and accuracy, higher values indicate a better performance, whereas for Hamming loss, lower values point to an improvement. These values mostly confirm the improvement shown by the average difference.

The performance of BCC is especially good for the AUC measure which is a ranking measure in that it measures how well instances are ranked

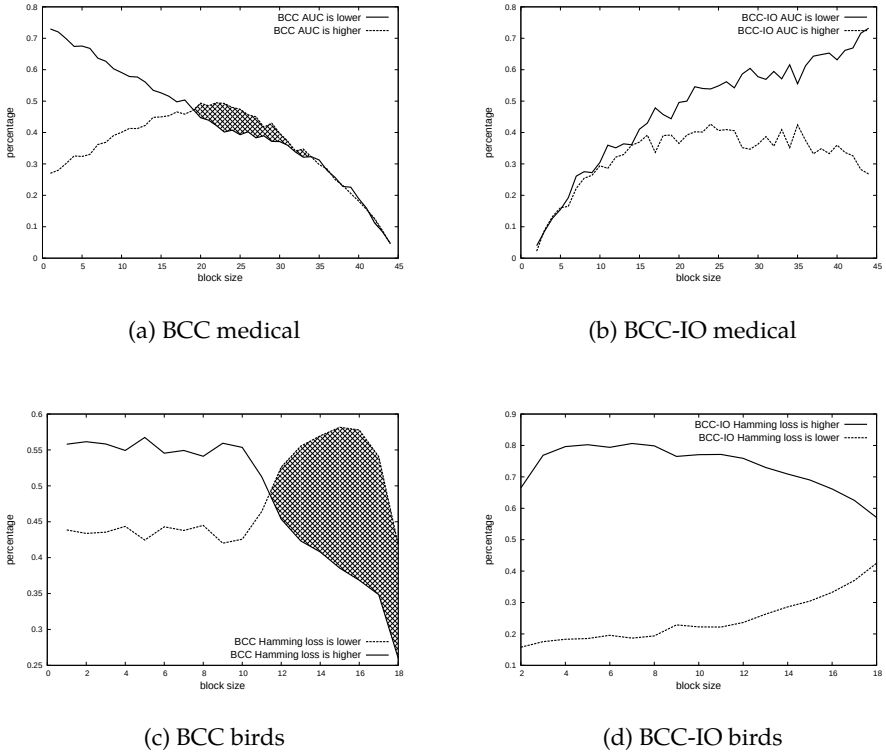
based on prediction confidences. The results of the other measures however depend on a cut-off that has to be set in order to classify an instance as positive or negative for each label. Therefore, results for AUC are especially meaningful, since they do not depend on a possibly suboptimal cut-off.

Further, it was investigated whether there is one block size that leads to overall good performance values across all measures. To do that, the best block size was computed for each measure according to the highest average performance value, as shown in Table 2.4. However, the optimal block size differs widely across different data sets and performance measures, with no clearly discernible pattern.

While Table 2.4 provides insight into the general difference between CC and BCC, the spectrum between BR and CC was further analyzed by differentiating between different block sizes. The result is illustrated in Figure 2.5, where the percentage of chains being better or worse than BR for increasing block sizes was plotted. The figure just provides an example since the plots varied a lot across different data sets and measures. On the left side of the spectrum we see the relationship between CC and BR, on the right side of the spectrum lies BR, and in between there are the results for BCC as compared to BR. Figure 2.5a shows that for CC the percentage of chains that are worse than BR is higher than that of chains that are better than BR. The percentage of better chains increases with bigger block sizes and reaches a peak at block sizes of about 20–25. After that, good and bad chains are more or less balanced and the difference to BR decreases. In Figure 2.5c Hamming loss is considered for the `birds` dataset. Here we see that the percentage of good chains for BR is higher than that for CC, at a block size between 13 and 18 however, BCC has an advantage over BR as can be inferred from the steep increase of the dotted line in the plot. Consequently, the spectrum is apparently more than just a smooth transition between CC and BR, but there are block sizes for which BCC is better than both, CC and BR, although, as is evident in Table 2.4, the differences are small.

According to Figure 2.5a the best block size for the optimization of AUC seems to be around 20–25, if the average AUC for each block size is computed however, 7 appears to be the best block size (Table 2.4). This is to be explained firstly with Figure 2.5a comparing BCC to BR whereas in Table 2.4 BCC is evaluated on its own. Secondly, in Figure 2.5a the percentages of chains being better or worse than BR are plotted, not the absolute improvements. So, while the biggest improvement over BR and CC in terms of the percentage of better chains is achieved at a block size of 25, the best performance on average is achieved at a block size of 7.

To better understand the reasons for the observed improvements of BCC, the distribution of the performance for different measures was plotted. The results for the `CAL500` dataset are depicted in Figure 2.6. Figure 2.6b shows the distribution of F-measure for the `CAL500` dataset. The distribution for BCC is shifted to the left so that there are less results



(a) BCC medical

(b) BCC-IO medical

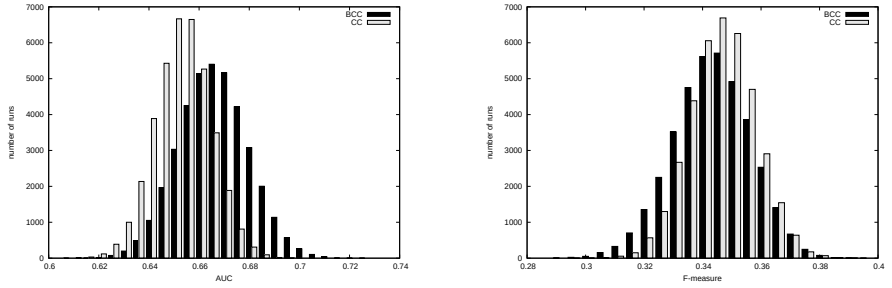
(c) BCC birds

(d) BCC-IO birds

Figure 2.5: The percentage of chains being better or worse than BR for different block sizes on the `medical` and `birds` datasets for BCC and BCC-IO. The performance measures were AUC and Hamming loss, respectively. The shaded area indicates the presence of the block effect.

with high accuracy for BCC and more results with a lower accuracy. For AUC on the other hand, the shift in distribution in favor of BCC is evident. As Figure 2.6a demonstrates, BCC not only has more results with higher performance, but its maximum performance values are also increased. This means that well performing chains in BCC have the potential of outperforming even the best chains of CC.

This result is uilized in the direct comparison of CC and BCC. Table 2.6 presents the average results over 15 runs. In each run, the best performing of 10 chains is selected via an inner 10-fold cross validation. The chains for CC are generated randomly whereas for BCC the blocksize is varied in steps of 10% of the overall number of labels. As the results show, there is an improvement of BCC over CC on precision and Hamming loss on 2 out of 6 datasets. The AUC of `CAL500` is also improved for BCC as was to be expected from Figure 2.6. While generally the difference is not large and for most cases not statistically significant, BCC improves over CC in some



(a) CAL500 AUC

(b) CAL500 F-measure

Figure 2.6: The distribution of performance on the CAL500 dataset for AUC and F-measure over randomly sampled block sizes. The performance of BCC is better than that of CC for AUC (2.6a), but worse for F-measure (2.6b).

cases. CC on the other hand, only improves on recall on the CAL500 dataset, where in turn it is worse on precision. It may be concluded that modeling too many label dependencies leads to lower precision (and Hamming loss) in some cases.

The results for the ensemble methods ECC and EBCC are shown in Table 2.5. Since the improvement of the non-ensemble method was only marginal, no larger improvement for the ensemble method was expected. This was confirmed in the experiments. 100 runs of ECC and EBCC were done with an ensemble size of 20. In EBCC the block size was chosen randomly for each chain. The similarity threshold was set to 0.99 to exclude equivalent chains but enforce maximal diversity, since this was found to be detrimental to the performance. This might be due to the small number of well-performing chains: If one well-performing chain is selected, maximal diversity would drive EBCC to select more inaccurate chains. One interesting anomaly was found for the CAL500 dataset, where BCC improves over CC on AUC and Hamming loss, but the performance of EBCC is worse than that of ECC on those measures. Generally, the performance depends on the dataset, and the results do not indicate a reliable improvement of either of the two methods.

Regarding the alternative generalization BCC-IO, the *block effect* that was described above, could not be observed. As Figure 2.5 shows, for BCC there is clearly an area in the spectrum where BCC is better than both, BR and CC, which is not the case for BCC-IO (Figures 2.5b and 2.5d). Here, the percentage of chains being worse than BR is dominant throughout the whole block size parameter space. Therefore, the generalization BCC seems to be preferable over the generalization BCC-IO.

Table 2.6: CC compared to BCC; the best of 10 chains is determined using 10-fold cross validation; an average over 15 runs was computed. Significant results (0.5% significance level, Wilcoxon signed-rank test) are shown in bold.

	medical	birds	enron	yeast	genbase	CAL500
			Micro-averaged AUC			
CC	0.921	0.718	0.811	0.672	0.993	0.664
BCC	0.919	0.726	0.810	0.679	0.993	**0.677**
			Example-based F-Measure			
CC	0.781	0.593	0.511	0.527	0.986	0.348
BCC	0.777	0.593	0.515	**0.545**	0.986	0.343
			Example-based Recall			
CC	0.801	0.589	0.496	0.555	0.985	**0.321**
BCC	0.800	0.590	0.495	0.569	0.985	0.299
			Example-based Precision			
CC	0.786	0.622	0.581	0.553	0.991	0.406
BCC	0.780	0.623	0.596	**0.582**	0.991	**0.424**
			Example-based Accuracy			
CC	0.754	0.563	0.409	0.421	0.981	0.218
BCC	0.749	0.562	0.410	0.428	0.981	0.211
			Hamming Loss			
CC	0.010	0.053	0.056	0.273	0.002	0.173
BCC	0.010	0.053	0.055	**0.260**	0.002	**0.166**

Conclusion

To conclude, a new generalization of BR and CC called blockwise CC or BCC, was introduced. A comparison between CC and BCC on the same chains showed that BCC improves over CC on AUC, precision and Hamming loss. This can be attributed to the so-called *block effect* that prevents the error from propagating along the chain to a certain extent. By looking at the spectrum between BR and CC, certain areas in the parameter space were identified that indeed lead to an improved performance over both BR and CC. A comparison to another generalization of the two methods showed that the *block effect* is not present in that case. To sum up, it was shown that it is not always advantageous to model all potential label dependencies or even model them redundantly as in the discussed stacking approaches (see Section 2.2.4). Modeling a portion of potential label dependencies is, at least sometimes, the better choice if one wants a stable classifier that does not disregard label dependencies altogether.

2.2.6 Extreme Multi-Label Classifiers

While BR, CC and BCC can be considered to be efficient and scalable classifiers, they still require to learn one classifier per label, which can lead to very large models. Most multi-label algorithms in the literature are even more inefficient, many have a complexity that is quadratic in the number of labels (see e.g. Zhang and Zhou [77], Wicker *et al.* [71]), and therefore are not applicable in the kind of large-scale setting considered here. Recently, there have also been some approaches using deep learning and neural networks, but none of them scales well with very large label numbers [72, 22, 76, 43].

In light of these issues, a new line of work on so-called extreme multi-label classification has developed in recent years. This work is concerned with datasets having several hundred thousand or even millions of labels and features. For example, FastXML (Fast eXtreme Multi Label) by Prabhu and Varma [46], an ensemble of decision trees, has prediction cost that is logarithmic in the number of labels. Another method, PD Sparse (Primal Dual Sparse approach to extreme multi-class and multi-label classification) by Yen *et al.* [75] has a training runtime sublinear in the number of classes or labels.

The methods introduced in this thesis are not extreme multi-label classifiers since they are linear in the number of labels. However, they are generally more efficient than BR since they only learn one model for all labels instead of separate models for each label.

2.3 Multi-Label Topic Models

2.3.1 Labeled LDA

Labeled LDA (LLDA) was introduced by Ramage *et al.* [47]. In this work, the collapsed Gibbs sampling topic model by Griffiths and Steyvers [23] (see Section 2.1.3) is extended by introducing document labels Λ_d that are generated from a Bernoulli distribution for each topic k. The set of document labels is given as $Y_d = \{k|\Lambda_{dk} = 1\}$. They furthermore define a document-specific label projection matrix $L^{(d)}$ of size $|Y_d| \times K$, where $|Y_d|$ is the number of labels per document and K is the number of topics. In this matrix, the ith row has an entry of 1 in column j if the ith label of document d is equal to topic j. Otherwise the entry is zero. The Dirichlet parameter vector α of the Gibbs sampling topic model is then projected to a lower dimensional vector $\alpha_d = L^{(d)} \times \alpha$. The dimensions of this vector correspond to the document labels.

The model is defined in a slightly different way in Rubin *et al.* [54] although in practice the training procedure is the same. Here, the model is called Flat-LDA and does not include a generative procedure for the set of

labels via Bernoulli variables. During training of both models, the Bernoulli variables do not play any role. In practice, both models correspond to LDA with a restriction of sampling only from the document labels during training. If each document is only assigned a single label, the model reduces to Naive Bayes [47].

Ramage *et al.* and Rubin *et al.* proposed collapsed Gibbs sampling as a training algorithm, however, this is only one potential variant of the model. Since the idea of Flat-LDA is simply to replace unsupervised topics with labels, the same idea can be applied to topic models with other training methods as well:

1. Variational inference can be used as an alternative training algorithm (see e.g. Papanikolaou *et al.* [44]). The disadvantage is that the algorithm is biased. Also it is more difficult to implement sparse updates. On the positive side, variational inference makes it possible to train the model online.

2. Nonparametric topic models are another alternative for supervised training (see Section 5.7.3). These hierarchical Dirichlet process topic models provide an asymmetric topic/label prior. This model may also be trained using different algorithms:

   (a) Variational Bayes

   (b) Gibbs sampling

   (c) Hybrid Variational-Gibbs (see Chapter 5)

3. More complex hierarchical topic models may be used. In particular, I discuss

   (a) the author-topic model (Section 3.5): Gibbs sampling is used for training.

   (b) Dependency-LDA (Section 2.3.2): Gibbs sampling is used for training.

   (c) Fast-Dependency-LDA (Chapter 3): This model can be trained with Gibbs sampling or variational inference.

   (d) Stacked HDP (Chapter 4): Gibbs sampling is used for training.

This summary shows that the simple idea of supervised topic models has many variants depending on the one hand on the exact model that is used (parametric or nonparametric, simple flat or hierarchical) and depending on the other hand on the training algorithm for the chosen model.

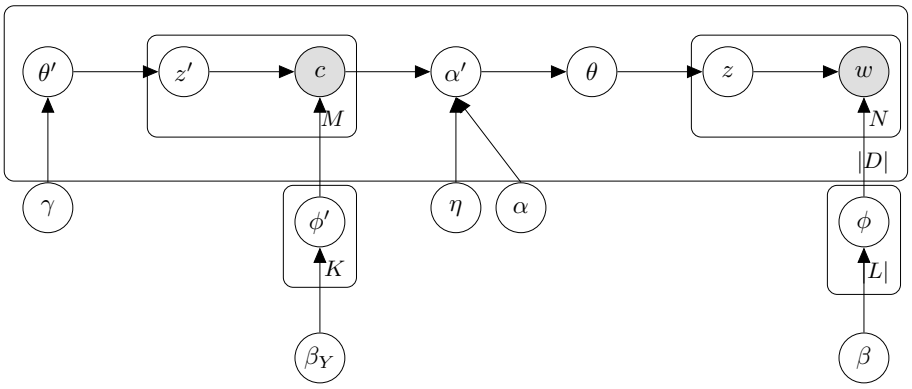


Figure 2.7: Dep.-LLDA from Rubin *et al.* [54]

2.3.2 Dependency-LDA

In this section Dependency-LDA (Dep.-LLDA) is introduced, a topic model for multi-label classification introduced by Rubin *et al.* [54]. The idea of Dep.-LLDA is to learn a model with two types of latent variables: the labels and the topics. The labels are associated with distributions over words, while the topics are associated with distributions over labels. The topics capture dependencies between the labels since the frequent labels in one topic are labels that tend to co-occur in the training data.

The graphical model of Dep.-LLDA is shown in Figure 2.7 and the generative process is given as follows:

1. For each topic $k \in 1, \ldots, K$ sample a distribution over labels $\phi'_k \sim Dirichlet(\beta_Y)$

2. For each label $y \in L$ sample a distribution over words $\phi_y \sim Dirichlet(\beta)$

3. For each document $d \in D$:

   (a) Sample a distribution over topics $\theta' \sim Dirichlet(\gamma)$

   (b) For each label token in d:

      i. Sample a topic $z' \sim Multinomial(\theta')$

      ii. Sample a label $c \sim Multinomial(\phi'_{z'})$

   (c) Sample a distribution $\theta \sim Dirichlet(\alpha')$

   (d) For each word token in d:

      i. Sample a label $z \sim Multinomial(\theta)$

      ii. Sample a word $w \sim Multinomial(\phi_z)$

Table 2.7: Notation for Dep.-LLDA Gibbs sampling models

V	words
K	number of topics
L	labels
D	documents
N_d	number of words in document d
i,j,y,k	indices over word tokens, documents, labels and topics resp.
z,c	label indicator variables
z'	topic indicator variables
α,β,β_Y,γ	hyperparameters (see generative processes)
ϕ, ϕ'	word-label distribution, label-topic distribution
θ,θ'	document-label, document-topic distribution
n_{-wy}	count for word w with label y excluding the current token
$n_{-\cdot y}$	count for label y excluding the current token
n_{-dy}	count for label y in document d excluding the current token
n_{-yk}	count for label y with topic k excluding the current token
$n_{-\cdot k}$	count for topic k excluding the current token
n_{-dk}	count for topic k in document d excluding the current token

The Gibbs sampling equations for the labels z and the topics z' are given by:

$$P(z = y|w, z_{-i}, z'_{-i}) \propto \frac{n_{-wy} + \beta}{n_{-\cdot y} + |W|\beta}(n_{-dy} + \alpha') \tag{2.40}$$

$$P(z' = k|c = y, c_{-i}, z'_{-i}) \propto \frac{n_{-yk} + \beta_Y}{n_{-\cdot k} + |L|\beta_Y}(n_{-dk} + \gamma), \tag{2.41}$$

where n_{-wy} is the number of times word w_i occurs with label y. $n_{-\cdot y}$ is the number of times label y occurs overall, n_{-dy} is the number of times label y occurs in the current document, n_{-yk} is the number of times label y occurs with topic k, $n_{-\cdot k}$ is the number of times topic k occurs overall and n_{-dk} is the number of times topic k occurs in document d. The subscript $-$ indicates that the current token is excluded from the count.

The connection between the labels and the topics is made through the prior α'. To calculate α', Rubin *et al.* propose to make use of the label tokens c. According to these M_d label tokens, α' for document d is calculated as follows:

$$\alpha' = [\eta\frac{n_{d1}}{M_d} + \alpha, \eta\frac{n_{d2}}{M_d} + \alpha, ..., \eta\frac{n_{d|L|}}{M_d} + \alpha], \tag{2.42}$$

where n_{dy} is set to one during training, and to the number of times a particular label was sampled during testing, and η and α are parameters.

During testing however, instead of taking M samples and calculating α' as described above, a so-called "fast" inference method is used. This means the sampled z variables are used directly instead of c, and α' is calculated as follows:

$$\alpha' = \eta \hat{\theta}' \hat{\phi}' + \alpha, \tag{2.43}$$

where $\hat{\phi}$ and $\hat{\theta}$ are the current estimates of ϕ and θ. During training, since the labels of each document are given, ϕ and ϕ' are conditionally independent which allows separate training of both parts of the topic model. Finally, they apply a heuristic to scale α' according to the document length during testing.

Overall, Dep.-LLDA is an effective and efficient method for multi-label classification. The next two chapters will further analyze and improve this method.

Chapter 3

Online Multi-label Topic Model

3.1 Online Topic Modeling for Multi-Label Classification

As discussed in Section 2.2, an important feature of multi-label datasets is that labels often exhibit dependencies. Assume for example that a text has two labels, "Language" and "Programming". This could mean the text is about programming languages, showing that there is some overlap between the two labels. Therefore there is a certain dependency between them, one that is perhaps not exhibited by labels such as "Dog" and "Matrices". A text about dogs is unlikely to also be about matrices, whereas a text about languages has a certain probability to also be about programming. Therefore, modeling dependencies has the potential to improve the accuracy of multi-label classifiers. The potential improvement is expected to be higher when the amount of training data for one of the labels is limited. For example if there are only few documents on programming, but a lot of documents about languages, the dependency between those labels could enhance the prediction of new documents about programming. One could understand this as providing additional positive examples that describe a certain aspect of documents about programming. A model that does not use dependencies only has the few programming documents as positive examples. Experimental evidence by Read *et al.* [49] suggests that in cases where large numbers of training examples are available, the modeling of label dependencies may be unnecessary or in some cases even detrimental. In the streaming setting there are large amounts of training data, however, in real world data sets most labels are rare and only few labels are very frequent. This means that despite the large amount of data, the modeling of label dependencies might be extremely important if the number of labels is high enough.

To be able to model label dependencies, the model proposed in this

chapter includes unsupervised topics in addition to the supervised labels. In general, there are two approaches to achieve this. First, each label might be associated with a distribution over topics, thereby modeling different aspects of labels. Second, each topic might have a distribution over labels, thereby grouping similar labels. As an example, consider two labels "Python" and "C++" grouped into one topic about "programming languages". This chapter, in accordance with previous work [54], follows the second approach.

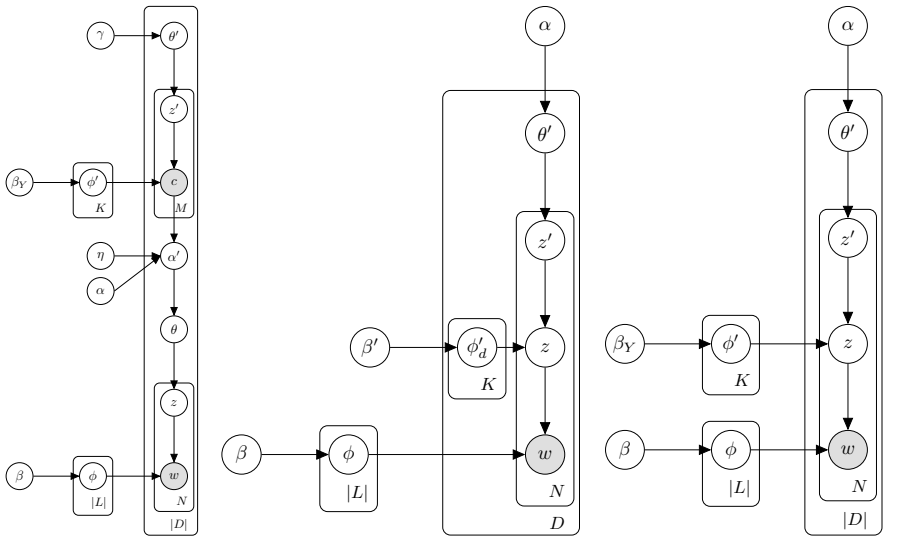
The method proposed in this chapter is based on topic modeling using latent Dirichlet allocation (LDA, Section 2.1.2). The underlying multi-label approach was originally introduced as LabeledLDA (LLDA, Section 2.3.1) [47] and not only has a competitive performance, but also the advantage of interpretability of the resulting model. Word clouds for each label can be extracted and human feedback may be incorporated in the priors of this generative model. The method was extended to Dependency-LDA (Dep.-LLDA, see Section 2.3.2) by Rubin *et al.* [54], who incorporated the modeling of label dependencies to develop a method that is competitive with state-of-the-art discriminative SVM-based methods. However, their method is a complex model with many parameters and it is not usable in an online setting.

In this chapter a simplified version of Dependency-LDA is proposed to yield a practically more appealing and well performing model that may be used in the batch as well as the online setting. To make this simplification possible without sacrificing classification accuracy, the proposed method uses a greedy training strategy inspired by recent advances in neural networks training to train the model one layer at a time. Similarly to boosting, the label level is learned on the input directly while the next layer receives the output of the label level as input. Thereby, two models are essentially stacked on top of each other during training and combined into one big model during inference. This approach can be justified by considering that the first level learns the connection between labels and words where the labels are given during training (see Figures 3.6 and 3.8), whereas the higher level learns more abstract label dependencies (see Figures 3.7 and 3.9 for an example of learned label dependencies).

An example for an application area of online multi-label classification based on topic modeling is the monitoring of news as they appear every day. The proposed topic modeling method allows the classification of newly arriving documents into predetermined categories, but also to extract word clouds from each topic and thus identify terms or aspects that become relevant over time. Furthermore, the model can be updated with new training data at any time. The feasibility of such an approach has already been shown for unsupervised topic modeling [1], however, this work is the first online supervised multi-label topic modeling approach that models label dependencies.

The main contributions of this chapter are as follows:

1. A new LDA topic model that models label dependencies and can be used on streaming data is introduced.
2. Gibbs sampling equations for the batch version are provided as well as variational Bayes update equations for the online version of the method.
3. It is shown that this model is competitive with the previously existing models in modeling label dependencies in the batch as well as the online setting on a range of publicly available large-scale multi-label text datasets with thousands of labels.



(a) Dep.-LLDA from Rubin *et al.* [54]

(b) PAM [37]

(c) The proposed method, Fast-Dep.-LDA

Figure 3.1: The graphical models of the original Dep.-LLDA by Rubin *et al.* [54], PAM [37], and Fast-Dep.-LDA.

3.2 Fast-Dep.-LLDA

The proposed Fast-Dep.-LLDA and Dep.-LLDA have strong similarities. Dep.-LLDA and the corresponding notation is described in Section 2.3.2. The main difference is the omission of θ and α' in Fast-Dep.-LLDA. Both models learn the label dependencies through the label-topic distributions ϕ'. Dep.-LLDA passes the dependency information down via the label-prior α' and the label distribution θ. Fast-Dep.-LLDA, however, takes the

Table 3.1: The generative process of Fast-Dep.-LLDA

For each topic $k \in 1, \ldots, K$
- sample a distribution over labels $\phi'_k \sim Dirichlet(\beta_Y)$
For each label $y \in L$
- sample a distribution over words $\phi_y \sim Dirichlet(\beta)$
For each document $d \in D$:
1. Sample a distribution over topics $\theta' \sim Dirichlet(\alpha)$
2. For each token in d:
2.1 Sample a topic $z' \sim Multinomial(\theta')$
2.2 Sample a label $z \sim Multinomial(\phi'_{z'})$
2.3 Sample a word $w \sim Multinomial(\phi_z)$

more direct approach of generating the labels from ϕ' directly instead of using the intermediary distribution θ (see the graphical models in Figures 3.1a and 3.1c). Thereby Fast-Dep.-LLDA avoids a couple of heuristics that are employed by Dep.-LLDA:

1. Dep.-LLDA does not perform the calculation of the parameter α' according to the proposed model, but rather a fast inference method is used that was empirically found to be faster and was leading to more accurate results.

2. The calculation of the parameter α' itself involves two parameters η and γ that are determined heuristically by the authors. It is unclear how these parameters could be estimated from the data, except by doing expensive grid search.

3. During evaluation the parameter α' is scaled according to the document length.

4. During evaluation, the label tokens c and in particular the number of labels are unknown. To circumvent this problem, the authors replace the label tokens c by the label indicator variables z during testing, thereby assuming that the number of labels is equal to the document length.

The full generative process of Fast-Dep.-LLDA is given in Table 3.1. Each document is only associated with one document-specific distribution θ' over the topics. In comparison, Dependency-LDA has two document-specific distributions, θ and θ', where θ is a label distribution. The label distribution θ is implicitly contained in Fast-Dep.-LLDA and can be obtained by multiplying the document-specific topic distributions θ' with the global topic-label distributions ϕ'.

From the graphical model and the generative process, the joint distribution of Fast-Dep.-LLDA is given by:

$$P(w, z, z') = P(w|z, \phi)P(z|z', \phi')P(z'|\theta') \tag{3.1}$$

To obtain a collapsed Gibbs sampler, ϕ, ϕ', and θ' have to be integrated out from the three conditional probabilities respectively. The integrals can be performed separately as in Griffiths and Steyvers [23] (see Section 2.1.3), resulting in the following conditional distribution for the latent variables z and z':

$$P(z = y, z' = k|w, z_{-i}, z'_{-i}) \propto \frac{n_{-wy} + \beta}{n_{-\cdot y} + |V|\beta} \frac{n_{-yk} + \beta_Y}{n_{-\cdot k} + |L|\beta_Y}(n_{-dk} + \alpha) \tag{3.2}$$

This sampling equation results in a blocked Gibbs sampler that samples two variables at a time instead of just one: each word is assigned a topic and a label. For the new model the use of a basic Gibbs sampler is proposed that only samples one variable at a time instead. This may have the disadvantage of making successive samples more dependent [6], but the advantages outweigh this potential disadvantage. Mainly, the sampling complexity is reduced from $O(K \cdot |L|)$ to $O(K + |L|)$. Also, it makes more sense in this case to view each document as a whole entity and sample the variables in a top down manner (see Algorithm 3.4). First, more abstract topics are sampled (line 3–5, Equation 3.4), representing the label dependencies of the document, and second, the labels are sampled based on these document-specific label dependencies (line 6–8, Equation 3.3).

The corresponding sampling equations for the alternate sampling of labels and topics are given as follows. Given z', the equation for sampling z is:

$$P(z = y|w, z' = k, z_{-i}, z'_{-i}) \propto \frac{n_{-wy} + \beta}{n_{-\cdot y} + |V|\beta}(n_{-yk} + \beta_Y) \tag{3.3}$$

The sampling equation for z' follows from $P(z'|z) = \frac{P(z,z')}{\sum_{z'} P(z,z')}$, where $P(z, z') = P(z|z', \phi')P(z'|\theta')$. The same steps as for sampling z apply, giving:

$$P(z' = k|z = y, z_{-i}, z'_{-i}) \propto \frac{n_{-yk} + \beta_Y}{n_{-\cdot k} + |L|\beta_Y}(n_{-dk} + \alpha) \tag{3.4}$$

3.3 Greedy Layer-Wise Training

Intuitively, the model cannot benefit from a prior on the label distribution if the prior is an untrained model of label dependencies. The abstract level of label dependencies has to be trained first, before it can be used as a

```
1: for each iteration h = 1 ... I do
2:   for each document d in D do
3:     for each word w_i in d do
4:       sample topic z'_i (Eq. 3.4)
5:     end for
6:     for each word w_i in d do
7:       sample label z_i (Eq. 3.5)
8:     end for
9:   end for
10: end for
```

Figure 3.2: Greedy Algorithm

```
1: for each iteration h = 1 ... I do
2:   for each document d in D do
3:     for each word w_i in d do
4:       sample topic z'_i (Eq. 3.4)
5:     end for
6:     for each word w_i in d do
7:       sample label z_i (Eq. 3.3)
8:     end for
9:   end for
10: end for
```

Figure 3.3: Non-Greedy Algorithm

Figure 3.4: The pseudo code for batch Fast-Dep.-LLDA

prior, otherwise the model is more likely to be "confused" by the untrained prior.

Another view is to consider the "explaining away" effect as the problem. If a document is well explained by a certain label, other labels become less likely. In supervised learning, the labels that explain the document are given and ideally all of them should contribute to the document-label distribution. Gibbs sampling can lead to certain labels being sampled extremely rarely or not at all if the document is well explained by a subset of the documents labels. During training this can be a problem, since similar labels become anti-correlated through the observed words whereas in fact they are often correlated.

Instead of training the complete model at once, a greedy layer-wise training procedure is proposed inspired by Hinton *et al.* [25] and Bengio *et al.* [5]. This means first the label layer is trained using Equation 3.3. Since the topic-label distributions ϕ' are not trained yet, it is assumed that they form a uniform prior on the label assignments z such that $P(z|w) \propto P(w|z)$. This leads to the following equation for sampling label assignments z during training of Fast-Dep.-LLDA:

$$P(z = y|w, z' = k, z_{-i}, z'_{-i}) \propto \frac{n_{-wy} + \beta}{n_{-\cdot y} + |V|\beta} \tag{3.5}$$

The model is guaranteed to converge to the optimum given the chosen parameters. The greedy model may be viewed as letting $\sum \beta_Y \to \infty$ which means the Dirichlet becomes a uniform distribution in case of symmetric β_Y. Greedy training corresponds to choosing the most extreme parameter value for β_Y, which leads to the second term vanishing from Equation 3.3 completely. Empirically, it is the case that on all tested multi-label datasets the convergence was better using greedy training than non-greedy training

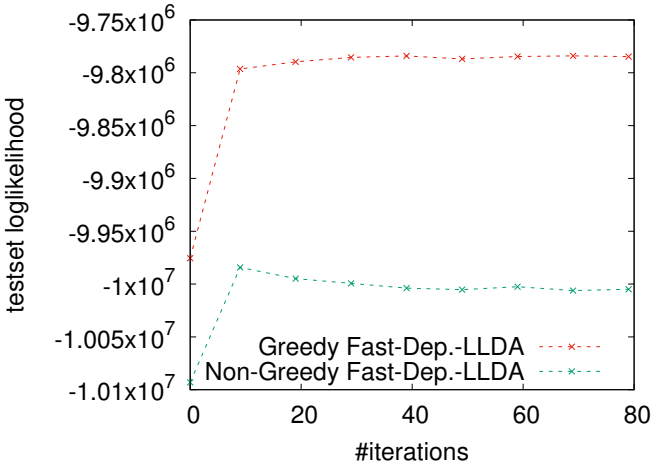


Figure 3.5: Testset log-likelihood on the EUR Lex dataset for the greedy and the non-greedy version of Fast-Dep.-LLDA. For the non-greedy version $\beta_Y = 0.01$ is chosen.

(compare the algorithms in Figure 3.4). As Figure 3.5 shows for the `EUR Lex` datasets, the testset log-likelihood increases for the greedy training procedure. For the non-greedy training procedure, the likelihood increases at first, but then decreases slightly and remains at a much lower level. This effect might be due to overfitting. The greedy training can be viewed as performing a kind of regularization by imposing a simplified prior during the training of the lower level of the model.

As Hinton *et al.* note, in the supervised setting each label only provides a few bits of constraints on the parameters which makes overfitting a much bigger problem than underfitting and going back to retrain the first level given the information from the topic level is likely to do more harm than good. The procedure is also similar to boosting in that weak models are stacked on top of each other, using the output of the previous models. Going back to improve earlier models based on later ones is not likely to do the model any good. Dep.-LLDA inadvertently also employed this principle by training the two parts of the model separately. As Fast-Dep.-LLDA shows, it is however not necessary to plug in a completely observed variable c and make the two parts conditionally independent (see Figure 3.1a).

Following Bengio *et al.*, the two layers are trained at the same time while still maintaining the greedy idea. By sampling with the above equation, the parameters of both layers are learned together, meaning, the second layer can immediately start training on the output of the first layer. This means it is not necessary to pick two separate parameters for the number of training iterations.

The analogy to Hinton's layer-wise greedy training is rather loose and, to

Table 3.2: Additional notation for SCVB-Dep. model

M	minibatch
C, C_j	number of overall tokens, number of tokens for document j
$\rho^{\phi}, \rho^{\phi'}, \rho^{\theta'}$	update parameters between zero and one
N, $\hat{N}$	expected counts, estimate for expected counts

my knowledge, without a formal, mathematical connection: Both methods rely on the same idea, however, Fast-Dep.-LLDA does not have a theoretical justification in terms of complimentary priors or convergence guarantees based on variational bounds. Both methods are based on the idea of training a hierarchical generative model in a greedy, layer-wise manner. Both methods perform well empirically as compared to their non-greedy counterparts. Thus, no claim is made that the connection goes deeper than one method being loosely inspired by the other.

Overall a compact model named Fast-Dep.-LLDA was derived with only three parameters, α, β, and β_Y, and an efficient Gibbs sampler with a greedy layer-wise training procedure.

3.4 Online Fast-Dep.-LLDA (SCVB-Dep.)

Scalability to large datasets or even potentially infinite data streams is an important requirement for many application contexts. Additionally to the training runtime, memory constraints may become a problem when performing Gibbs sampling over large datasets since the whole dataset has to be kept in memory. Online algorithms allow incremental updates using batches of data, making them memory efficient. Therefore, the online version of Fast-Dep.-LLDA, SCVB-Dependency, is presented in this section. For this, a method similar to the stochastic collapsed variational Bayes (SCVB) method by Foulds *et al.* [19] is developed with two important differences. First, variational update equations are derived that are suitable to Fast-Dep.-LLDA with its additional topic level. Second, a greedy layer-wise training algorithm is put forward that is applicable in the online setting. This enables to train a classifier with only one iteration over the dataset.

The fully factorized variational distribution of Fast-Dep.-LLDA is given by

$$q(z, z', \theta', \phi, \phi') = \prod_{ij} q(z_{ij}|\gamma_{ij}) \prod_{ij} q(z'_{ij}|\gamma'_{ij}) \prod_{j} q(\theta'_j|\tilde{\alpha_j}) \tag{3.6}$$

for tokens i and documents j.

In the equation, an additional variational parameter γ' is introduced for the topic assignments z'. However, computing the updates for γ and γ'

Algorithm 6 Online Training/Inference of SCVB-Dependency

Randomly initialize N^{ϕ}, $N^{\theta'}$, $N^{Z} := \sum_w N^{\phi}_w$, $N^{\phi'}$ and $N^{Z'} := \sum_y N^{\phi'}_y$
for each minibatch M **do**
  $\hat{N}^{\phi} := 0$; $\hat{N}^{\phi'} := 0$; $\hat{N}^{Z} := 0$; $\hat{N}^{Z'} := 0$
  for each document $\in M$, where j is the document index **do**
    for each burn-in pass **do**
      for each token i **do**
        Update $\lambda_{w_{ij}}$ (Equation 3.9)
        Update $N^{\theta'}_j$ (Equation 3.12)
      end for
    end for
    for each token i **do**
      Update λ_{ij} (Equation 3.9)
      Update $N^{\theta'}_j$ (Equation 3.12)
      $\hat{N}^{\phi}_{w_{ij}} := \hat{N}^{\phi}_{w_{ij}} + \frac{C}{|M|} \sum_k \lambda_{w_{ij}yk}$
      $\hat{N}^{\phi'}_{y_{ij}} := \hat{N}^{\phi'}_{y_{ij}} + \frac{C}{|M|} \lambda_{w_{ij}y_{ij}}$
      $\hat{N}^{Z} := \hat{N}^{Z} + \frac{C}{|M|} \sum_k \lambda_{w_{ij}yk}$
      $\hat{N}^{Z'} := \hat{N}^{Z'} + \frac{C}{|M|} \lambda_{w_{ij}y_{ij}}$
    end for
  end for
  Update N^{ϕ} (Equation 3.13)
  Update $N^{\phi'}$ (Equation 3.14)
  Update N^{Z} (Equation 3.15)
  Update $N^{Z'}$ (Equation 3.16)
end for

separately would lead to unnecessary computational effort. I propose to instead compute an intermediate value λ_{wyk} which corresponds to the expectation of a joint occurrence of word w, label y and topic k which can be expressed in terms of an expectation of the indicator function $\mathbb{1}$ which is one if these values occur together and otherwise zero: $\mathbb{E}[\mathbb{1}[w_i = w, y_i = y, k_i = k]]$, where i is the index of the token.

According to standard variational Bayes derivations, the lower bound for the posterior is given by

$$\begin{aligned}\mathcal{L}(q) =& \mathbb{E}[\log(P(\theta'|\alpha)] + \\ & \mathbb{E}[\log(P(z'|\theta')] + \mathbb{E}[\log(P(z|z', \phi')] + \mathbb{E}[\log(P(w|z, \phi)] - \mathcal{H}(q)\end{aligned}$$

where $\mathcal{H}(q)$ denotes the entropy of the variational distribution.

The update for λ_{ij} is now obtained by expanding the lower bound and

isolating all terms containing λ,

$$\mathcal{L}_{[\lambda]}(q) = \sum_N \sum_L \sum_K \lambda_{wyk}(\psi(\tilde{\alpha_k}) - \psi(\sum_{j=1}^{K} \tilde{\alpha_j})) + \sum_N \sum_L \sum_K \sum_V \lambda_{wyk} w_n^v \log(\phi_{vy}\phi'_{yk})$$

adding Lagrange multipliers, computing the derivative with respect to λ_{ij}, and setting it to zero. This leads to the following update equation for λ_{ij}, assuming that w is the word at position i of document j.

$$\lambda_{wyk} \propto \phi_{wy}\phi'_{yk} \exp(\psi(\tilde{\alpha_k})) \tag{3.7}$$

If ϕ and ϕ' are included in the variational distribution, it is marginalized over θ', ϕ, and ϕ' and only the first term of the Taylor approximation is used as in [58], the following update is obtained:

$$\lambda_{wyk} \propto \frac{N^{\phi}_{w_{ij},y} + \eta_w}{N^Z_y + \sum_w \eta_w} \frac{N^{\phi'}_{y_{ij},k} + \eta_y}{N^{Z'}_k + \sum_y \eta_y} (N^{\theta'}_{jk} + \alpha) \tag{3.8}$$

From this $\gamma_{wk} = \sum_k \lambda_{wyk}$ and $\gamma_{yk} = \sum_w \lambda_{wyk}$ can be recovered by marginalization, however, these variational distributions are not needed for the updates, since it is straightforward to use λ directly. N^Z is a vector storing the expected number of words for each label. N^{ϕ} is the expected number of tokens for words w and labels y in the whole corpus. Additionally, $N^{Z'}$ stores the expected number of tokens for each topic, $N^{\phi'}$ is the expected number of tokens for labels y and topics k, and $N^{\theta'}_j$ is the expected number of words per topic, only for document j.

The stochastic update equations are derived analogously to the ones by Foulds *et al.* [19], except that we update ϕ, ϕ' and θ' instead of just ϕ and θ. The terminology of Foulds *et al.* is used for easier comparison. Updates are done for minibatches M with update percentage ρ. C is the number of overall tokens and C_j is the number of tokens in document j.

For each token (the ith word in the jth document) λ_{ijyk} is calculated for label y and topic k, where during training λ only has to be calculated for the labels of the document and should be set to zero for all other labels.

$$\lambda_{ijyk} :\propto \lambda^W_{ijy}\lambda^T_{ijyk} \tag{3.9}$$

$$\lambda^W_{ijy} :\propto \frac{N^{\phi}_{w_{ij},y} + \eta_w}{N^Z_y + \sum_w \eta_w} \tag{3.10}$$

$$\lambda^T_{ijyk} :\propto \frac{N^{\phi'}_{y_{ij},k} + \eta_y}{N^{Z'}_k + \sum_y \eta_y} (N^{\theta'}_{jk} + \alpha) \tag{3.11}$$

Because greedy layer-wise training is used, the two parts of the model can be trained separately (see Algorithm 7), whereas during testing the full model has to be used (see Algorithm 6). In Algorithm 7, the first layer treats every word as an input token and updates the word-label distribution based on λ^W (lines 11–15), whereas the second layer treats each label assignment as an input token and learns the label-topic distributions based on λ^T (lines 16–21). Since the model is supposed to be trained online, it is not possible to wait for the greedy algorithm to learn the first layer before moving on to the second layer. Therefore, the input probabilities of the second layer are initialized by using the true labels. Thereby, both layers can be trained simultaneously while not having to view any document more than once.

The update equations using λ and the update parameter ρ for estimating N^{ϕ}, N^{Z}, $N^{\phi'}$, $N^{Z'}$ and $N^{\theta'}$ are:

$$N_j^{\theta'} := (1-\rho^{\theta'})N_j^{\theta'} + \rho^{\theta'} C_j \sum_y \lambda_{ijy} \tag{3.12}$$

$$N^{\phi} := (1-\rho^{\phi})N^{\phi} + \rho^{\phi}\hat{N}^{\phi}, \tag{3.13}$$

where $\hat{N}^{\phi} = \frac{C}{|M|}\sum_{i,j\in M}\sum_k Y^{i,j,k}$ and $Y^{i,j,k}$ is a $|V|$ times $|L|$ matrix with the w_ith row corresponding to $\sum_k \lambda_{i,j,y,k}$ and zero in all other places. Note that we multiply by $\frac{C}{|M|}$ here to scale the estimate obtained from the minibatch to the whole corpus. Multiplying an estimate for one token by C, the number of tokens, we arrive at the expectation for the whole corpus. Since we do this for each token in the minibatch, we divide by $|M|$, the number of tokens in the minibatch, to get an average over all tokens in the minibatch.

$$N^{\phi'} := (1-\rho^{\phi'})N^{\phi'} + \rho^{\phi'}\hat{N}^{\phi'} \tag{3.14}$$

where $\hat{N}^{\phi'} = \frac{C}{|M|}\lambda_{ij}$.

$$N^{Z} := (1-\rho^{\phi})N^{Z} + \rho^{\phi}\hat{N}^{Z}, \tag{3.15}$$

where $\hat{N}^{Z} = \frac{C}{|M|}\sum_{i,j\in M}\sum_k \lambda_{ij.k}$.

$$N^{Z'} := (1-\rho^{\phi'})N^{Z'} + \rho^{\phi'}\hat{N}^{Z'}, \tag{3.16}$$

where $\hat{N}^{Z'} = \frac{C}{|M|}\sum_{i,j\in M}\sum_y \lambda_{ijy}$.

The whole inference algorithm is given in Algorithm 6. For each document zero or more burn-in passes are done to get a better estimate of $N^{\theta'}$ (line 5–9). Then the estimates $\hat{N}^{\phi}$, $\hat{N}^{\phi'}$, $\hat{N}^{Z}$, and $\hat{N}^{Z'}$ are updated for each token in the current minibatch (line 11–18). Finally, for each minibatch the global estimates N^{ϕ}, $N^{\phi'}$, N^{Z}, and $N^{Z'}$ are updated (line 20–23).

To sum up, a variational Bayes learning algorithm was derived for Fast-Dep.-LLDA to use greedy layer-wise training in the online setting.

Algorithm 7 Online Greedy Layer-wise Training of SCVB-Dependency

Randomly initialize N^{ϕ}, $N^{\theta'}$, $N^{Z} := \sum_w N^{\phi}_w$, $N^{\phi'}$ and $N^{Z'} := \sum_y N^{\phi'}_y$
for each minibatch M **do**
  $\hat{N}^{\phi} := 0$; $\hat{N}^{\phi'} := 0$; $\hat{N}^{Z} := 0$; $\hat{N}^{Z'} := 0$
  for each document $\in M$, where j is the document index **do**
    for each burn-in pass **do**
      for each label y **do**
        Update λ^{T}_{ijy} (Equation 3.11)
        $N^{\theta'}_j := (1 - \rho^{\theta'})N^{\theta'}_j + \rho^{\theta'} C_j \lambda^{T}_{ijy}$
      end for
    end for
    for each token i **do**
      Update λ^{W}_{ij} (Equation 3.10)
      $\hat{N}^{\phi}_{w_{ij}} := \hat{N}^{\phi}_{w_{ij}} + \frac{C}{|M|}\lambda^{W}_{ij}$
      $\hat{N}^{Z} := \hat{N}^{Z} + \frac{C}{|M|}\lambda^{W}_{ij}$
    end for
    for each label y **do**
      Update λ^{T}_{ijy} (Equation 3.11)
      $N^{\theta'}_j := (1 - \rho^{\theta'})N^{\theta'}_j + \rho^{\theta'} C_j \lambda^{T}_{ijy}$
      $\hat{N}^{\phi'}_{y_{ij}} := \hat{N}^{\phi'} + \frac{C}{|M|}\lambda^{T}_{ijy}$
      $\hat{N}^{Z'} := \hat{N}^{Z'} + \frac{C}{|M|}\lambda^{T}_{ijy}$
    end for
  end for
  Update N^{ϕ} (Equation 3.13)
  Update $N^{\phi'}$ (Equation 3.14)
  Update N^{Z} (Equation 3.15)
  Update $N^{Z'}$ (Equation 3.16)
end for

3.5 Reversed Fast-Dep.-LLDA

One problem when learning a topic model where each topic corresponds to a label is that the labels might not be a good way to represent the data. There might be similar labels (e.g. model, models and modeling) that could be grouped together into a single topic, and there might be broad labels that could be subdivided into several topics. To get a better representation of the data, it is possible to use the same model with topics and labels swapped. This means each label is associated with a distribution over topics.

Fast-Dep.-LLDA can also be applied in this context, despite the fact that now the first layer of topics cannot be trained without information from its prior since it is not supervised anymore. The second layer however,

since it is the supervised level, can be trained greedily by using a uniform document-label distribution θ'. Thereby the well-known author-topic model is recovered during training. During testing the dependence on the prior document-label distribution θ' has to be put back into the model.

This leads to the following update equations for label assignments z' and topic assignments z during training:

$$P(z = y|w, z' = k, z_{-i}, z'_{-i}) \propto \frac{n_{-wy} + \beta}{n_{-\cdot y} + |V|\beta}(n_{-yk} + \beta_Y) \tag{3.17}$$

$$P(z' = k|z = y, z_{-i}, z'_{-i}) \propto \frac{n_{-yk} + \beta_Y}{n_{-\cdot k} + |L|\beta_Y} \tag{3.18}$$

During testing Equation 3.4 has to be used instead of Equation 3.18.

This model is not meant for classification, but for generating representative word clouds that are not as much dependent on the given label assignments.

(a) Antibiotics

(b) Multiple Abnormalities

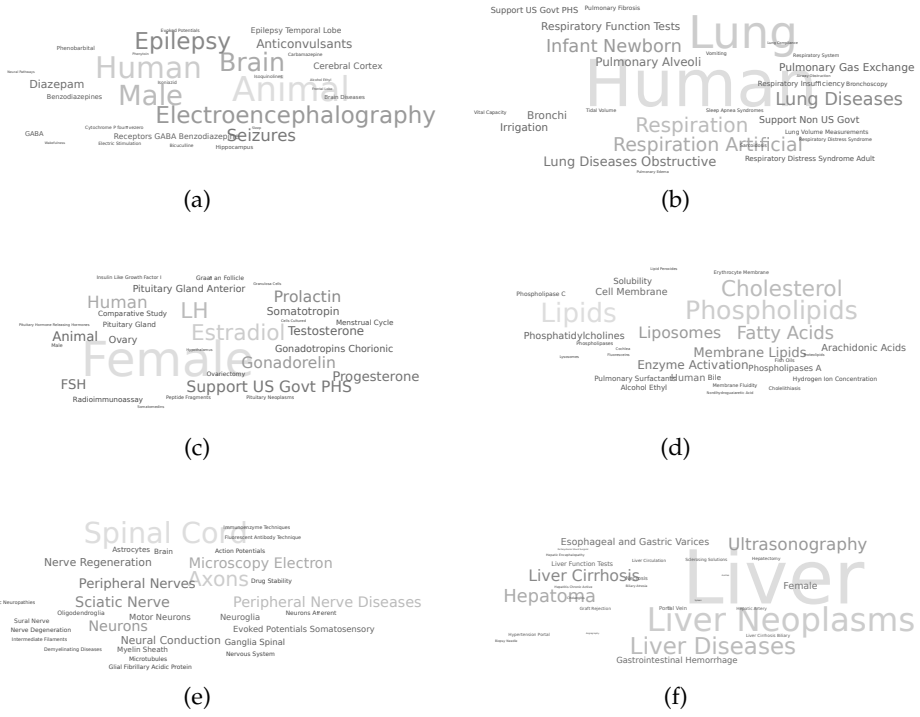
(c) Adenocarcinoma

(d) Adrenal Cortex Hormones

(e) Aerosols

(f) Alopecia

Figure 3.6: Word clouds for the `Ohsumed` dataset after training for 100 iterations using SCVB-Dep.. The size of the words is scaled according to their frequencies. Each word cloud corresponds to one label and includes the 30 most frequent words for this label.



(a) (b) (c) (d) (e) (f)

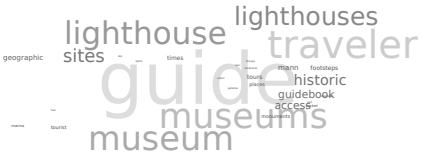
Figure 3.7: Label clouds for the `Ohsumed` dataset after training for 100 iterations using SCVB-Dep.. The size of the labels is scaled according to their frequencies. Each label cloud corresponds to one topic and includes the 30 most frequent labels for this topic.

3.6 Prediction

For prediction in Fast-Dep.-LLDA, two prediction strategies are differentiated. For both strategies, the estimated matrices ϕ and ϕ' are fixed and only the document counts are modified to get an estimate of θ'. However, unlike in Dep.-LLDA, there is no explicit document-label distribution incorporated in the model, there is only the document-topic distribution θ'. This is not a problem since the conditional probabilities for z can be used directly for prediction. The first strategy (S1) estimates the probability of label y in document d given $z' = k$ as $\frac{1}{n^d} \sum_{w \in d} \frac{n_{-wy}+\beta}{n_{-\cdot y}+|V|\beta} \frac{n_{-yk}+\beta_Y}{n_{-\cdot k}+|L|\beta_Y}(n_{-dk}+\alpha)$. The second strategy (S2) is more similar to the evaluation strategy of Dep.-LLDA. Here, the posterior predictive distribution over labels is estimated as $P(z = y|\phi', \theta') = \sum_{k=1}^{K} \phi'_{yk}\theta'_{dk}$. The labels are then sampled from a standard LLDA model with an asymmetric label prior $\alpha_y = n^d \cdot P(z = y|\phi', \theta')$. This means the conditional probability for sampling of the labels as well as estimation is given by $P(z = y|rest) \propto \frac{n_{-wy}+\beta}{n_{-\cdot y}+|V|\beta}(n_{-dy} + \alpha_y)$. This evalu-

(a) Breakfast & cereal bars

(b) Small Animals



(c) Tourist destinations & museums

(d) Vegetables

(e) Assessment

(f) Atlases & maps

Figure 3.8: Word clouds for the `Amazon` dataset after training for 100 iterations using SCVB-Dep.. The size of the words is scaled according to their frequencies. Each word cloud corresponds to one label and includes the 30 most frequent words for this label.

ation strategy is more expensive since the asymmetric label prior α has to be calculated, however, it allows to explicitly represent the document-label distribution needed for prediction.

In the experiments, ten chains are run for each classifier, taking 100 samples from each chain to get an estimate. These estimates are averaged again over the different chains to produce the final estimate.

For Dep.-LLDA, the original publication uses an estimate of θ for the prediction. However to make the comparison fair and because it delivers better results[1], an average over the conditional probability for $z = y$ (Equation 2.40) was also used in this case: $\frac{1}{n^d} \sum_{w \in d} \frac{n_{-wy}+\beta}{n_{-\cdot y}+|V|\beta}(n_{-dy} + \alpha')$.

In the case of the online models SCVB and SCVB-Dependency, a normalized version of N^Z was used for the prediction of each document.

[1]See the manuscript by Papanikolaou *et al.* [44] for a formal justification of this approach.

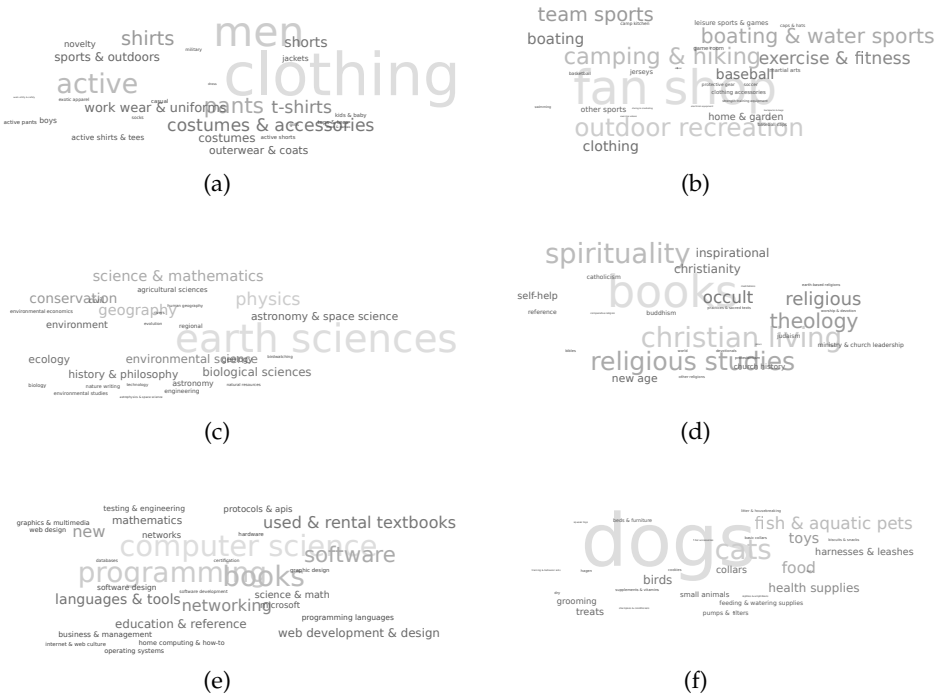


Figure 3.9: Label clouds for the `Amazon` dataset after training for 100 iterations using SCVB-Dep.. The size of the labels is scaled according to their frequencies. Each label cloud corresponds to one topic and includes the 30 most frequent labels for this topic.

3.7 Computational Complexity

Both methods, Dep.-LLDA as well as Fast-Dep.-LLDA have a computational complexity of $O(|L| + K)$ for one sampling step since one of $|L|$ labels and one of K topics have to be sampled. For Fast-Dependency-LDA this can be reduced to $O(|L_k| + K_d)$, where $|L_k|$ is the number of labels in topic k and K_d is the number of topics in document d using more efficient sampling techniques [35]. For Dependency-LDA only the sampling of topics can be sped up, leading to an improved complexity of $O(|L| + K_d)$. Furthermore, the calculation of α' leads to a higher computation time of Dep.-LLDA as compared to the evaluation strategy S1 even though the complexity is not affected as long as it is not assumed that the document length is much lower than the number of topics. The second evaluation strategy S2 also involves the calculation of α meaning the computational complexity is equal to that of Dep.-LLDA in that case. Summing up, both methods have the same complexity, however, Fast-Dependency-LDA using evaluation strategy S1 is expected to be faster in practice and when more efficient sampling methods are used, the S1 method has an improved complexity.

Table 3.3: Number of labels, attributes, instances, cardinality and average number of tokens per document for the used datasets

number of	EUR Lex	Ohsumed	Amazon
labels	3,955	11,220	13,330
attributes	5,000	20,000	20,000
instances	19,314	52,796	1,195,943
cardinality	5.255	10.576	5.041
av.#tokens	846	99	111

3.8 Experiments

3.8.1 Binary Predictions

All compared classifiers produce predictions in the form of a probability distribution over labels. To be able to compare on standard multi-label classification measures such as the F-measure, the probabilities need to be transformed to binary predictions (see Section 2.2.2 for background). The approach taken is to train a regression model on the features x that predicts the number of labels to set to true for each instance. The resulting number is then rounded to the nearest integer and the most probable labels are set to true. The reason for this approach is that it is independent of the used classifier, meaning that the number of labels can be predicted for each test instance once and the prediction can be used for the output of each classifier. In experiments not reported here in detail, this approach outperformed the approach with a cut-off set to an arbitrary fixed value. To train the model, a subset of 10,000 training documents was used. As a regression method Lasso was chosen. Parameters were chosen via cross-validation.

3.8.2 Datasets

The evaluation was performed on three publicly available multi-label text datasets with a focus on large datasets with many labels. The dataset statistics are shown in Table 3.3. A fixed testset was used for all datasets. `EUR Lex` is a dataset of legal documents concerning the European Union. It is hand annotated with almost 4,000 labels. For `EUR Lex` [39] the last 10% are reserved for testing. The `Ohsumed` dataset[2] is a subset of MEDLINE medical abstracts that were collected in 1987 and that have 11,220 different human-assigned MeSH descriptors. The provided abstracts and MeSH descriptors were used and other information, such as further qualifiers or authors, disregarded. Stop words were removed and the most frequent 20,000 features

[2] `http://trec.nist.gov/data/t9_filtering.html`

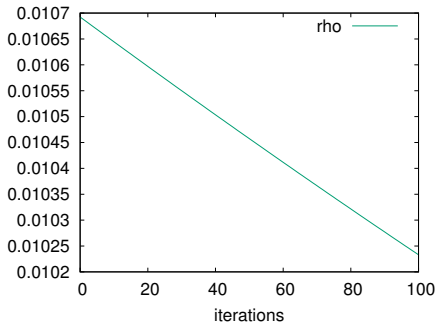


Figure 3.10: This figure shows the parameter ρ^{ϕ} over 100 iterations for the settings mentioned in the text.

retained. 10% of the data were separated for the testset. The `Amazon` dataset consists of more than one million product reviews, annotated with corresponding product categories. The original dataset is available from `http://manikvarma.org/downloads/XC/XMLRepository.html` under the name AmazonCat-13K. 10,000 documents were randomly sampled from the testing dataset since the proposed evaluation methods are not feasible for larger testsets. The features were further pruned to exclude stopwords and only use the 20,000 most frequent features. Since the dataset is only provided in a processed format and the authors were not able to provide the unprocessed dataset, the following steps were taken to arrive at a raw tokenized format: The document length was normalized to 100. The feature values were then rounded to the nearest integer with a minimum value of 1.

For the experiments with BR, a TF-IDF transform of the features is applied for all datasets.

3.8.3 Experimental Setting

Topic Modeling Methods

For the batch setting the original Dep.-LLDA was compared to Fast-Dep.-LLDA. Dep.-LLDA serves as the main baseline since it is known to consistently outperform other topic modeling methods such as LLDA. It is also closest to Fast-Dep.-LLDA with respect to interpretability. Most other multi-label classifiers are not able to deal with datasets of the size and the number of labels that are tested here. The implementation of Dep.-LLDA uses the fast inference method and applies the heuristic of scaling α' according to the document length. 100 topics were used for all datasets and methods. This is the same default setting employed by Rubin *et al.* who reported that higher numbers tended to induce redundancy in the topics. I also did not observe a performance improvement. However, it could be hypothesized that using nonparametric methods based on hierarchical Dirichlet processes [59] pos-

sibly enable the use of larger topic numbers in future work. Additionally, $\beta = \beta_Y = 0.01$, $\sum \gamma = 10$, $\sum \alpha = 30$ and $\eta = 100$ are used during testing. These values also correspond to the values used by Rubin *et al.* and the values for β and β_c are frequently used in most literature on LDA topic models. The optimization of β was shown to have no positive effect on results in previous work [67], however, grid search for optimal symmetric values would possibly lead to a small improvement in the batch setting. The main emphasis is on the streaming setting and a grid search could lead to poor results if it is only done on the first batch. Fast-Dep.-LLDA uses the same values for β and $\sum \alpha = 30$.

To evaluate the online classifier, SCVB-Dependency, each classifier is first trained for 100 iterations on an initial batch of the data and then sequentially tested on the next batch before training on it using only one iteration over each batch. As a baseline Fast-Dep.-LLDA is compared to the standard stochastic collapsed variational Bayes (SCVB) model by Foulds *et al.* [19]. The update parameter ρ is determined as $\frac{s}{(\tau+t)^\kappa}$ for iteration t with $s = 1$, $\tau = 1000$ and $\kappa = 0.9$ for $\rho^{\theta'}$ and $s = 10$ and $\tau = 2000$ for ρ^{ϕ} and $\rho^{\phi'}$ (see Figure 3.10). Additionally, default parameter values $\eta_w = \eta_y = 0.01$ and $\alpha = 0.1$ are used.

BR(SVM)

BR (see Section 2.2.3) is the second baseline because it does not consider label dependencies and therefore allows assessing the effect of modeling dependencies. Binary Relevance learns separate SVMs for each label. The publicly available LIBLINEAR (version 1.98) is used for the SVMs, implemented in C++. The SVM parameter C is optimized in the range $[10^{-3}, \ldots, 10^{3}]$ using 3-fold cross-validation for each label separately. The output of an SVM for each label represents the distance to the decision surface. To transform this to probability outputs, it would be necessary to apply Platt's scaling or a similar method. Platt's scaling involves training a logistic regression model on the output of the SVM. In this case this would mean, producing the SVM-output for the whole training set and training an additional logistic regression model for each label. Possibly, this procedure would improve the SVM results (I am are not aware of any systematic study concerning this issue for multi-label classification), however, the distances were used directly in this work since the training of BR is already expensive without further postprocessing of the results. Previous work follows the same procedure [43] and Rubin *et al.* [54] also do not mention a transformation of SVM outputs to probabilities. This is especially problematic in their evaluation since they do not use thresholds for each instance, but use a label-based cut-off for the whole testset, i.e. they select the number of positive instances per label. For each label, outputs are generated using a sigmoid as $o_1 = 1/(1 + \exp(-distance))$ and $o_2 = 1 - o_1$, where distance

is the distance to the decision surface. To be able to train a BR on the large datasets, a python reimplementation was used and the models were trained in parallel. Note that the use of linear kernels is well supported by previous literature on text classification methods as they are not only much more efficient in training, but also have a comparable performance to nonlinear kernels in this high dimensional setting [34].

FastXML

As a state-of-the-art method of comparison in extreme multi-label learning, FastXML by Prabhu and Varma [46] was chosen. This model is an ensemble of decision trees similar to random forests, but optimizing a different loss function leading to improved results. The C-Code is provided on the author's website[3] and the default settings of the software were used.

PD-Sparse

PD-Sparse is another multi-label method for extreme multi-label classification. Yen *et al.* [75] propose to use a margin-maximizing loss with L1-penalty yielding an extremely sparse solution. This classifier is sublinear in the number of labels and thus applicable in even more extreme settings with millions of labels. The C-Code is available at `http://manikvarma.org/downloads/XC/XMLRepository.html` and default settings were used.

3.8.4 Results

Batch Methods

As the results in Tables 3.4–3.7 show, with regards to Fast-Dep.-LLDA, evaluation strategy S2 is superior to evaluation strategy S1. This shows that it is important to explicitly represent the label distributions during prediction. Tables 3.4 and 3.5 show the results for the batch methods where the final distributions were obtained by averaging the distributions from ten different chains. As the table shows, Fast-Dep.-LLDA has almost always the best results for micro- and macro-averaged AUC. However, BR is better on the binary classification measures. Probably, this is due to the issue discussed in Section 3.8.3. BR does by default not produce probabilities as outputs, which might affect the ranking performance between different instances. The binary cut-offs are determined instance-wise, which means that the binary results only depend on the ranking of labels per instance. The only binary measure where BR is not best is macro-averaged F-measure, where BR is only best on the `Amazon` dataset, whereas the topic modeling methods are better on the other two datasets. The `Amazon` dataset is the largest

[3] `http://manikvarma.org/downloads/XC/XMLRepository.html`

Table 3.4: Ranking measures for Binary Relevance with SVMs, Dep.-LLDA, Fast-Dep.-LLDA, and Fast-Dep.-LLDA with the more complex evaluation strategy S2. Results are produced by averaging distributions from 10 different sampling chains. S1 and S2 denote the different evaluation strategies for Fast-Dep.-LLDA. The best result for each dataset is highlighted in bold.

method	EUR Lex	Ohumed	Amazon	av. rank
		Micro-averaged AUC		
PD-Sparse	-	-	-	-
FastXML	-	-	-	-
BR(SVM)	0.8253 (4)	0.9186 (4)	0.9800 (4)	4.0000
Dep.LLDA	**0.9565** (1)	0.9608 (3)	0.9902 (3)	2.3333
F.Dep.-S1	0.9442 (3)	0.9618 (2)	0.9921 (2)	2.3333
F.Dep.-S2	0.9564 (2)	**0.9620** (1)	**0.9931** (1)	**1.3333**
		Macro-averaged AUC		
PD-Sparse	-	-	-	-
FastXML	-	-	-	-
BR(SVM)	0.7680 (4)	0.8521 (4)	0.9478 (4)	4.0000
Dep.LLDA	0.9228 (2)	0.8929 (3)	0.9707 (3)	2.6667
F.Dep.-S1	0.8993 (3)	0.8978 (2)	0.9749 (2)	2.3333
F.Dep.-S2	**0.9277** (1)	**0.9077** (1)	**0.9754** (1)	**1.0000**
		Rank One Error		
PD-Sparse	0.7659 (5)	0.3395 (3)	0.9725 (6)	4.6667
FastXML	0.8156 (6)	**0.1610** (1)	0.9227 (5)	4.0000
BR(SVM)	**0.3692** (1)	0.1726 (2)	**0.1133** (1)	**1.3333**
Dep.LLDA	0.4148 (2)	0.4315 (5)	0.4993 (3)	3.3333
F.Dep.-S1	0.5365 (4)	0.4330 (6)	0.5091 (4)	4.6667
F.Dep.-S2	0.4363 (3)	0.4185 (4)	0.4981 (2)	3.0000

Table 3.5: Classification measures for Binary Relevance with SVMs, Dep.-LLDA, Fast-Dep.-LLDA, and Fast-Dep.-LLDA with the more complex evaluation strategy S2. Results are produced by averaging distributions from 10 different sampling chains. S1 and S2 denote the different evaluation strategies for Fast-Dep.-LLDA. The best result for each dataset is highlighted in bold.

method	EUR Lex	Ohumed	Amazon	av. rank
		Micro-averaged F-Measure		
PD-Sparse	0.1227 (6)	0.3101 (6)	0.0186 (6)	6.0000
FastXML	0.1138 (5)	0.3656 (4)	0.0491 (5)	4.6667
BR(SVM)	**0.3802** (1)	**0.4388** (1)	**0.6113** (1)	**1.0000**
Dep.LLDA	0.3546 (2)	0.4011 (2)	0.3760 (2)	2.0000
F.Dep.-S1	0.2697 (4)	0.3350 (5)	0.3226 (4)	4.3333
F.Dep.-S2	0.3376 (3)	0.4002 (3)	0.3640 (3)	3.0000
		Macro-averaged F-Measure		
PD-Sparse	0.0166 (5)	**0.1958** (1)	0.0322 (5)	3.6667
FastXML	0.0137 (6)	0.0554 (6)	0.0252 (6)	6.0000
BR(SVM)	0.1492 (2)	0.1409 (4)	**0.3185** (1)	2.3333
Dep.LLDA	**0.1571** (1)	0.1786 (3)	0.1935 (3)	2.3333
F.Dep.-S1	0.0712 (4)	0.0760 (5)	0.1342 (4)	4.3333
F.Dep.-S2	0.1466 (3)	0.1819 (2)	0.1959 (2)	2.3333
		Micro-averaged Precision		
PD-Sparse	0.1279 (6)	0.2861 (6)	0.0183 (6)	6.0000
FastXML	0.1187 (5)	0.3373 (5)	0.0484 (5)	5.0000
BR(SVM)	**0.3964** (1)	**0.4537** (1)	**0.6211** (1)	**1.0000**
Dep.LLDA	0.3698 (2)	0.4147 (2)	0.3820 (2)	2.0000
F.Dep.-S1	0.2794 (4)	0.3464 (4)	0.3278 (4)	4.0000
F.Dep.-S2	0.3520 (3)	0.4138 (3)	0.3698 (3)	3.0000
		Micro-averaged Recall		
PD-Sparse	0.1179 (5)	0.3385 (5)	0.0189 (6)	5.3333
FastXML	0.1094 (6)	0.3990 (2)	0.0499 (5)	4.3333
BR(SVM)	**0.3652** (1)	**0.4248** (1)	**0.6018** (1)	**1.0000**
Dep.LLDA	0.3407 (2)	0.3883 (3)	0.3701 (2)	2.3333
F.Dep.-S1	0.2607 (4)	0.3243 (6)	0.3176 (4)	4.6667
F.Dep.-S2	0.3244 (3)	0.3875 (4)	0.3583 (3)	3.3333
		Hamming Loss		
PD-Sparse	0.0023 (5)	0.0035 (6)	0.0024 (6)	5.6667
FastXML	0.0023 (6)	0.0032 (5)	0.0023 (5)	5.3333
BR(SVM)	**0.0016** (1)	**0.0011** (1)	**0.0003** (1)	**1.0000**
Dep.LLDA	0.0017 (2)	0.0012 (3)	0.0005 (2)	2.3333
F.Dep.-S1	0.0019 (4)	0.0013 (4)	0.0005 (4)	4.0000
F.Dep.-S2	0.0018 (3)	0.0012 (2)	0.0005 (3)	2.6667

Table 3.6: Ranking measures for Binary Relevance with SVMs, Dep.-LLDA, Fast-Dep.-LLDA, and Fast-Dep.-LLDA with the more complex evaluation strategy S2. Results are produced from only **one sampling chain**. */°Statistically significant difference to Dep.-LLDA at a level of **0.05**. **/°°Statistically significant difference to Dep.-LLDA at a level of **0.01** according to a Wilcoxon signed-rank test/T-test. The best result for each dataset is highlighted in bold.

dataset	Dep.-LLDA	Fast-Dep.-LLDA (S1)	Fast-Dep.-LLDA (S2)
		Micro-averaged AUC	
EUR Lex	0.9341 (2)	0.9236**°° (3)	**0.9348***° (1)
Ohsumed	0.9514 (3)	**0.9553****°° (1)	0.9552****°° (2)
Amazon	0.9853 (3)	0.9804****°° (2)	**0.9907****°° (1)
average rank	2.6667	2.0000	**1.3333**
		Macro-averaged AUC	
EUR Lex	0.8765 (2)	0.8637****°° (3)	**0.8885****°° (1)
Ohsumed	0.8565 (3)	0.8702****°° (2)	**0.8796****°° (1)
Amazon	0.9570 (2)	0.9505 (3)	**0.9634****°° (1)
average rank	2.3333	2.6667	**1.0000**
		Rank One Error	
EUR Lex	**0.4323** (1)	0.6323**°° (3)	0.4438*°° (2)
Ohsumed	0.4447 (2)	0.4721**°° (3)	**0.4352****°° (1)
Amazon	**0.5228** (1)	0.6875**°° (3)	0.5300****°° (2)
average rank	**1.3333**	3.0000	1.6667

Table 3.7: Classification measures for Binary Relevance with SVMs, Dep.-LLDA, Fast-Dep.-LLDA, and Fast-Dep.-LLDA with the more complex evaluation strategy S2. Results are produced from only **one sampling chain**. */°Statistically significant difference to Dep.-LLDA at a level of **0.05**. **/°°Statistically significant difference to Dep.-LLDA at a level of **0.01** according to a Wilcoxon signed-rank test/T-test. The best result for each dataset is highlighted in bold.

dataset	Dep.-LLDA	Fast-Dep.-LLDA (S1)	Fast-Dep.-LLDA (S2)
		Micro-averaged F-Measure	
EUR Lex	**0.3335** (1)	0.2179**°° (3)	0.3293* (2)
Ohsumed	0.3856 (2)	0.3102**°° (3)	**0.3866** (1)
Amazon	**0.3574** (1)	0.2076**°° (3)	0.3388**°° (2)
average rank	**1.3333**	3.0000	1.6667
		Macro-averaged F-Measure	
EUR Lex	**0.1466** (1)	0.0561**°° (3)	0.1449 (2)
Ohsumed	0.1673 (2)	0.0680**°° (3)	**0.1711****°° (1)
Amazon	**0.1774** (1)	0.0810**°° (3)	0.1712**°° (2)
average rank	**1.3333**	3.0000	1.6667
		Micro-averaged Precision	
EUR Lex	**0.3477** (1)	0.2272**°° (3)	0.3433* (2)
Ohsumed	0.3987 (2)	0.3208**°° (3)	**0.3998** (1)
Amazon	**0.3632** (1)	0.2110**°° (3)	0.3442**°° (2)
average rank	**1.3333**	3.0000	1.6667
		Micro-averaged Recall	
EUR Lex	**0.3204** (1)	0.2093**°° (3)	0.3163* (2)
Ohsumed	0.3733 (2)	0.3004**°° (3)	**0.3743** (1)
Amazon	**0.3519** (1)	0.2044**°° (3)	0.3335**°° (2)
average rank	**1.3333**	3.0000	1.6667
		Hamming Loss	
EUR Lex	**0.0018** (1)	0.0021**°° (3)	0.0018* (2)
Ohsumed	0.0012 (2)	0.0014**°° (3)	**0.0012***° (1)
Amazon	**0.0005** (1)	0.0006**°° (3)	0.0005**°° (2)
average rank	**1.3333**	3.0000	1.6667

dataset which might provide BR with enough training data to be able to predict the labels without taking dependencies into account. Macro-averaged measures are indicative for the performance on rare labels. Therefore, the conclusion is drawn that in cases where the number of labels is large relative to the size of the training dataset, the topic modeling methods are better at predicting rare labels.

For PD-Sparse and the decision tree method FastXML only binary measures and the ranking loss are provided since the methods do not produce probability outputs. In comparison to BR and the topic modeling methods, these two extreme multi-label methods are mostly worse. Only for the `Ohsumed` dataset PD-Sparse is best on macro-averaged F-measure whereas FastXML is best on rank one error. While these extreme multi-label classification methods are scalable and could easily handle even bigger label and feature sets, they are not competitive in terms of classification performance with the other medium-large-scale multi-label classifiers.

In Tables 3.6 and 3.7 the results of the topic modeling classifiers are compared on only one chain and with significance tests. Significance is tested using a t-test comparing Fast-Dep.-LLDA to the original Dep.-LLDA. Fast-Dep.-LLDA (S2) is significantly better than Dep.-LLDA on micro- and macro-averaged AUC. For the binary classification measures the results are less pronounced. Fast-Dep.-LLDA (S2) has better results on the `Ohsumed` dataset, but for the micro-averaged measures they are not significant. For macro-averaged F-measure, Fast-Dep.-LLDA (S2) is significantly better for the `Ohsumed` dataset, but worse for the other two datasets.

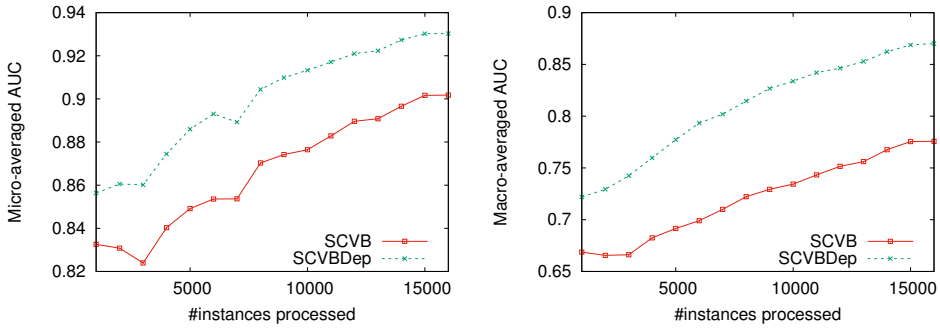
Online Methods

Figure 3.11 shows the performance of the online methods for micro- and macro-averaged AUC. Each classifier is first trained on an initial batch of instances for 100 iterations and then subsequently tested on the next batch and updated on the next batch using one iteration. The results are averaged over ten runs. The Fast-Dependency-SCVB method outperforms the plain SCVB on all datasets and both measures. This shows that Fast-Dependency-SCVB is able to learn the dependencies in the online setting.

Runtime

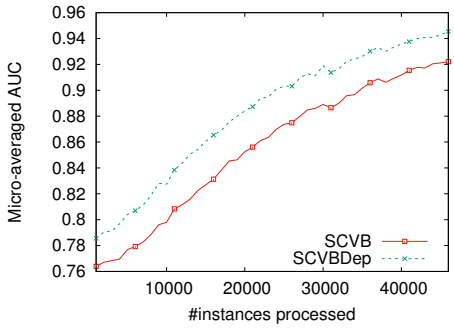
All experiments are performed on an Intel Core i7-4770K CPU 3.50GHz x 8. All methods were implemented in Java.

The *training* runtime of the batch vs. the online version of Fast-Dependency-SCVB is plotted in Figure 3.12a to 3.12c against micro-averaged AUC. On the `EUR Lex` and `Amazon` datasets, the online method converges much faster than the batch method and is able to provide results long before the batch method has even finished the first iteration. On the

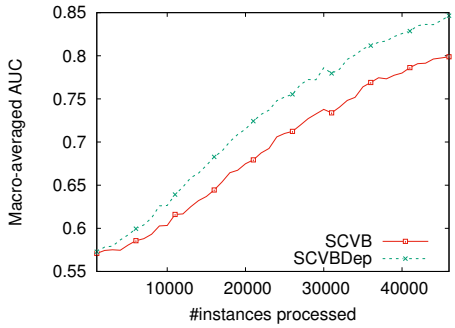


(a) EUR Lex dataset

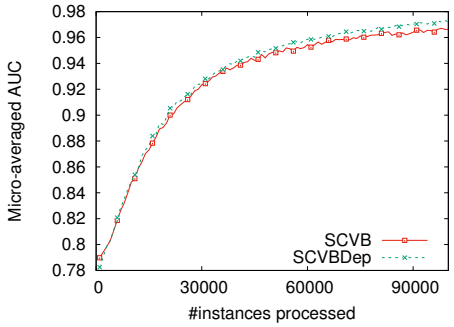
(b) EUR Lex dataset



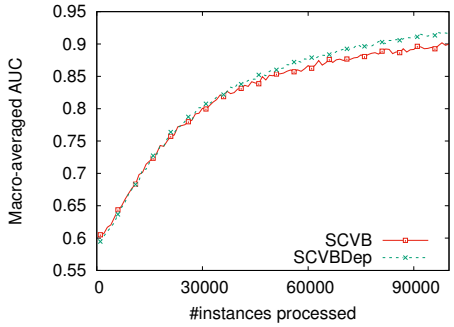
(c) Ohsumed dataset



(d) Ohsumed dataset



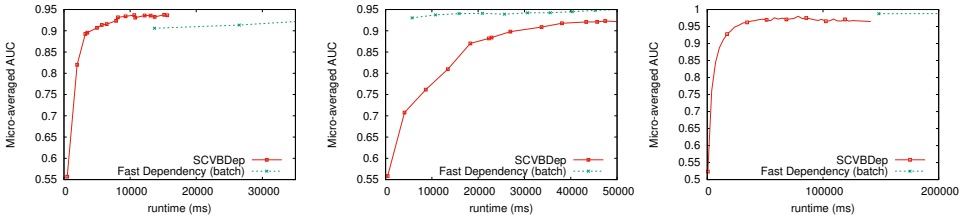
(e) Amazon dataset



(f) Amazon dataset

Figure 3.11: Performance of the online classifier SCVB-Dependency compared to SCVB (micro-/macro-averaged AUC). An initial batch is trained with 1000 instances. Classifiers are then updated with the next batch size instances and tested on the following batch size instances. Results are averaged over ten runs with random orderings of the datasets.

Ohsumed dataset, the batch method converges faster. This is probably the case because the Ohsumed dataset has large feature and label sets relative to the training dataset size. Therefore the overhead introduced by the



(a) `EUR Lex` dataset (b) `Ohsumed` dataset (c) `Amazon` dataset

Figure 3.12: Runtime comparison, 3.12a to 3.12c: training runtime against testset performance in terms of micro-averaged AUC (the testset is fixed). Compared are the online method SCVB-Dep and the batch method Fast-Dep.-LLDA. For the batch method, samples are only taken from a single chain which is why the performance of the online method may be better after convergence.

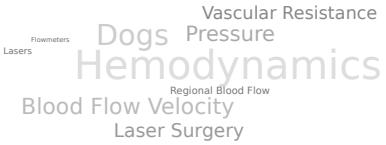
batch updates of the online method slows the model down. With a smaller feature set and/or a larger training dataset, the online method would also be faster in this case. To sum up, the online method is faster to train and converges earlier than the batch method given enough data.

Reversed Fast-Dep.-LLDA

For the reversed Fast-Dep.-LLDA model described in Section 3.5, three topics after training on the `Ohsumed` dataset with 100 topics are shown in Figure 3.13. For each topic the frequent labels are shown as a label cloud and the frequent words are shown as a word cloud. As we can see, many labels are strongly related and the topic clouds might in some cases be more useful descriptions of the dataset content than the label clouds that result from non-reversed Fast-Dep.-LLDA.

3.9 Discussion

A batch method based on Gibbs sampling and an online method based on variational Bayes were introduced. Gibbs sampling generally has the advantage of being unbiased, variational Bayes is faster to converge but biased. Online variational Bayes has been shown to converge to a local optimum of the variational Bayes objective function [27]. For Gibbs sampling many iterations are needed for convergence. In practice, however, it might in some cases be feasible to employ the Gibbs sampling method in an online setting, meaning that each instance is processed only once [14]. Since the labels are predetermined during training, it might not be necessary to perform more than one iteration over the training dataset. However, there is no theoretical guarantee that this is the case. The higher the label cardinality of

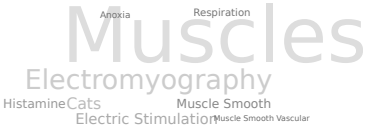


(a) Labels for Topic 1

(b) Words for Topic 1

(c) Labels for Topic 3

(d) Words for Topic 3



(e) Labels for Topic 6

(f) Words for Topic 6

Figure 3.13: Label and Word Clouds for Fast-Dep.-LLDA-Reversed on the `Ohsumed` dataset. The 30 most frequent words and 10 most frequent labels are shown for three exemplary topics.

a dataset, the longer it generally takes for the Gibbs sampler to converge. Therefore, the Fast-Dep.-LLDA online method is preferable in true online settings where a stream of data arrives and it is not possible to store all arriving data.

Different outcomes of the ranking performance were observed as compared to the classification performance. While the ranking performance is mostly better than that of Dep.-LLDA and the ranking performance of all topic modeling methods is much better than that of BR with SVMs, the results are different for the binary classification measures. Here, BR(SVM) clearly emerges as the overall best performing method, except on macro-averaged F-measure. The performance on the macro-averaged F-measure is an indication for the good performance on rare labels due to the learned label dependencies. Potentially, the ranking performance of BR(SVM) could be further improved by postprocessing its prediction outputs. Nevertheless, BR(SVM) is a far more complex method in terms of training runtime. Despite parallelization, the training of BR for the `Amazon` dataset took several days, whereas the training runtime of the topic modeling methods is a matter of several minutes to a few hours on a single core. BR is therefore

more expensive in terms of runtime, but only preferable if the classification performance is the only important factor and computational resources are not taken into account. In summary, the topic modeling methods are more scalable and efficient, ranking performance seems to be better without requiring expensive postprocessing, and most importantly, they may be used in the online setting where real-time analyses are required.

3.10 Conclusion

The main success factor of Dep.-LLDA was identified, namely the separate training of the topic and label level. Based on this, an improved version of the multi-label topic model Dep.-LLDA was developed, called Fast-Dep.-LLDA, which uses a greedy layer-wise training procedure. It was shown that Fast-Dep.-LLDA has theoretical as well as practical advantages. The sampling procedure is consistent with the defined model and heuristics are avoided. In terms of ranking performance, Fast-Dep.-LLDA is superior to BR(SVM). While the label ranking performance is superior to the existing Dep.-LLDA model, the binary classification performance is still competitive. Overall Fast-Dep.-LLDA is easier to implement than Dep.-LLDA, able to handle thousands of labels as opposed to BR and most other multi-label methods, and therefore much more appealing for practical applications. Additionally it can easily be modified so it may be used for analyzing label dependencies, and it generalizes the well-known author-topic model.

Also, an online classifier called SCVB-Dep. was introduced, which is based on the same graphical model as Fast-Dep.-LLDA, but is trained in an online fashion. It was shown that it has a better performance than the non-dependency SCVB and that it converges faster than Fast-Dep.-LLDA during training on large datasets.

Possible future research directions include the following:

1. In real-world applications, the label set is usually not static. New labels may be added over time whereas others could become irrelevant. The capability of adding and removing new labels could be added to Fast-Dep.-LLDA.

2. Streaming data exhibits properties such as concept drift and recurring concepts. For example, a label might become less frequent during winter and more frequent in summer. Such scenarios are not handled by the current model.

3. The frequency of different labels and topics may differ a lot. This is not modeled since the current model only considers symmetric Dirichlet priors. Learning asymmetric priors might improve classification performance.

This last point outlines the idea of the next chapter.

Chapter 4

Nonparametric Multi-label Topic Model

4.1 Introduction

The previous chapter introduced a multi-label topic model, which can be trained on streaming data. This model utilizes two LDAs that are stacked on top of each other. One shortcoming of this model is that the different frequencies of the topics and labels are not modeled, i.e. they are given a symmetric prior. This problem is addressed by the hierarchical Dirichlet process (HDP), which is used to train nonparametric topic models. HDP topic models are nonparametric in the sense that the number of topics is automatically determined from the data. However, their main advantage is the modeling of different topic frequencies thus leading to better representations of the data. In this chapter the idea of the previous chapter is extended to use HDPs instead of standard LDAs.

There are several ways in which HDPs might be employed in the kind of multi-label setting considered here. In the literature there are two models in particular with a similar structure, albeit they are just employed in unsupervised settings. First, there is a variant of nested DPs, called coupled DP mixtures (cHDP), by Shimosaka *et al.* [56]. This model groups the documents into topics in addition to clustering them by labels (or rather sub-topics since the model is unsupervised). cHDP is restricted in that each document belongs to exactly one topic. Second, there is a hierarchical topic model called nonparametric Pachinko allocation model (PAM), which associates a distribution over labels and topics with each document so that each document may belong to several labels and topics (see Fig. 4.2c). This, however, leads to a complex model with a three-level HDP and having to save document-specific distributions over topics as well as labels [36].

Building on the work presented in the previous chapter, I propose a third option that is less complex than option two and does not have the restric-

tion of option one. It is a combination of two two-level HDPs which are not nested as in option one, but rather stacked. This means that the word-tokens are clustered by labels and the labels are further clustered into different topics. Therefore, the model is called stacked HDP (sHDP).

To make the model applicable in large-scale settings a novel Gibbs sampling method is introduced, which is inspired by Li *et al.* [35]. Whereas Li *et al.* reduce the sampling complexity from $O(K)$ to $O(K_w)$, where K_w is the number of topics occurring with word w, this algorithm samples in $O(K_d)$, where K_d is the number of topics per document. Because of the sparsity assumption $K_d \ll K$, the runtime is improved, especially for large datasets, since $K_w \to K$ for $D \to \infty$ (assuming the probability of topic and word occurring together is bounded from below by δ, the probability of the topic occurring at least once is $1 - (1 - \delta)^D \geq 1 - e^{-D\delta} \to 1$ for $D \to \infty$ [35]). In other words, the number of topics occurring with a certain word approaches the total number of topics as the number of documents D increases, whereas the number of topics per document is assumed to be independent of the dataset size.

In summary, the contributions of this chapter are as follows:

1. A new generative topic model called Stacked HDP (sHDP) is introduced that provides a more efficient alternative to related models such as nonparametric PAM and cHDP.

2. A novel sampling method for HDPs is proposed which is not only effective in classification but is applicable in any setting where Gibbs sampling is performed for HDPs and which is furthermore independent of the used language model. This significantly speeds up sampling for HDPs in all application settings.

The chapter is structured as follows. Differences to existing methods are discussed in Section 4.2. Section 4.3 presents the new model and sampling procedure. The sampling method is then improved to be more memory efficient in Section 4.3.2. A detailed analysis of the influence of the hyperparameters on the classification results is provided in Section 4.4.3 and used to provide an effective method of hyperparameter selection. The experiments in Section 4.4 evaluate the classification performance of sHDP on a range of multi-label text datasets and find it to be competitive with the existing Dependency-LDA model. Thereby, a model is provided that is, although worse in classification performance than a neural network, less complex and has better interpretability.

The background for this chapter was covered in chapter 2. In particular, HDPs and the efficient sampling method by Chen *et al.* [16] were introduced in Section 2.1.8. The alias sampling method was introduced in Section 2.1.8. Apart from that the model introduced in this chapter is closely related to the model in chapter 3 and the same background also applies to this chapter.

4.2 Differences to Existing Methods

The main difference between LLDA, Dep.-LLDA and sHDP is that the number of topics in those models is fixed and cannot be learned from the data. Another difference between Dep.-LLDA and sHDP is illustrated in Fig. 4.1. Whereas sHDP samples a topic for each word token, Dependency-LDA only samples topics per label in the document's labelset (see Fig. 4.1).

The partially labeled topic model [48] is not able to share topics between the labels, and has no means of modeling label dependencies. It is also not intended for classification, and therefore not compared to any multi-label classifiers.

The CoL model by Wang *et al.* [70] also models each label as a distribution over latent topics. However, the number of latent topics is fixed here, and it does not incorporate an asymmetric topic prior as HDP does.

A model that does allow topic sharing was proposed by Salakhutdinov *et al.* [55]. In contrast to sHDP, however, this model is not a multi-label model. In this work, it is shown that it is also possible to assign several labels to a document while enabling the sharing of topics. The computational complexity is the same for both models.

An unsupervised topic model called nonparametric PAM based on HDPs that does model topic correlations was proposed by Li [36] (see Fig. 4.2c). The crucial difference between this model and the proposed sHDP is the label-topic distribution. Nonparametric PAM has a separate label-topic distribution for each document, whereas in sHDP it is possible to extract global label dependencies using a single label-topic distribution for all documents. Also, the proposed sampling method was not efficient.

The main difference between cHDP [56] and sHDP is that in cHDP each document may only belong to one super-topic, which makes the model more restrictive. In sHDP each document has a distribution over super-topics, each of which has a distribution over sub-topics. Considering that their method is a variant of *nested* DP models, the proposed model may be viewed as a *stacked* DP model.

Generally, Bayesian hierarchical modeling based on HDPs is a fundamental technique for domains such as image and text modeling, and has previously been used in applications such as image classification [55], data compression [73], and modeling data that evolves over time [50], however, so far it has not been used for multi-label classification.

4.3 Proposed Method

4.3.1 Hierarchical Model

The proposed method sHDP models a potentially infinite number of super-topics z' each of which is associated with a distribution over all sub-topics or

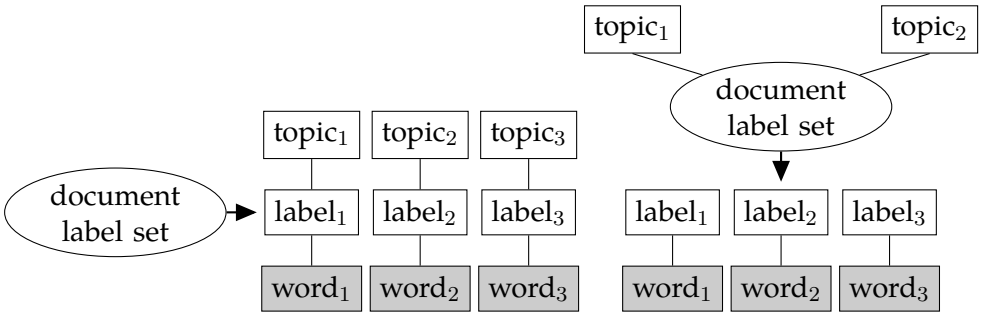
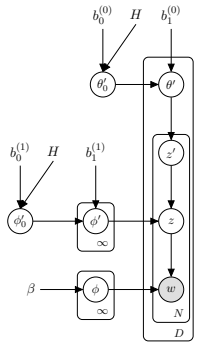


Figure 4.1: Illustration of the difference between Stacked HDP (left) and Dependency-LDA (right). The labels are drawn from the document label set in both cases. Stacked HDP samples one topic for each word/label token, whereas Dependency-LDA samples one topic for each label in the document label set. The white rectangles are sampled variables.

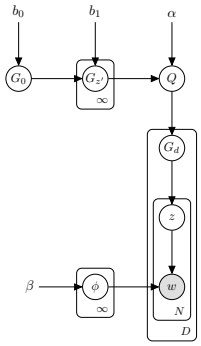
labels. Thus the same sub-topic may appear in multiple super-topics. This allows the modeling of topic correlations. Additionally, sHDP is nonparametric, which allows the number of sub- and super-topics to be automatically determined from the data. Using Gibbs sampling each word-token is associated with a sub-topic and a super-topic that can be sampled independently and that only depend on the variables in their respective Markov-blanket.

The graphical model of sHDP is shown in Fig. 4.2a. The generative process is defined as follows:

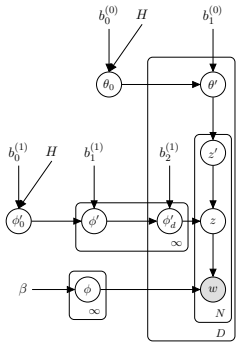
- A distribution θ'_0 over super-topics is sampled from a DP
- A distribution ϕ'_0 over sub-topics is sampled from a DP
- For each super-topic k':
  - a distribution over sub-topics $\phi'_{k'}$ is sampled from a DP with base distribution ϕ'_0
- For each sub-topic k:
  - a distribution over words ϕ_k is drawn from a symmetric Dirichlet distribution
- For each document:
  - a distribution θ' over super-topics is sampled from a DP with prior θ_0
  - For each token in the document:
    - a super-topic z' is sampled from the document specific distribution over super-topics θ'



(a) Stacked HDP consists of two two-level HDPs, one for the topic distributions θ', and one for the label distributions ϕ'.



(b) This extension of Nested HDP also allows for sub-topics to be shared among all super-topics. However, each document has to belong to exactly one super-topic.



(c) Wei Li's nonparametric PAM: In contrast to sHDP, here we have a 3-level HDP consisting of ϕ'_0, ϕ', and ϕ'_d.

Figure 4.2: The graphical model of sHDP compared to two alternative models, the coupled HDP (cHDP) model by Shimosaka *et al.* [56] and the nonparametric PAM model by Wei Li [36]. sHDP is a simplified model with a more effective sampling procedure.

- a sub-topic z is sampled from the distribution over sub-topics $\phi'_{z'}$ associated with super-topic z'
- a word w is sampled from the word-topic distribution ϕ_z associated with sub-topic z

$$\theta'_0|b_0, H \sim DP(b_0, H), \qquad \theta'|b_1^{(0)}, \theta'_0 \sim DP(b_1^{(0)}, \theta'_0)$$
$$z'|\theta' \sim Mult(\theta')$$
$$\phi'_0|b_0, H \sim DP(b_0, H), \qquad \phi'|b_1^{(1)}, \phi'_0 \sim DP(b_1^{(1)}, \phi'_0)$$
$$z|\phi'_{z'}, z' \sim Mult(\phi'_{z'})$$
$$w|\phi_z, z \sim Mult(\phi_z), \qquad \phi \sim Dirichlet(\beta)$$

We can see from the above that the model corresponds to two two-level HDPs "stacked" on top of each other.

The sampling process is divided into two steps: First, z'_i is sampled conditioned on all z'_j with $i \neq j$, and z. Second, z_i is sampled conditioned on all z_j with $i \neq j$, z'_i, and w. In (2.32) to (2.35) this is summarized as $rest$ for brevity. Since ϕ' is sampled from a DP and ϕ is sampled from a Dirichlet distribution, the equations for both steps are slightly different in one term namely P_{wk}.

When no alias-sampling is used, the sampling equations for sampling the sub-topics k are equivalent to (2.32) to (2.36). When sampling the super-topics (2.32) to (2.35) are used, but P_{wk} is now given by:

$$P_{wk} = \sum_{u'} P'(z = w, u = u'|rest)$$

where w in this case corresponds to the sub-topic and k corresponds to the super-topic. P' is calculated using equations 2.32 to 2.35 disregarding the prior term given by 2.36. P_{wk} therefore corresponds to the summed probability mass for sub-topic w given super-topic k.

In this section, sHDP was introduced along with a description of how the variables can be sampled using the standard HDP sampling equations. However, this model is not efficient especially if the number of sub-topics or super-topics is high. Therefore, a more efficient sampling procedure is proposed in the next section.

4.3.2 Alias-sampling for the HDP

The idea of Li *et al.*'s alias-sampling (see Section 2.1.8) is employed of storing a stale part of the probability distribution and sample from it in $O(1)$ correcting the difference with a Metropolis-Hastings acceptance step. However, in contrast to the original alias-sampling, the hierarchical structure of HDPs is exploited. Recall that the conditional probability for topic k is given by:

$$\begin{aligned} P(z = k|rest) = P(z = k, u = 0|rest)+ \\ P(z = k, u = 1|rest) + P(z = k, u = 2|rest). \end{aligned}$$

The last term is usually sparse since it is only non-zero for all topics that already have a table in the corresponding restaurant. The second part is dense, but changes rather slowly since the overall topic distribution changes much slower than the topic distribution within a document or label. Therefore, instead of dividing the distribution according to the language model term $\frac{N_{wk}+\beta}{\sum_{w'}(N_{w'k}+\beta)}$, it is divided according to the table indicator u, thus yielding a sampler that runs in $O(K_d)$ instead of $O(K_w)$ (in case of a standard two-level HDP).

More formally, the proposal distribution $q(k)$ over topics k is constructed as a combination of the stale distribution q_w and the fresh distribution p_{jw}, where w is a word type or a sub-topic when sampling sub-topics or super-topics respectively, and j is a restaurant where there is one for each document when sampling super-topics or one for each super-topic when sampling sub-topics. The normalization terms are defined as $P_{jw} = \sum_k p_{jw}(k)$ and $Q_w = \sum_k q_w(k)$ and the proposal distribution is defined as follows:

$$q(k) := \frac{p_{jw}(k) + q_w(k)}{P_{jw} + Q_w}$$

In contrast to the original alias-sampling, the stale distribution q_{jw} is defined as the distribution over all topics and a table indicator of 0 or 1:

$$q_{jw}(k) := P(z = k, u = 0|rest) + P(z = k, u = 1|rest)$$

Accordingly, the fresh distribution p_{jw} is defined as the distribution over all topics that exist in restaurant j and a table indicator of 2:

$$p_{jw}(k) := P(z = k, u = 2|rest)$$

The Metropolis-Hastings acceptance ratio for the transition from topic s and table indicator u_s to topic t and table indicator u_t is given by $min(1, \pi)$, where

$$\pi = \frac{P(z = t, u = u_t|rest)}{P(z = s, u = u_s|rest)} \cdot \begin{cases} P_{jw}p_{jw}(s), & \text{if } u_s = 2 \\ Q_w q_w(s), & \text{otherwise} \end{cases} \cdot \begin{cases} \frac{1}{P_{jw}p_{jw}(t)}, & \text{if } u_t = 2 \\ \frac{1}{Q_w q_w(t)}, & \text{otherwise} \end{cases}$$

Improving the sparse sampler

The described method reduces the sampling complexity to $O(K_j)$, but, as can be inferred from equations 2.33 and 2.35, q_{jw} depends on document j. This means the global topic distribution has to be saved separately for every document. The same is true for the alias-sampler by Li *et al.* [35], which puts a restriction on the size of the used datasets since a topic distribution has to be saved for every single document. In this section, a method is proposed that instead only uses a single global distribution.

The main idea is to assume for each topic that it does not exist in the document and save the resulting distribution q_w^e for an empty pseudo document e. This can be understood as replacing Equation 2.35 with Equation 2.33. In case a topic is sampled from this distribution that exists in the current document, it is discarded and a new one is drawn from the same distribution.

$$\tilde{p}_{jw}(k, u') := P(z = k, u = u'|rest)\mathbb{1}[n_{jk} > 0] \ , \tag{4.1}$$

where $\mathbb{1}[n_{jk} > 0]$ is one if the number of tokens in document-restaurant j associated with topic k is at least one and zero otherwise. Accordingly, the normalization sum is $\tilde{P}_{jw} = \sum_k \sum_u \tilde{p}_{jw}(k, u)$.

An amount Δ_j needs to be subtracted from the normalization sum Q_w which is different for each document j and accounts for the topics that are present in document j and would be rejected if drawn from distribution q. It is called the discard mass Δ and defined as follows:

$$\Delta_j := \sum q_k^e \mathbb{1}[n_{jk} > 0] \tag{4.2}$$

Δ_j can also be computed in $O(K_j)$ time and therefore does not increase the overall computational complexity. The modified normalization sum is accordingly given by $\tilde{Q}_{jw} = Q_w - \Delta_j$, where $Q_w = \sum q_w^e$.

The difference to the true distribution needs to be corrected using Metropolis-Hastings (MH). The modified MH acceptance ratio is given by:

$$\pi = \frac{P(z=t, u=u_t|rest)}{P(z=s, u=u_s|rest)} \cdot \begin{cases} \tilde{P}_{jw}\tilde{p}_{jw}(s), & \text{if } n_{js} > 0 \\ Q_w q_w^e(s), & \text{otherwise} \end{cases}. \tag{4.3}$$

$$\begin{cases} \frac{1}{\tilde{P}_{jw}\tilde{p}_{jw}(t)}, & \text{if } n_{jt} > 0 \\ \frac{1}{Q_w q_w^e(t)}, & \text{otherwise} \end{cases} \tag{4.4}$$

The algorithm for a single HDP topic model is summarized in Alg. 8. For each word in each document a topic z_i and a table indicator u_i is sampled by doing H MH-iterations. If no stored samples are left, new samples are produced from the alias table (lines 6–10). Then $\tilde{p}$ and Δ_d are computed by iterating over the document topics (line 11). Now, it is decided whether to sample the topic from $\tilde{p}$ or to use a stored sample from q^e (line 13). If a sample from q^e is chosen, it needs to be rejected in case the chosen topic exists in the document (lines 16–18). Finally the MH-acceptance ratio is computed to decide if the sample is accepted or not (line 20–23). For stacked HDPs, lines 4–25 need to be repeated for the sampling of the labels, otherwise the algorithm is the same.

Complexity

The complexity of Li *et al.*'s alias-sampling is $O(K_w)$, where K_w is the number of topics that occur with word w. In contrast, the proposed algorithm runs in $O(K_j)$, where K_j is the number of topics that occur in restaurant j. E.g., in a standard two-level HDP there is one restaurant for each document meaning the sampling complexity is reduced to $O(K_d)$, where K_d is the number of topics per document. This is the case since the computation of P_{jw} has this complexity as well as the sampler for p_{jw}. Sampling from q_w is done in amortized $O(1)$ so this does not add to the overall complexity. The computation of π is also done in $O(1)$.

Alias-sampling for Stacked HDP

The sampling method is applicable in Stacked HDP at the sub-level as well as the super-level. At the sub-level the prior probability for the sub-topics is expected to change slowly relative to the probability of the sub-topics inside a given super-topic restaurant. Therefore it is sufficient to iterate over the sub-topics that already exist for a given super-topic, whereas samples from the prior distribution are drawn in amortized $O(1)$. When sampling a sub-topic, it is decided whether to choose an existing table or a new one. In the

Algorithm 8 Train Stacked HDP Topic Model

for each iteration $j = 1 \dots I$ **do**
  for each document d in D **do**
    for each word w_i in d **do**
      $result \leftarrow (z_i, u_i)$
      for Metropolis-Hastings iteration $h = 1 \dots H$ **do**
        if no more samples stored for type w_i **then**
          recompute q_w^e (Equation 2.33)
          compute alias table A_w from q_w^e
          store k samples from A_w
        end if
        compute $\tilde{p}$ and Δ_d (Equations 4.1 and 4.2)
        $r \leftarrow$ uniform random number
        if $r < \tilde{P}_{dw_i}/(\tilde{P}_{dw_i} + \tilde{Q}_{dw_i})$ **then**
          sample topic and table indicator z_i, u_i from $\tilde{p}$
        else
          repeat
            $z_i, u_i \leftarrow$ stored topic and table indicator
          until topic z_i does not occur in document d
        end if
        compute acceptance ratio π (Equation 4.3) for transition $result \rightarrow (z_i, u_i)$
        if sample accepted **then**
          $result \leftarrow (z_i, u_i)$
        end if
      end for
      $z_i, u_i \leftarrow result$
    end for
  end for
end for

first case, it is only iterated over existing sub-topics, in the second case, a sample is drawn from the saved distribution of the sub-topics. At the super-level the prior probability for the super-topics is expected to change slowly relative to the probability of a super-topic in a given document. To place a customer at an existing table it is only necessary to iterate over the topics that are present in the document. When a new table is opened a sample can be drawn from the slowly changing stale prior distribution in $O(1)$. The overall complexity of sHDP is reduced from $O(Y + K)$ to $O(Y_k + K_d)$, where Y and K are the number of labels and topics respectively, Y_k is the number of labels occurring with topic k and K_d is the number of topics occurring in document d.

Advantages of the proposed sampling method as opposed to the origi-

nal one by Li *et al.* are:

1. It is independent of the language model used. Thus it is also applicable when using e.g. a Poisson Dirichlet process for the language model to capture the power law distribution of words.

2. It exploits the hierarchical structure of the model. This makes it applicable in models with more levels with increasingly slower changing distributions.

3. It has computational complexity of amortized $O(K_d)$, which may be much less than $O(K_w)$ if the dataset is large.

4. The space complexity of the improved sparse sampler is much lower since only one global distribution needs to be stored whereas Li et al.'s method stores a different distribution q_{jw} for each document j.

4.3.3 Supervised Training

In supervised training, the sub-topics for a certain document are restricted to its labels. This means the exact label combination is known and we have partial knowledge of the result during sampling. If the actual probability estimates are used during training, the Gibbs sampler has a tendency to get stuck in local minima and less frequent labels are not sampled for many iterations. To alleviate this problem, a uniform document-label distribution is used during training.

In order to achieve decent prediction performance the label variables z are sampled according to the following equation during training:

$$P_{supervised}(z_i = k, u_i = u|rest) \propto \frac{N_{wk} + \beta}{\sum_{w'}(N_{w'k} + \beta)} \frac{P(z_i = k, u_i = u|rest)}{\sum_{u'} P(z_i = k, u_i = u'|rest)}$$

This means the document-topic probability is equal for all topics during training, similar to the model in chapter 3. The main difference is that here, the probabilities of the different table indicators are retained.

4.3.4 Prediction

The probability of label y in document d given z' is estimated as

$$\frac{1}{N_d} \sum_{w_i \in d} \Big(\frac{N_{w_i y} + \beta}{\sum_{w'}(N_{w'y} + \beta)} \sum_{u'} P(z = y, u = u'|rest) \sum_{u'} P(z'_i, u = u'|rest) \Big),$$

where N_d is the number of words in document d.

Ten chains are run for each classifier, taking 1500 samples from each chain to get an estimate (for bookmarks and EUR Lex, only 300 samples per

chain are taken because of runtime concerns). These estimates are averaged again over the different chains to receive the final estimate.

For Dependency-LDA, the original publication uses an estimate of θ for the prediction. However to make the comparison fair and because it was found to deliver better results, an average over the conditional probability for $z = y$ is also used in this case:[1] $\frac{1}{N_d} \sum_{w \in d} \frac{N_{wy}+\beta}{N_{.y}+|V|\beta}(n_{dy} + \alpha')$. For Dependency-LDA only 100 samples per chain were taken since this method converges faster (see Section 4.4.4 for the experimental justification).

4.4 Experiments

To evaluate the sHDP, it is compared to five state-of-the-art multi-label classifiers on the ranking measures micro- and macro-averaged AUC and the binary measures micro- and macro-averaged F-Measure on six publicly available multi-label text datasets, which are considered large as compared to the majority of multi-label datasets.

4.4.1 Algorithms

The multi-label classifiers for comparison are:

- Binary Relevance (BR) with a linear SVM as a base classifier (BR(SVM)): BR learns one separate SVM for the prediction of each label. In the case of large datasets BR is one of the strongest available multi-label classifiers [49] and the LIBLINEAR SVM is especially well suited for large datasets with many features. Parameter C was optimized in the range $[10^{-3}, \ldots, 10^{3}]$.

- A neural network (NN) using advancements from recent deep learning methods [43]. On large datasets it was shown to consistently outperform BR. The training progress is monitored on a separate validation dataset and training is stopped when the ranking loss does not improve anymore.

- Dependency-LDA (Dep.-LLDA), a multi-label topic model introduced by Rubin *et al.* [54] that is especially well suited for large-scale multi-label datasets with many labels.

- PD-Sparse by Yen *et al.* [75]: See Section 3.8.3.

- FastXML by Prabhu and Varma [46]: See Section 3.8.3.

Table 4.1: number of labels, attributes, instances, cardinality and average number of tokens per document for the used datasets

number of	delicious	bookmarks	rcv1v2	bibtex	Reuters-21578	EUR Lex
labels	983	208	101	159	90	3,955
attributes	500	2,150	47,144	1,836	18,637	5,000
instances	16,105	87,856	23,149	7,395	10,789	19,314
cardinality	19.020	2.028	3.184	2.402	1.130	5.300
av.#tokens	19	127	122	70	70	846

4.4.2 Datasets

Six publicly available multi-label text datasets were used. The available train-test splits were used for `delicious` [64], `bookmarks` [29], `bibtex` [29], and `Reuters-21578, Distribution 1.0`. For `Rcv1v2` [34] only the training set was used and 10% were separated for testing, as the available testset is extremely large. For `EUR Lex` [39] the last 10% were used for testing. Additionally, a cosine-normalized TF-IDF transform was done on the features for `Rcv1v2`, `Reuters-21578`, and `EUR Lex`, which further improved the BR and NN results. However, the topic modeling methods cannot process real-valued features, so they were given the original datasets. The statistics for all used datasets are summarized in Table 4.1. All instances without labels were removed from the testsets, since they only distort the results for ranking measures.

4.4.3 Parameter Settings

The topic number is learned from the data. However, as was noted in previous literature [11], it is beneficial to initialize the model with a larger number of topics and subsequently allow only the removal of topics. Therefore a topic number of 100 was chosen for all datasets, a number that appeared to be large enough for the given text corpora. Additionally, $\beta = 0.01$ was chosen, a standard default value that is frequently used in this area. The nonparametric model was run for 1000 iterations since it is assumed to take longer to converge. On EUR Lex only 100 iterations were used. Dependency-LDA converges quickly so the number of iterations was set to 100 since higher values do not lead to any improvement.

The hyperparameters in HDPs determine how similar a distribution should be to its prior distribution. The results of the initial experiments are shown in Table 4.2. These results show that the variability in the results on micro-averaged AUC is mainly due to the setting of the parameter for

[1] See Papanikolaou *et al.* [44] for a formal justification of this approach.

Table 4.2: Preliminary experiments concerning the hyperparameter settings of b_1 at the super-topic- and sub-topic-level respectively for sHDP using micro-averaged AUC. The best result for each dataset is highlighted in bold. The datasets and train-test splits are described in Section 4.4.2.

sub-topic-b_1 \ super-topic-b_1	**0.01**	**0.1**	**1.0**	**10.0**	estimated
			bibtex		
0.01	0.7607	0.7820	0.8430	0.8688	0.8238
0.1	0.7665	0.7938	0.8492	0.8688	0.8245
1.0	0.7635	0.7921	0.8524	0.8695	0.8284
10.0	0.7714	0.7911	0.8487	0.8664	0.8308
estimated	0.7676	0.7939	0.8526	**0.8722**	0.8281
			reuters		
0.01	0.9667	0.9769	0.9861	**0.9872**	0.9821
0.1	0.9692	0.9787	0.9859	0.9872	0.9821
1.0	0.9686	0.9775	0.9868	0.9864	0.9811
10.0	0.9698	0.9742	0.9825	0.9833	0.9795
estimated	0.9705	0.9782	0.9848	0.9863	0.9843
			bookmarks		
0.01	0.7730	0.7910	0.8118	0.8311	0.8010
0.1	0.7775	0.7847	0.8150	**0.8317**	0.8025
1.0	0.7803	0.7877	0.8148	0.8286	0.8008
10.0	0.7783	0.7851	0.8129	0.8280	0.8022
estimated	0.7791	0.7893	0.8124	0.8291	0.7874
			EUR Lex		
0.01	0.9039	0.9122	0.9197	0.9280	0.9164
0.1	0.9051	0.9103	0.9208	0.9289	0.9171
1.0	0.9074	0.9117	0.9206	0.9289	0.9190
10.0	0.9092	0.9131	0.9211	0.9281	0.9180
estimated	0.9081	0.9139	0.9229	**0.9309**	0.9189
			Rcv1v2		
0.01	0.9662	0.9716	0.9806	0.9806	0.9778
0.1	0.9667	0.9731	**0.9818**	0.9811	0.9777
1.0	0.9674	0.9730	0.9804	0.9807	0.9773
10.0	0.9674	0.9712	0.9808	0.9793	0.9776
estimated	0.9670	0.9734	0.9803	0.9802	0.9783
			delicious		
0.01	0.7813	0.7782	0.7653	0.7563	0.7655
0.1	0.7845	0.7784	0.7616	0.7566	0.7632
1.0	0.7857	**0.7991**	0.7682	0.7537	0.7615
10.0	0.7845	0.7913	0.7638	0.7539	0.7626
estimated	0.7820	0.7847	0.7652	0.7537	0.8059

Table 4.3: Results for all datasets on four different measures. The best value for each dataset is highlighted in bold. The rank is given in brackets.

Datasets	NN	BR(SVM)	sHDP	Dep.-LLDA	PD-Sparse	FastXML
			Micro-averaged AUC			
Reuters	**0.9949**	0.9867	0.9937	0.9927	-	-
Rcv1v2	**0.9915**	0.9867	0.9859	0.9812	-	-
bibtex	**0.9470**	0.9163	0.9270	0.9330	-	-
delicious	**0.9208**	0.8876	0.8207	0.8234	-	-
bookmarks	**0.9388**	0.8984	0.8804	0.8700	-	-
EUR Lex	DNF	0.9099	0.9535	0.9565	-	-
av. rank	1.0000	2.8333	2.8333	2.8333	-	-
			Macro-averaged AUC			
Reuters	0.9837	**0.9896**	0.9884	0.9799	-	-
Rcv1v2	0.9731	**0.9762**	0.9756	0.9686	-	-
bibtex	**0.9276**	0.8984	0.9228	0.9242	-	-
delicious	**0.8302**	0.7806	0.7002	0.6996	-	-
bookmarks	**0.9193**	0.8720	0.8514	0.8548	-	-
EUR Lex	DNF	0.8687	0.9090	0.9228(1)	-	-
av. rank	1.8000	2.1667	2.6667	3.0000	-	-
			Micro-averaged F-Measure			
Reuters	**0.8050**	0.1914	0.2478	0.2394	0.1560	0.6812
Rcv1v2	**0.7853**	0.7200	0.7416	0.7243	0.1722	0.4097
bibtex	0.1078	0.0350	0.0491	0.0372	0.3961	**0.4592**
delicious	0.1975	0.1795	0.1278	0.1101	0.0468	**0.3683**
bookmarks	**0.3753**	0.3671	0.2572	0.2493	0.0159	0.3337
EUR Lex	DNF	**0.3871**	0.2668	0.3558	0.0421	0.1138
av. rank	1.6000	3.5000	3.3333	4.0000	5.1667	2.6667
			Macro-averaged F-Measure			
Reuters	0.2314	0.0189	0.0154	0.0188	0.0292	**0.3002**
Rcv1v2	0.4504	0.5108	**0.5235**	0.4988	0.0558	0.1888
bibtex	0.0321	0.0177	0.0152	0.0171	0.3188	**0.3337**
delicious	0.0344	0.0144	0.0168	0.0176	0.0140	**0.1196**
bookmarks	**0.2843**	0.2567	0.1444	0.1523	0.0090	0.2241
EUR Lex	DNF	0.1487	0.0679	**0.1572**	0.0040	0.0137
av. rank	2.4000	3.1667	4.1667	3.5000	4.6667	2.5000

the upper-level HDP. Therefore, b_1 was estimated at the sub-topic-level HDP during training [16] and kept fixed for testing. For the super-level HDP, b_1 was determined by grid search since, as Table 4.2 shows, estimating the parameter from the data does not lead to optimal classification results. 1000 random documents were used from the training dataset for validation and the procedure was repeated 5 times over the values $b_1 \in \{0.1, 1.0, 5.0, 10.0, 15.0, \text{estimate}\}$. b_0 was estimated for both HDPs.

4.4.4 Results

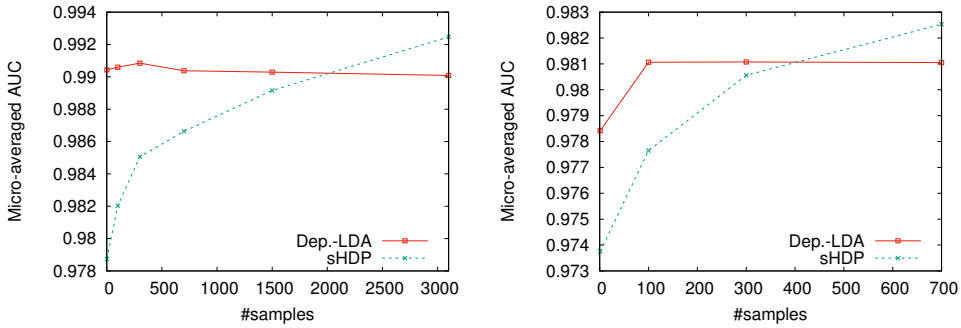
Convergence Speed

The convergence speed of sHDP and Dependency-LDA are compared in Fig. 4.3. While Dependency-LDA mostly takes 100 or fewer samples to arrive at a performance close to the maximum, sHDP needs more samples, sometimes several thousand. However, in three out of four cases the final performance of sHDP is better than that of Dependency-LDA. This means, while sHDP takes longer to converge, it may nevertheless lead to improved results over Dependency-LDA.

Multi-label classification

The results for all measures are presented in Table 4.3. PD-Sparse and FastXML are only compared on the binary measures because they do not produce a full ranking over the labels as an output. Generally, the neural network classifier produces the best prediction results, followed by FastXML which is better than the neural network on two datasets. For micro-averaged AUC sHDP is on par with BR(SVM) and Dependency-LDA according to the average rank. On macro-averaged AUC it is better than Dependency-LDA but worse than BR. On this measure, the neural network is outperformed on two datasets. The best performance is achieved for micro-averaged F-measure where sHDP outperforms both BR(SVM) and Dependency-LDA. However, it is particularly bad on macro-averaged F-measure, except for the Rcv1v2 dataset. On this dataset sHDP has the best overall performance, even compared to the neural network classifier. It is to be expected that sHDP fares best on the micro-averaged F-measure since micro-averaged measures are indicative of the performance on frequent labels. The infrequent labels have a low prior probability in this model and are therefore predicted less often.

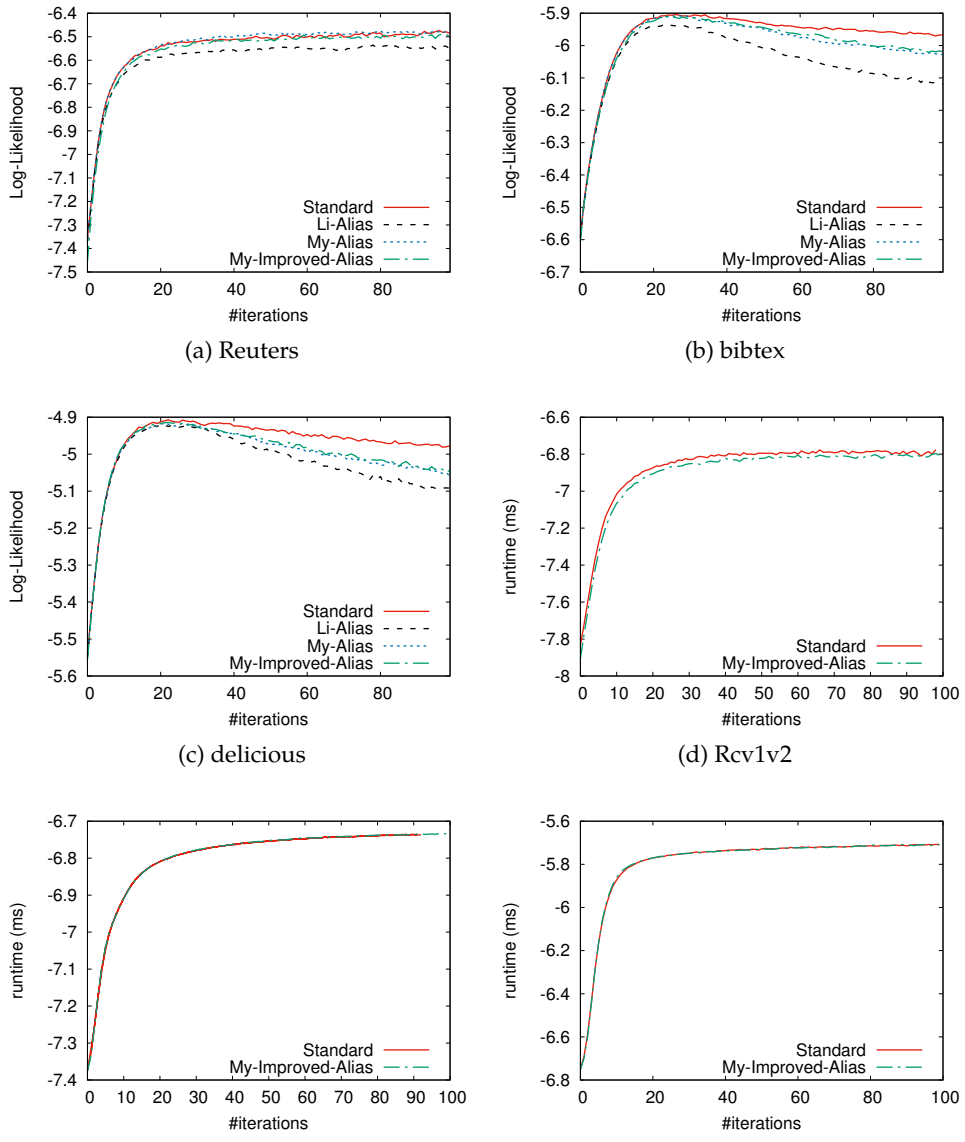
The main method of comparison is Dependency-LDA since it is most closely related to sHDP. It is also based on topic modeling and has the advantage of interpretability of learned label dependencies. BR and NN are different in that they may assign one feature to several labels depending on different weights, whereas in Gibbs sampling one word is always assigned to a single label. Especially on the `delicious` and `bookmarks` datasets this



(a) Reuters (b) Rcv1v2

(c) bibtex (d) delicious

Figure 4.3: Performance on micro-averaged AUC with respect to the number of samples used for evaluation. For sHDP all hyperparameters were estimated from the data. The convergence of Dependency-LDA is faster, but for three out of four datasets, the final evaluation performance of sHDP is better than that of Dependency-LDA.



(a) Reuters (b) bibtex

(c) delicious (d) Rcv1v2

(e) bookmarks (f) EUR Lex

Figure 4.4: The testset log-likelihood is compared for the different sampling methods for HDP. The improved sampling method is overall closest to that of the standard sampler, showing that a single global distribution improves convergence.

seems to be an advantage as is reflected in the gap in results between the two topic modeling methods and the other methods, BR, NN and FastXML. PD-Sparse only produces good results on the bibtex dataset. On all other datasets, sHDP has better results.

Sampling efficiency

To compare the effectiveness of the sampling method, the log-likelihood was plotted on the testset during unsupervised training with 1000 topics in Fig. 4.4 and the runtime per iteration in Fig. 4.5. Experiments were run on a single core. For this experiment four HDP topic models were implemented. All four models use the same Java-framework, only the code for the sampling step is implemented differently.

1. The standard HDP implementation uses the original method as described by Chen *et al.* [16].

2. The improved alias-sampling is described by Li *et al.* [35].

3. The third implementation uses the proposed alias-sampling method utilizing table indicators.

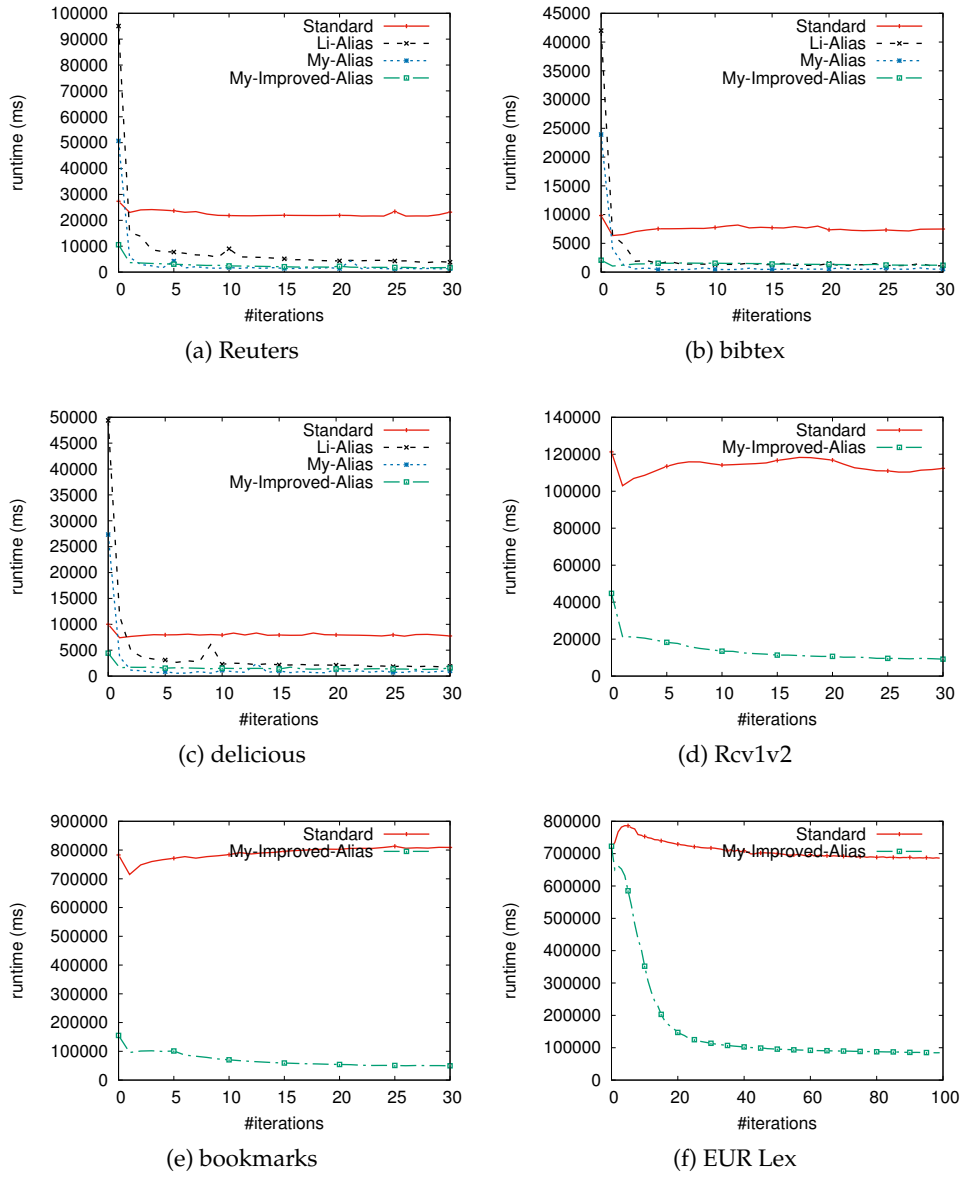
4. The improved alias sampling only uses one global distribution that is shared among all documents for the alias sampler.

Methods 2 and 3 are only compared on the three smallest datasets since they incur a large memory usage on the other datasets. Considering memory requirements, only methods 1 and 4 are feasible on a standard desktop PC with 16 GB RAM for large datasets.

The testset log-likelihood of the improved alias sampling method is shown in Fig. 4.4. It converges almost as well as that of the standard method. As Fig. 4.5 shows, the alias-sampling needs significantly less time per iteration than the standard method. Summing up, it was shown on two representative datasets that the new sampling method greatly improves runtime at unchanged log-likelihood.

4.5 Conclusion

To conclude, a new generative topic model called Stacked HDP (sHDP) was introduced that is more efficient than related models. Furthermore, a novel sampling method for HDPs was proposed which is not only effective in classification but is applicable in any setting where Gibbs sampling is performed for HDPs. This significantly speeds up sampling for HDPs in all application settings.



(a) Reuters (b) bibtex

(c) delicious (d) Rcv1v2

(e) bookmarks (f) EUR Lex

Figure 4.5: The runtime per iteration is compared for the different sampling methods for HDP. The alias samplers have a much lower runtime per iteration than the standard sampler.

While sHDP is presented in the context of multi-label classification in this chapter, it is a highly interpretable and flexible model that allows the extraction of specific label correlations and may also be used in unsupervised settings. The appeal of sHDP lies not mainly in its classification performance. Rather, it closes a gap in the literature for methods that are less complex, more efficient and better interpretable than neural networks, provide a flexible model that is easily adaptable to different settings and datasets, and is efficient enough to allow the application of HDPs in large-scale hierarchical models.

Possible future work includes:

1. sHDP could be extended to the streaming setting in a way similar to the method in chapter 3. The ability of the nonparametric model to add new topics could then be used to allow adjusting the number of topics over time, adapting to changes in the input.

2. Another line of future work is the exploration of the unsupervised version of sHDP. Optimization of hyperparameters becomes more important in the unsupervised setting since the parameter space is less constrained and different optimization methods could be explored.

The next chapter lays the groundwork for this by applying the proposed sampling method in an unsupervised streaming setting.

Chapter 5

Online Nonparametric Topic Model

5.1 Introduction

In the previous chapter I introduced a multi-label topic model utilizing a novel sampling method for HDPs. This sampling method may also be used in unsupervised topic models. In this chapter I show how it is applied in a hybrid variational-Gibbs topic model. This hybrid model combines the advantages of both variational and Gibbs sampling methods to yield an efficient model that can be trained online.

The two main algorithms for training LDA topic models are Gibbs sampling and variational Bayes. While Gibbs sampling is an unbiased method, it takes long to converge on large datasets. To make Gibbs sampling online, one has to use particle samplers which are rather inefficient. Variational Bayes on the other hand may be combined with stochastic gradients to be trained online and converges faster. However, it is biased.

Hybrid methods have gained popularity in recent years, especially in deep neural networks where black box variational inference is an efficient training algorithm [30]. Sampling can be used to approximate the gradient in variational Bayes which leads to a second source of stochasticity (in addition to random choices of data subsets). In previous work this was applied to *parametric* topic models by Hoffman *et al.* [26]. In this work, a sparse update scheme was proposed that allowed to do variational updates for only the topic-word-combinations that were actually sampled. Experiments showed that the method is faster especially for large topic numbers. Additionally the convergence is improved as compared to other variational methods since the variational distribution considered is not completely factorized but considers each document as a unity. An efficient variant of this algorithm was recently proposed that takes advantage of the sparsity in topic distributions during sampling [38]. They used a fast Gibbs sampling

method to further speed up sampling for parametric LDA. Thereby they exploited sparseness in the variational updates as well as the document-topic distributions. Hybrid algorithms have the combined advantages of a reduced bias of the variational method and a faster convergence as compared to pure Gibbs sampling, as well as the possibility of online training through stochastic gradient estimation.

Another extension of the original work by Hoffman *et al.* was proposed by Wang and Blei [68], who developed a similar method for the nonparametric HDP. The main contribution in this work is the development of a truncation-free variational method that allows the number of topics to grow. This is made possible by the sampling step which does not depend on a truncation as in pure variational methods. In contrast to this work, their method builds on Teh's direct assignment sampler [59], whereas the method proposed in this chapter relies on the more advanced table indicator sampler proposed by Chen *et al.* [16]. Also, the proposed method starts out with the maximum number of topics, and subsequently removes topics (by letting their expected counts approach zero over time). This was found to be beneficial in previous work [11]. Finally, their method does not take advantage of any kind of sparse sampling.

This chapter shows how a hybrid sparse method can be applied to *nonparametric* topic models. Section 5.2 introduces the proposed hybrid method. The hybrid method uses the efficient sampling method introduced in chapter 4.3.2. In this chapter, it is further simplified to be even more time efficient by showing that the probability of opening a new table with a topic which is not present in the current document is always lower or equal to the probability of opening a new table with the same topic if it does occur in the current document (Section 5.3.1). The experiments in Section 5.7 show that the new hybrid method converges better than the purely variational topic modeling method as well as the Gibbs sampler.

The main contributions of this chapter are as follows:

1. A hybrid HDP topic model is introduced that is based on the table indicator sampling scheme instead of the direct assignment sampler.

2. A doubly sparse sampling scheme, utilizing an efficient sampling method for HDPs is introduced.

3. Experiments show that the method is competitive in log-likelihood as well as runtime with existing approaches.

The background for this chapter was explained in chapter 2. The sampling method by Chen *et al.* [16] is introduced in Section 2.1.8. Online variational inference is described in Section 2.1.6 and Gibbs sampling is explained in Section 2.1.3.

5.2 Proposed Method – Hybrid Variational-Gibbs

This section describes how the sampling method by Chen *et al.* [16] (see Section 2.1.8) can be used to construct a hybrid Variational-Gibbs training algorithm for the HDP. The proposed algorithm is online since it is based on stochastic gradient ascent [27, 26, 19]. This means the model can be continuously updated with new batches of data.

As described in Section 2.1.6 on variational inference and following Hoffman *et al.* [26], the natural gradient of the ELBO with respect to the variational topic-word distribution $\tilde{\beta}$ is defined by

$$\mathbb{E}_q[N_{dkw}] + \frac{1}{D}(\beta - \tilde{\beta}_{kw}), \tag{5.1}$$

where N_{dkw} is the expected count for topic k and word w in document d.

To evaluate the expectation in this equation one would need to evaluate all possible topic configurations for each document. For using stochastic gradient ascent however, an approximation is sufficient. This is where Gibbs sampling comes into play. By taking samples from the distribution $q^\star$ the expectation in the above equation can be approximated.

$$q^\star(z_{di} = k|z_{-i}) \propto \exp\{\mathbb{E}_{q(-z_d)} \log(p(z_d|b_1, G_0)p(w_d|z_d, \phi))\}, \tag{5.2}$$

where $-z_d$ denotes all topic indicators z except the ones for document d. This distribution is difficult to normalize since one would have to consider all possible topic configurations z_d. However, one can easily sample from it and estimate the variational Dirichlet parameters as follows [26]:

$$\tilde{\beta}_{kv} = \beta + \sum_d \sum_i \mathbb{E}_q[\mathbb{1}[z_{di} = k]\mathbb{1}[w_{di} = v]], \tag{5.3}$$

where the expectation is approximated by the samples from $q^\star$.

In contrast to Hoffman *et al.*, the proposed hybrid model has an additional variational distribution over the topics G_0. This is the global topic prior. The global variational distribution for G_0 and the mixture components ϕ is

$$q(G_0, \phi|\tilde{\gamma}, \tilde{\beta}) = \prod_k q(G_{0_k}|\tilde{\gamma})q(\phi_k|\tilde{\beta}_k), \tag{5.4}$$

where $\tilde{\gamma}$ and $\tilde{\beta}$ are Dirichlet parameters.

The variational Dirichlet parameter for the global topic distribution is analogously estimated as follows:

$$\tilde{\gamma}_k = \gamma + \sum_d \sum_i \mathbb{E}_q[\mathbb{1}[z_{di} = k]\mathbb{1}[u_{di} = 1|u_{di} = 0]], \tag{5.5}$$

where γ is a hyperparameter, and $\mathbb{1}[u = 1|u = 0]$ is one if the table indicator u is either zero or one, which means that a new table is being opened, and otherwise zero.

The expectations in Equations 5.3 and 5.5 can be estimated by sampling from $q^\star$ which is given by the following set of equations (compare to Equations 2.32–2.35, differences are highlighted in bold):

If the topic is new for the root restaurant (table indicator is zero):

$$q^\star(z_{di} = k, u = 0|z_{-i}) \propto \frac{(b_1 + aM_d)(b_0 + aK)}{(b_1 + N_j)(b_0 + \sum_{\boldsymbol{k'}} \tilde{\boldsymbol{\gamma}}_{\boldsymbol{k'}})} \mathbf{exp}\left(\mathbb{E}[\boldsymbol{log\phi_{wk}}]\right) \quad (5.6)$$

If the topic is new for the base restaurant (e.g. a document), but not for the root restaurant (table indicator is one):

$$q^\star(z_{di} = k, u = 1|z_{-i}) \propto \frac{b_1 * \tilde{\boldsymbol{\gamma}}_{\boldsymbol{k}}^2}{(b_1 + N_j)(\tilde{\boldsymbol{\gamma}}_{\boldsymbol{k}} + 1)\left(\sum_{\boldsymbol{k'}} \tilde{\boldsymbol{\gamma}}_{\boldsymbol{k'}} + b_0\right)} \mathbf{exp}\left(\mathbb{E}[\boldsymbol{log\phi_{wk}}]\right) \quad (5.7)$$

If the topic exists at the base restaurant and a new table is opened (table indicator is one):

$$\begin{aligned} q^\star(z_{di} = k, u = 1|z_{-i}) \propto \frac{b_1 + aM_d}{b_1 + N_j} \frac{S^{n_{jk}+1}_{m_{jk}+1,a}}{S^{n_{jk}}_{m_{jk},a}} \frac{m_{jk} + 1}{n_{jk} + 1} \\ \frac{\tilde{\boldsymbol{\gamma}}_{\boldsymbol{k}}^2}{(\tilde{\boldsymbol{\gamma}}_{\boldsymbol{k}} + 1)\left(\sum_{\boldsymbol{k'}} \tilde{\boldsymbol{\gamma}}_{\boldsymbol{k'}} + b_0\right)} \mathbf{exp}\left(\mathbb{E}[\boldsymbol{log\phi_{wk}}]\right) \end{aligned} \quad (5.8)$$

If the topic exists at the base restaurant and an old table is chosen (table indicator is two):

$$q^\star(z_{di} = k, u = 2|z_{-i}) \propto \frac{S^{n_{jk}+1}_{m_{jk},a}}{S^{n_{jk}}_{m_{jk},a}} \frac{n_{jk} - m_{jk} + 1}{n_{jk} + 1} \mathbf{exp}\left(\mathbb{E}[\boldsymbol{log\phi_{wk}}]\right) \quad (5.9)$$

In the above equations, the number of tables M_k (Equations 2.32 and 2.33) is substituted by the global variational parameter $\tilde{\gamma}$. $\exp(\mathbb{E}[log\phi_{wk}])$ is expensive to compute, since $log(\phi_{wk}) = \Psi(\tilde{\beta}_{wk}) - \Psi(\sum_w \tilde{\beta}_{wk})$, where $\Psi(\cdot)$ is the digamma function. Following Wang *et al.* [68] and Li *et al.* [38] $\frac{\tilde{\beta}_{wk}+\beta}{\sum_{w'}(\tilde{\beta}_{w'k}+\beta)}$ is used instead. The remaining variables are the local counts equivalent to the counts in Equations 2.32 to 2.35.

Updating variational parameters $\tilde{\beta}$ and $\tilde{\gamma}$ for one minibatch M is done as follows, where the counts for one minibatch are scaled by $\frac{|D|}{S|M|}$ for S sampling iterations, to arrive at the expectation for the whole corpus and ρ is a parameter between zero and one.

$$\tilde{\beta}_{kw} = (1 - \rho_t)\tilde{\beta}_{kw} + \rho_t \left(\beta + \frac{|D|}{S|M|} \sum_{d \in M} N_{dkw}\right) \quad (5.10)$$

$$\tilde{\gamma}_k = (1 - \rho_t)\tilde{\gamma}_k + \rho_t \left(\gamma + \frac{|D|}{S|M|} \sum_{d \in M} \sum_{n \in d} \mathbb{1}[u_{dn} = 1|u_{dn} = 0]\right) \quad (5.11)$$

Summing up this section, the table indicators from Chen *et al.*'s sampling method for the HDP are used to be able to approximate the global topic distribution from minibatch samples. This yields an online algorithm for the HDP topic model.

5.3 Doubly Sparse Updates for Online HDP

Having introduced the hybrid variational algorithm based on the table indicator sampling scheme, now a doubly sparse sampling method for the nonparametric topic model is proposed. This is similar to Li *et al.*'s [38] method for the parametric topic model and would not be possible for the direct assignment sampler. The method is doubly sparse because there are two aspects that make updates sparse. First, the online update of the global parameters is sparse since it is only necessary to take into account those words that actually occur in the minibatch. This is due to the nature of the hybrid update that uses Gibbs sampling for the local updates which leads to sparse update matrices as opposed to variational updates that are always dense. Second, an additional sparsity assumption was added in the local document-topic updates of the Gibbs sampler. Conditioned on the assumption that the number of topics in the document is smaller than the total number of topics, this leads to more efficient local updates. Therefore the doubly sparse hybrid method for the HDP does sparse updates not only at the global level but also on the local document level.

The sparse sampling method is described in detail in Section 4.3.2. This sampling method reduces the sampling complexity from $O(K_w)$ to $O(K_d)$, where K_w is the number of topics for word w and K_d is the number of topics for document d. Further, the sampling method was improved to also reduce the space complexity so that only one global topic distribution has to be saved instead of one for each document.

5.3.1 Further Improving the Sparse Sampler.

The efficient sparse sampling method for HDPs introduced in Section 4.3.2 is now further improved by taking advantage of the special setting of this hybrid method where the global topic distribution does not change during the sampling of one minibatch, i.e. only the document counts are updated as is the case during evaluation of a trained model. This case also arises for the method proposed here because the global distributions are not updated during the sampling of one minibatch. The calculation of the discard mass could then be avoided, if for all topics occurring in the document, $\tilde{p}_{jw}(k, u = 1) > q_k^e$. If this is the case, we can compute $\tilde{p}' = \tilde{p} - q^e$ and be guaranteed that $\tilde{p}' >= 0$. To see that it is indeed the case it needs to be shown that $\frac{S_{m+1}^{n+1}}{S_m^n} \frac{m+1}{n+1} \geq 1$, which is equivalent to showing that

Algorithm 9 Train Hybrid HDP Topic Model

while not converged **do**
  $M \leftarrow$ get minibatch from D
  compute q^e (Equations 4.3.2,5.6,5.7) and $Q = \sum q^e$
  $A_w \leftarrow$ computeAliasTable($\frac{q^e_w}{Q_w}$) for each word w (see Algorithm 10)
  for document $d \in M$ **do**
    $z_d \leftarrow$ initialize randomly
    for iteration $i = 1, \ldots, S + B$ **do**
      for token $n = 1, \ldots, N_d$ **do**
        $z_{dn}, u_{dn} \leftarrow$ Sample(A, w_{dn}) (Algorithm 11)
      end for
      if $i > B$ **then**
        Save sample
      end if
    end for
  end for
  update $\tilde{\beta}$ and $\tilde{\gamma}$ (Equations 5.10 and 5.11)
end while

$$\frac{S^{n_{jk}}_{m_{jk}}}{S^{n_{jk}+1}_{m_{jk}+1}} \leq \frac{\binom{n_{jk}}{m_{jk}}}{\binom{n_{jk}+1}{m_{jk}+1}}$$

This inequality is intuitive, considering that Stirling numbers grow faster than binomial coefficients. The hereby improved method is more elegant, however, it can only be applied when the global counts are fixed since otherwise stale distributions q might occur for which the inequality does not hold.[1]

The whole algorithm is summed up in Algorithm 9. For each minibatch, the dense distributions q^e are computed for each word that occurs in the minibatch. Alias tables are computed for these distributions (see Algorithm 10) to be able to sample from them in $O(1)$. For each document d in the minibatch, the topics z_d are sampled using Algorithm 11 and stored after a burn-in period of B iterations. The stored samples are finally used to update the global variational distributions.

5.4 Algorithm Description

In this section, the algorithm is described in more detail. The model is trained using Algorithm 9. For each minibatch, the dense distribution q^e_w

[1]The improved method can also be applied if $a > 0$, i.e. a hierarchical Poisson-Dirichlet topic model is used. In this case q needs to be divided by $(b_1 + M_d)$ and remultiplied with this factor when subtracting q from p.

Algorithm 10 computeAliasTable(q)

1: $r \leftarrow 1/q$.length {the average bucket size}
2: $L = H = A = \emptyset$
3: **for** $k = 1, \ldots, K$ **do**
4: **if** $q(k) \leq r$ **then**
5: {put all probabilities lower than the average bucket size into one stack}
6: L.push$(k, q(k))$
7: **else**
8: {put all probabilities higher than the average bucket size into the other stack}
9: H.push$(k, q(k))$
10: **end if**
11: **end for**
12: **while** $L \neq \emptyset$ **do**
13: {take one entry from each stack, a small and a large one}
14: $(l, pl) \leftarrow L$.pop()
15: $(h, ph) \leftarrow H$.pop()
16: $A \leftarrow A \cup (l, h, pl)$ {store the two topics and the smaller probability in the alias table}
17: **if** $ph + pl - r > r$ **then**
18: H.push$((h, ph + pl - r)$ {If the remaining probability is higher than average, put it into one stack}
19: **else**
20: L.push$((h, ph + pl - r)$ {If the remaining probability is lower than average, put it into the other stack}
21: **end if**
22: **end while**
23: **return** A

and the sum Q_w for each word w are computed (line 3). Here, it is assumed that an empty pseudo document e is used which enables to compute q^e only once for the whole minibatch instead of once for each document. Now, the alias tables A_w are computed for each word using distributions q^e_w (line 4).

The computation of an alias table according to Walker [66] is given in algorithm 10. Here, r is calculated as the average bucket size (line 1). First the distribution values are distributed to two sets L and H (lines 3–11). Each value (with the corresponding index) is added to L if it is equal or smaller than r or to H if it is larger. Now, the alias table is calculated by repeatedly taking one value which is smaller or equal than r and one which is larger from L and H, respectively (lines 14–15). Then, it is calculated how much probability is left after taking the smaller value and adding from the larger value to fill the bucket to be of size r (line 17). The left-over probability is

Algorithm 11 Sample(A, w)

1: compute $\tilde{p}$ and $\tilde{P} = \sum \tilde{p}$ (Equation 4.1)
2: compute Δ (Equation 4.2)
3: $i = -1$, $u \leftarrow 1$
4: sample $r \sim$ Uniform$(0, \tilde{P} + \tilde{Q})$
5: **if** $r < \tilde{P}$ **then**
6: {Sample a topic already present in the document}
7: **while** $r > 0$ **do**
8: $i \leftarrow i + 1$
9: $t \leftarrow i/2$ {t is the topic index}
10: $u \leftarrow 2 - (i \mod 2)$ {u is the table indicator}
11: $r \leftarrow r - \tilde{p}_{j,w}(t, u)$
12: **end while**
13: **else**
14: {Use a sample from the alias table for word w}
15: **while** t is not new **do**
16: $t \leftarrow$ sample from Alias A_w
17: **end while**
18: **end if**
19: **return** t,u {topic and table indicator}

added back into L or H depending on whether it is larger or smaller than r (lines 17–21).

After the alias table has been built, one can sample from it by generating one sample from a uniform distribution and selecting the corresponding entry from the alias table. Each entry in the alias table stores two topic indices and one probability p_a. To select one of the topics another sample is drawn from a uniform distribution and check whether or not it is smaller than p_a. Thus, it is possible to draw from the distribution by generating two samples from a uniform distribution without having to iterate over the whole distribution.

After all alias tables have been computed, one starts iterating over the documents of the current minibatch (Algorithm 9, line 5). $S + B$ iterations are done over each document and samples are saved after the burn-in iterations B (lines 11-13). For each token in the current document, a topic and table indicator are sampled (line 9) using Algorithm 11. Finally, the global distributions are updated (line 16).

Alg. 11 describes the sampling of one topic and table indicator for word w using the corresponding alias table A_w. First, the sparse distribution $\tilde{p}$ is computed by iterating over the topics that exist in the current document (line 1). At the same time the discard mass Δ can be calculated (line 2). Now it is checked whether to choose a topic from the sparse bucket or the dense

bucket (lines 4–5). If the dense bucket is chosen, a sample is taken from the alias table as previously described (lines 13–18). Then, it is checked whether the sampled topic exists in the document and sampling is continued until it is a new topic (line 15–17). If the sparse bucket is chosen, one iterates over $\tilde{p}$ to determine the topic and table indicator (lines 7–12).

5.5 Datasets

Four publicly available datasets were used and preprocessed as follows:

1. `BioASQ`:
   This dataset consists of paper abstracts from the PubMed database. It was made available for the BioASQ competition, a large-scale semantic indexing challenge [60]. The 500,000 most recent documents were separated, plus 10,000 documents as a separate testset. After stopword removal, the 20,000 most frequent features were kept.

2. `Enron`:
   The `Enron` dataset consists of ca. 500,000 emails and is available at `https://www.cs.cmu.edu/~./enron/`[31]. The header was removed, the emails were tokenized, stopwords were removed and the 20,000 most frequent features were kept. 10,000 documents were randomly separated as a testset.

3. `NIPS`:
   This dataset is available in a preprocessed format from the UCI Machine Learning Repository [45]. It has 5,812 documents and 11,463 features and is the second smallest dataset that was used. It consists of NIPS conference papers published between 1987 and 2015. In comparison to the other datasets, the individual documents are large. 1000 documents were randomly separated as a testset.

4. `KOS`:
   The 3430 blog entries of this dataset were originally extracted from `http://www.dailykos.com/`, the dataset is available in the UCI Machine Learning Repository `https://archive.ics.uci.edu/ml/datasets/Bag+of+Words`. The number of features is 6906.

5.6 Evaluation

The models were evaluated on the per-word-log-likelihood according to Heinrich [24]:

Table 5.1: Statistics for the datasets used in the experiments, $|D|$ train and test: the number of documents in the train- and testset, respectively, $|V|$: size of vocabulary

Dataset	$\|D\|$ train	$\|D\|$ test	$\|V\|$
Enron	507,401	10,000	20,000
BioASQ	500,000	10,000	20,000
NIPS	4,811	1,000	11,463
KOS	2930	500	6906

$$\frac{\log p(w|M)}{|n_d|} = \left(\sum_{w \in d} n_{dw} \log \left(\sum_{k=1}^{K} \phi_{kw} \theta_{kw} \right) \right) / |n_d|, \tag{5.12}$$

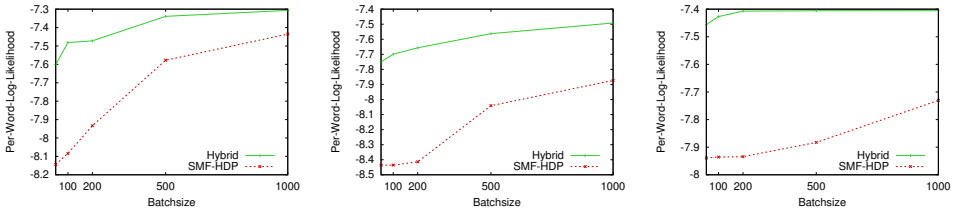
where n_{dw} is the number of times word w occurs in document d, K is the overall number of topics, and ϕ are the model parameters. θ are the document specific parameters that need to be estimated using the model. In this case, the sampler is run with a fixed point estimate of parameters ϕ, whereas θ is estimated analogously to the training procedure with S samples that are saved after B burn-in iterations.

5.7 Experiments

5.7.1 Mean-field vs. Hybrid Approach.

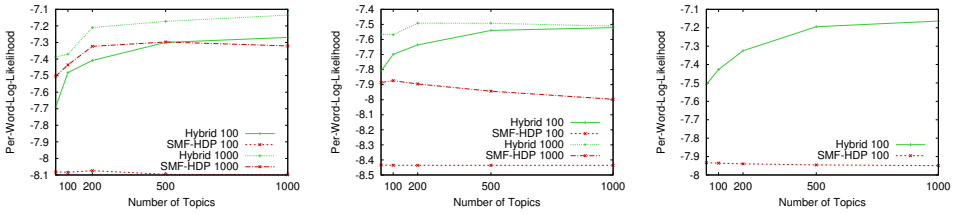
First of all the hybrid approach is compared to Wang *et al.*'s SMF-HDP [69] on the three largest datasets. It can be noticed that the performance of SMF-HDP heavily depends on the batchsize. Small batchsizes lead to a much worse performance (see Figure 5.1). The same observation was made by Wang and Blei [68]. SMF-HDP starts with the maximum topic number and then reduces the number of topics. In the given experiments, often only a handful of topics remained for small batchsizes. Wang and Blei hypothesized that this is due to the algorithm being strongly dependent on the initialization and not being able to add topics occurring in later batches that had not been present from the start. This behavior is problematic, especially in the online setting, where it is not guaranteed that the first batch contains all the topics. The proposed hybrid method is more robust and better suited to settings where small batch sizes are a requirement.

Second, the two algorithms are compared for different settings of the truncation for the number of topics (50, 100, 200, 500, 1000). The results are shown in Figure 5.2. While the truncation does not seem to influence the performance of SMF-HDP for small batchsizes, the hybrid method has an



(a) Enron (b) BioASQ (c) NIPS

Figure 5.1: Effect of batchsize on the log-likelihood after 1000 updates. The truncation was set to 100 topics.



(a) `Enron` (b) `BioASQ` (c) `NIPS`

Figure 5.2: Effect of truncation on the log-likelihood after 1000 updates. The batchsize was set to 100 and 1000 documents as mentioned in the plot labels. For `Enron` only a batchsize of 100 was used since higher batchsizes lead to a large increase in runtime. The hybrid method has a higher log-likelihood for all settings.

improved performance for higher topic numbers on all three datasets. It can therefore be seen, that it is not necessary to start with one topic and add more topics subsequently as was suggested by Wang and Blei [68]. Agreeing with the observations in previous work [11], it can be concluded that it is beneficial to start out with the maximum number of topics and subsequently reduce it. Overall, the hybrid method has a higher log-likelihood for all truncation settings.

5.7.2 Gibbs Sampling vs. Hybrid Approach.

Since training and evaluation of the Gibbs sampler take too long on the large datasets, the smaller `KOS` dataset was also included in the experiments. The convergence of the Gibbs sampler is compared to the convergence of the hybrid method by measuring the testset per-word-log-likelihood after each iteration over the full dataset for the Gibbs sampler, and evaluating the hybrid method after each batch. Figure 5.3a shows the performance of four different methods trained with a truncation to 100 topics. It can be seen that the two hybrid methods converge much faster initially than the Gibbs sam-

plers. Comparing the sparse and the original sampler, the sparse sampler is worse in the beginning since it does not sample from the true distribution, but manages to catch up and even surpass the original sampler due to its faster sampling[2].

Figure 5.3b shows the performance for a truncation to 1000 topics. It can be seen that here the difference in log-likelihood between the sparse and the original sampling method is much bigger. This is because the number of topics per document K_d does not grow that much when the number of total topics is increased. Therefore, while for small topic numbers the differences might be negligible in practice, for higher topic numbers, the sparse sampling method is preferable.

For the `NIPS` dataset, the difference between 100 and 1000 topics is even bigger (Figure 5.3c and 5.3d). `NIPS` has long documents which means that it has more topics per document on average. With only 100 topics, it is possible that almost all topics are present in the document. Therefore, the original sampler is faster than the sparse sampler for 100 topics, but not for 1000 topics, where it is the other way around. The Gibbs samplers have barely even started to converge in the first 1000 seconds where the convergence for the hybrid methods is far ahead.

5.7.3 Supervised Variant

This thesis has introduced the parametric LLDA model (Section 2.3.1), proposed an extension that includes dependencies (Chapter 3) and a nonparametric multi-label model in Chapter 4. However, the HDP model itself has not been applied as a classifier. In this section, the application of HDP and hybrid HDP in the multi-label setting is investigated. As a first step, the batch Gibbs sampling HDP is evaluated. Then, the hybrid HDP algorithm is compared in the online multi-label setting. In both cases the HDP model is compared to the corresponding parametric model.

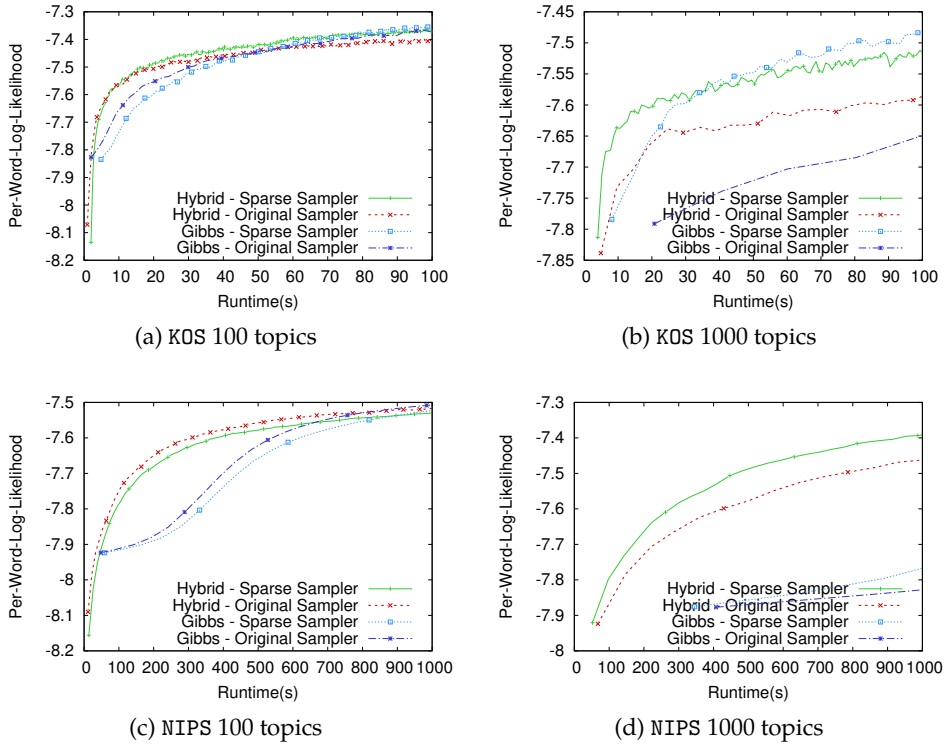
Labeled HDP

HDP can be made supervised in the same way as LDA: by assigning one topic to each label. Analogously to LLDA, the modification of HDP for multi-label classification is called Labeled HDP (LHDP). LHDP allows to take different label frequencies into account. Since the number of labels is fixed, a truncated HDP is used.

Figure 5.4 shows the results for six different standard multi-label datasets[3] in terms of micro-averaged AUC for different settings of the hyper-

[2]Note that the implementation of the hybrid method uses the sparse sampling method, but does not use the sparse updating introduced by Hoffman *et al.* [26]. Therefore, a further speedup is possible.

[3]See Chapter 4 for more information on the datasets.



(a) KOS 100 topics

(b) KOS 1000 topics

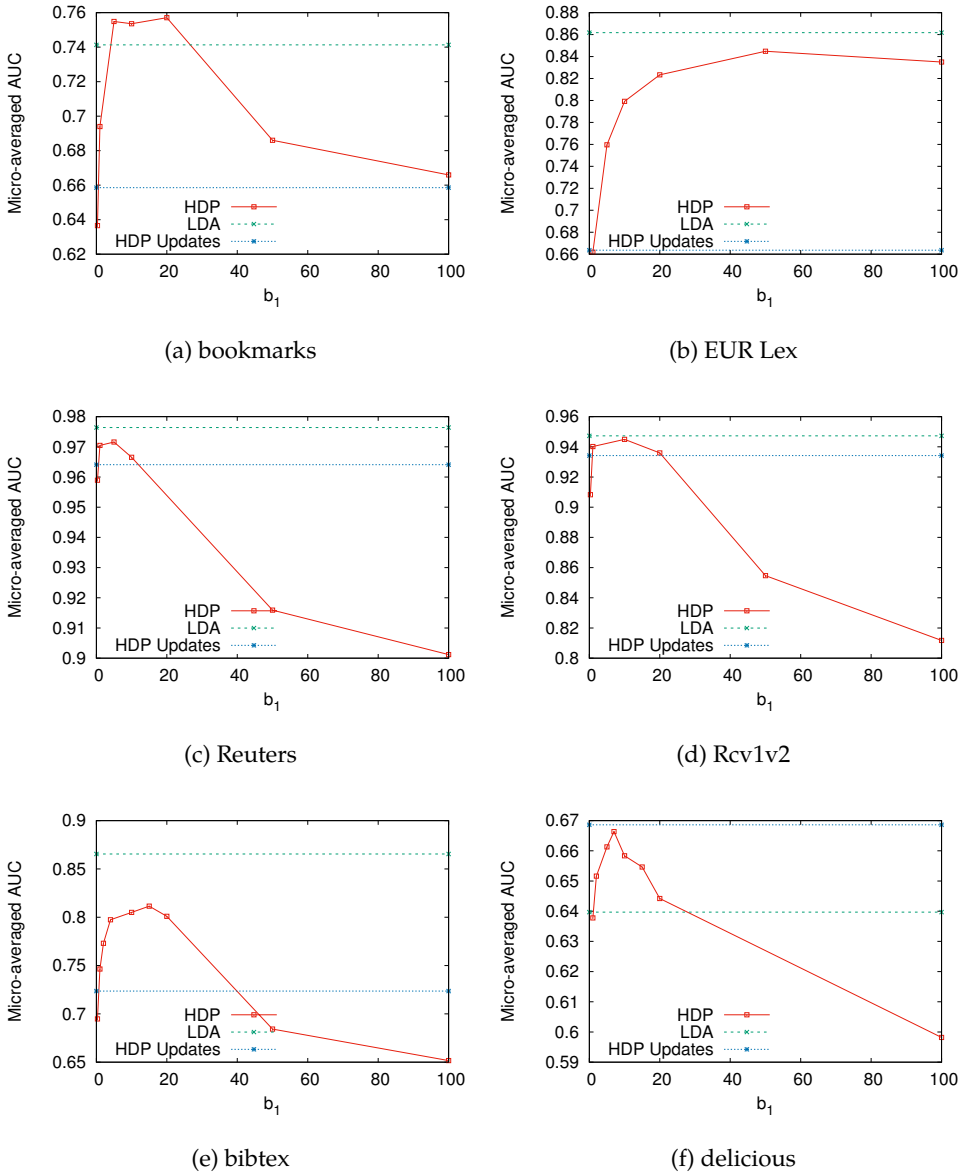
(c) NIPS 100 topics

(d) NIPS 1000 topics

Figure 5.3: Comparison of runtime with 100 and 1000 topics respectively. The hybrid methods were trained with a batchsize of 100 documents. Performance was evaluated on a separate testset. The hybrid method with the sparse sampling algorithm converges faster than the other methods.

parameter b_1. Compared are a labeled version of HDP, labeled LDA and a version of HDP, where the hyperparameter b_1 is updated automatically.

For the bookmarks and delicious datasets there are settings of b_1 where LHDP outperforms standard LLDA. Automatically updating the hyperparameter is advantageous in some cases (e.g. Rcv1v2, delicious) and less advantageous in other cases. The model that automatically updates b_1 is worse than LLDA in all datasets except delicious. It can therefore be inferred that, while LHDP is sometimes able to achieve better results than LLDA in multi-label classification, the correct hyperparameter setting is important for good results and finding it is still an open problem since every dataset has different properties.



(a) bookmarks (b) EUR Lex

(c) Reuters (d) Rcv1v2

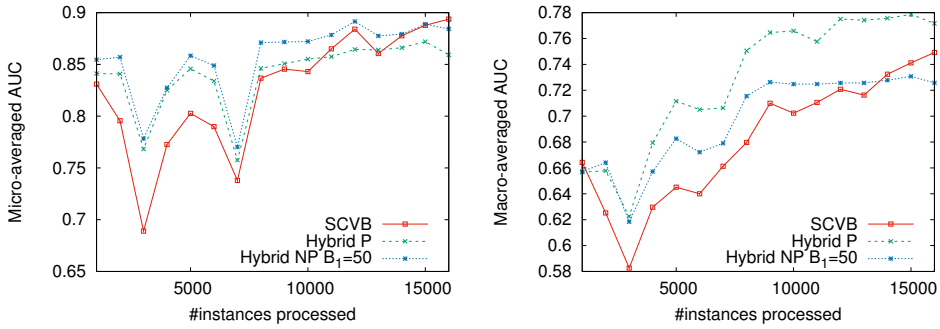
(e) bibtex (f) delicious

Figure 5.4: Ranking performance with respect to parameter b_1. LDA is the result of standard Labeled LDA, HDP Updates is the result of Labeled HDP with hyperparameter optimization.

Labeled Hybrid HDP

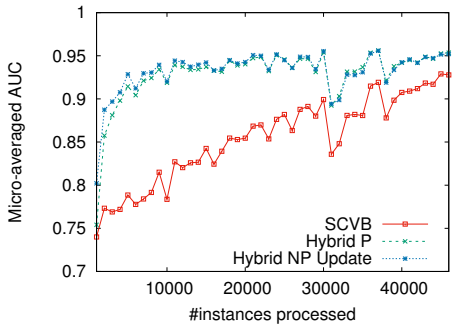
Figure 5.5 shows the micro- and macro-averaged AUC for one run over the `EUR Lex`, `Ohsumed` and `Amazon` datasets[4]. These datasets are larger and all

[4]See Chapter 3 for more information on the datasets.

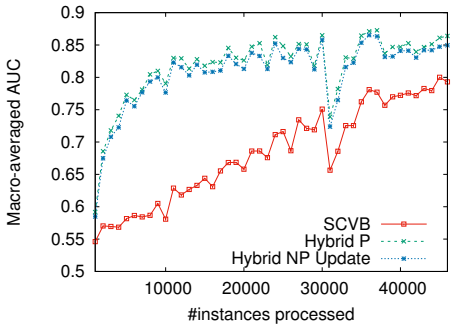


(a) EUR Lex dataset

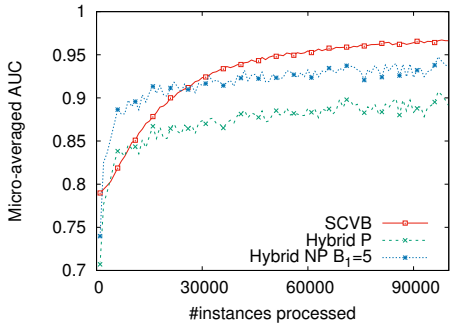
(b) EUR Lex dataset



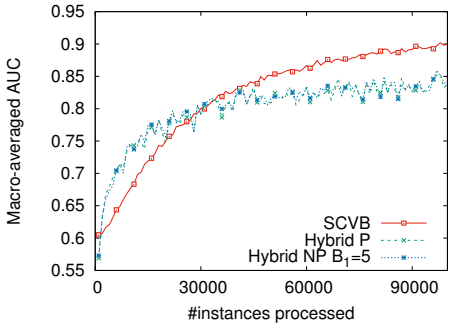
(c) Ohsumed dataset



(d) Ohsumed dataset



(e) Amazon dataset



(f) Amazon dataset

Figure 5.5: Performance of the online hybrid labeled HDP (Hybrid NP) classifier to the parametric hybrid LDA (Hybrid P) and SCVB on micro- and macro-averaged AUC. An initial batch is trained with 1000 instances. Classifiers are then updated with the next batch size instances and tested on the following batch size instances.

have thousands of labels as compared to the datasets used in the previous section that only have hundreds of labels. Compared are three different online classifiers in the same way as in Chapter 3. First, the SCVB method

(Section 2.1.6, Chapter 3) is used as a parametric baseline and is a purely variational method. Second, two versions of the hybrid Variational-Gibbs classifier are compared, the parametric and the nonparametric version. As can be seen, the hybrid methods both converge faster in the beginning of training than the SCVB classifier. This can be explained with the reduced bias of the hybrid method. The nonparametric hybrid method was run with hyperparameters $b_1 = 50$ for `EUR Lex` since the experiments in Section 5.7.3 indicated this as a good value, $b_1 = 5$ for `Amazon` and automatic updates for `Ohsumed`. For micro-averaged AUC the nonparametric method generally outperforms the parametric method. For macro-averaged AUC the parametric method is better. This is because the nonparametric method models label frequencies and thus has a tendency to prefer frequent labels over infrequent ones. Since macro-averaged AUC is a measure that indicates performance on rare labels, it is to be expected that the nonparametric method would perform worse in this case.

For the `Amazon` dataset, the hybrid methods initially converge faster than SCVB, but get worse later on in the training process. Since both the parametric and the nonparametric hybrid method show this behavior, it is most likely due to the sparse updates of the hybrid methods. `Amazon` is the largest dataset with the most labels, which explains why the behavior is not visible with the other two datasets. Most likely, this could be alleviated by increasing the number of samples taken to find a balance between sparseness and taking the whole label distribution into account.

Overall, there seems to be a slight advantage of the nonparametric method in these large-scale experiments (at least on micro-averaged AUC). Since the previous section found that the parametric method was best in most cases, it can be inferred that nonparametric methods fare best on larger datasets where the number of labels is high. While this can only be seen as a preliminary experiment, it shows that, given the right hyperparameters, the nonparametric method is able to perform well in the supervised setting, especially on frequent labels as compared to the parametric method which performs better on rare labels.

5.8 Conclusion

To conclude, this chapter introduced a hybrid sparse Variational-Gibbs nonparametric topic model that can be trained online on large-scale or streaming datasets. Experiments on three large-scale datasets as well as one smaller dataset were conducted. The method was found to be superior to the purely variational Bayes mean field approach in per-word log-likelihood. Additionally, it is more robust to different settings of the batchsize. Compared to the pure Gibbs sampler it converges faster with improved log-likelihood.

The supervised variant was found to be effective, especially for the prediction of frequent labels on large-scale datasets, although the choice of hyperparameter remains an open problem.

Possible future work includes:

1. The model could be applied to hierarchical topic models with more levels.
2. The model could be adapted to streaming settings with concept drift where the global distribution changes over time.
3. The hyperparameter estimation for the supervised method could be improved.

The next chapter introduces a topic model for streaming settings that is able to deal with concept drift.

Chapter 6

Online Topic Model for Tracking News

6.1 Tracking News Related to the Refugee Crisis

Monitoring the news is an important application area for topics models. However, thousands of news articles are published every day, topics are changing fast, new topics emerge and others disappear. Therefore, there is a need for models that help to keep track of these changes to help analyze the situation, identify critical developments and ultimately make decisions and react based on this information. Due to the large amount of articles that are published every day, it is advantageous to have an online method that can directly process a stream of data without having to revisit old documents from the past. Such a method is introduced in this chapter.

Although, this chapter is restricted to using data from news articles, the method also has other potential applications. For example, another important area is the identification of emerging topics in the scientific literature to be able to establish new research projects or business endeavors that are in line with the state of the art [1]. On the whole, the proposed method is applicable in all settings where a large amount of data arrives in a stream.

The effectiveness of the proposed model is evaluated on documents that are related to the refugee crisis. Europe has been witnessing a large movement of immigrants and refugees from Africa and Middle East in recent years. Since the start of the main arrival wave in August 2015, the refugee crisis has been in the spotlight of the media, and an increasing number of events are reported. There have been and still are heated and polarized debates concerning the refugee crisis. The implications of this crisis are extensive and complex, however, little has been done by data mining experts to understand and interpret the media related to the crisis [18].

The goal of this work is to analyze the media in Germany, as an example for one of the highly affected European countries, to address the follow-

ing questions: "What are the main concerns of each party or news source? How does the perception evolve over time? How is the perception influenced by certain events? How similar are different parties and sources in this aspect?". As a result, the first method presented in this thesis is a topic model, which may be used to monitor the news over time to discover how certain topics change and evolve over time. The proposed model is based on the standard LDA topic model (see Section 2.1.2) and trained using variational inference (see Section 2.1.4), although it can just as well be trained with Gibbs sampling or any other training method.

The main contributions of this chapter are as follows:

1. A new online topic model is introduced that tracks topics over time whereby the overall theme of each topic stays consistent.

2. The proposed model is evaluated on a dataset related to the refugee crisis in Germany, showing that it can effectively track topics over time.

The chapter is organized as follows. First, the background of the method is introduced in Section 6.2, which is based on two existing approaches. Second, a new approach combining the advantages of both approaches is proposed. Finally, experiments on the dataset of texts from German media are described in Section 6.4.

6.2 Existing Online Topic Models

There are two broad areas of topic modeling approaches that discover how topics change over time. One analyzes the whole dataset at once, i.e. it processes the data as a batch (e.g., Blei and Lafferty [7]). The second area is the one relevant to this work. Here, the development of topics over time is discovered by processing the data as a stream of minibatches, meaning that each batch of data is viewed only once.

Recall from Section 2.1.2 that the generative process for LDA is given as follows:

$$\phi \sim \text{Dir}(\beta), \theta \sim \text{Dir}(\alpha), z \sim \text{Mult}(\theta), w \sim \text{Mult}(\phi_z) \tag{6.1}$$

For each topic k, a multinomial distribution ϕ_k over words is drawn from a Dirichlet distribution with parameter β. For each document d, a distribution over topics θ is drawn from a Dirichlet with parameter α. For each word w_{di} in document d a topic indicator z_{di} is drawn from the multinomial distribution θ. Finally the word w is drawn from the multinomial distribution $\phi_{z_{di}}$ associated with the chosen topic.

The online LDA (OLDA) method proposed by AlSumait *et al.* [1] separates the data into different time slices $D = \{D^1, \ldots, D^{t-1}, D^t\}$. For each

time slot t their method learns a topic model by Gibbs sampling [23] where the parameters β are a weighted mixture of the matrices $\phi^1, \ldots, \phi^{t-1}$ from the previous time slots:

$$\beta_k^t = \sum_{t'=1}^{t'=t-1} \omega^{t'} \phi_k^{t'}, \tag{6.2}$$

where ω^t is the weight associated with time slot t. The advantage of this method is mainly, that at each time slice, a separate topic model is trained, which only depends on the documents of the current batch. The relationship to the previous time slices is only established through the *prior* β.

In practice, Equation 6.2 entails that one has to keep all matrices ϕ^t associated with all time slots in memory to compute the weighted sum for the current time slot. This is inefficient in terms of memory and runtime and not in the spirit of a true online method. In their experiments, AlSumait *et al.* [1] therefore only use the previous time slot, meaning ω^t is zero for all other time slots. This makes the method more practically relevant, however, it introduces a problem: Consider the case where a certain topic occurs in one time slot, is absent in the next time slot, and reoccurs in the next. In this case, the model forgets everything from the previous occurrence of the topic since it only takes the previous time slot into account. This makes the results highly dependent on the size of the data slices and the content of the data.

In online variational Bayes (OVB, Section 2.1.6) [27], instead of taking samples, a natural gradient is calculated. After each batch, the model is then updated as

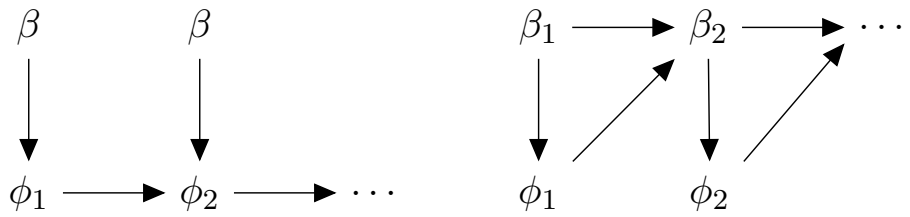
$$\phi^t = (1 - \rho)\phi^{t-1} + \rho\hat{\phi}^t, \tag{6.3}$$

where ρ is a real-valued update parameter in $[0, 1]$ interval and $\hat{\phi}$ is the estimate for ϕ based on the current batch. This way, the model converges to a stationary point of the variational objective function over time. While this model is capable of processing streaming data effectively, it does not take changing topic distributions and newly emerging topics into account. Instead, it assumes that all data come from a fixed true distribution that is approximated over time. Therefore, the differences between topics in subsequent time slices are small and become even smaller the more training data the method has processed.

6.3 New Online Topic Model

The proposed method aims to combine the advantages of the two described methods, OLDA and OVB. Both are efficient and able to process streaming data. OLDA is able to discover changing and emerging topics over time,

Figure 6.1: A word cloud for one topic related to the populist right-wing party "AfD" in Germany obtained by training a batch topic model on the corpus of news articles with 100 topics.



(a) Online Variational Bayes (b) Proposed Method

Figure 6.2: This figure illustrates the difference between online variational Bayes and the proposed method. Online variational Bayes continuously updates the word-topic distribution ϕ and keeps the prior β constant. In constrast, the proposed method updates the priors β_t using the distributions from the previous time slice. ϕ_t at different time slices t is only indirectly influenced via the prior.

whereas OVB has the ability to gradually converge to a desired distribution. I will now explain how the two approaches are combined.

In the same way as AlSumait *et al.* [1], the data is separated into different time slices $D = \{D^1, \ldots, D^{t-1}, D^t\}$. As shown in Equation 6.2, the hyperparameter β is determined based on the topic matrices ϕ^t from previous time slots t. However, instead of taking a weighted sum of all the previous topic matrices, the idea of OVB is used to *gradually* update the prior as follows:

$$\beta^t = (1 - \rho)\beta^{t-1} + \rho\phi^{t-1}. \tag{6.4}$$

This means that we let the prior parameter β converge to a stationary point instead of the topic-word distribution ϕ, whereas the topic matrices ϕ^t are specific to a certain time slot t. Thus, we can analyze the data in a

certain time slot while inducing the model to keep the topics stable over time without having to save any of the previous matrices ϕ. In contrast to the online method by Hoffman *et al.* [27], the proposed method learns topics that are only based on the data from the current time slot, making it easier to track changes or detect specific events. The difference between the proposed method and online variational Bayes is illustrated in Figure 6.2.

6.4 Experiments

A set of news articles related to the refugee crisis was extracted in the time frame of January 2016 to May 2017 from German media. The data was preprocessed by removing numbers, stop words and words with one letter and lowercased all words. Documents that are empty after preprocessing are removed. Hence, the dataset is reduced to 208,683 articles with 71,633 features. To run the online topic modeling method, we set the number of topics to 100. The time slots contain 10,000 instances each and the method is repeated 100 times for each slot. The update parameter ρ is set to $\frac{1}{100^{0.9}}$ in accordance with the default values used by Hoffman *et al.* [27]. Also, offline LDA is trained on the dataset with 100 topics.

Figure 6.1 illustrates the results of offline LDA as a word cloud for one of the topics which is mainly about the AfD party (a populist right-wing political party in Germany) and the results of the proposed online topic modeling method by presenting the topic evolution over time for the same topic. Each box presents the English translation of the top 20 most frequent words of the current topic. The dates show the start point of the time slot and every third slot is shown in this figure. It can be seen that although the topic changes over time, it is always about the AfD party. The advantage of this model over previous online LDA models (e.g. [27]) is that it puts more emphasis on the current batch, rather than updating the previous topic model incrementally with a marginal effect of the new batch.

Looking into the most frequent words of each topic in time, it can be observed that there is a lot of discussion about the Landtag elections and the Bundestag election ahead of time, e.g. the Landtag election in Saarland was supposed to be held on 26th of May 2017, but we already see the subject in the most frequent words at the 2017-02-06 time slot. Also, the Bundestag election was held in September 2017 but, because of its importance, it appears in the topics as early as in the beginning of 2017.

6.5 Discussion and Conclusion

The proposed online topic model finds topics related to refugees in German media. While the topics change over time, the main theme of each topic stays consistent over time, allowing the tracking of specific topics of interest.

Figure 6.3: An example of the topic evolution of a topic relevant to the AfD party. It is possible to see the influence of events such as party conventions and elections on the topic words.

There are many remaining questions for future work, related to both, the proposed method and the dataset used.

- Concerning this method, it is unclear how to best choose the window size. In this work, the window size was fixed to 10,000 documents, however, the window size should depend on the time frame, the number of documents that are available for a certain time frame and the required granularity.
- While the topics stay consistent over time in this method, it is still possible for new topics to emerge over time that gradually transform existing topics. It is therefore an interesting open problem to study different distance measures of topics to identify topics in the topic model where significant changes happen at certain points in time. This could be used to automatically send out alerts or warnings whenever something of relevance happens.
- Identifying the relevance of topics in general is also an open problem. It is difficult to automatically differentiate between uninteresting "background topics" and topics that capture interesting information.
- Further goals in relation to the dataset would be to identify the reasons for being against or in favor of accepting refugees among different opinions. To do that, the sentiment of the texts would have to be modeled in addition to the topics.
- It would also be interesting to cluster different news sources based on opinions expressed in their articles. This would help to provide a map of the media landscape based on their prevalent political views. However, clustering the data by source yields clusters that mainly depend on locality (e.g. Swiss media sources or southern Germany based news sources are clustered together) or on the core area of the news source (e.g. financial magazines or religious magazines are clustered together). The finer differentiating aspects between media in how they report on a controversial topic such as the refugee crisis are overshadowed by these more prevalent aspects. These questions are tied together by the open problem of clustering not just by similarity, but as a response to a specific question such as "Why are people opposed to accepting refugees?".

To conclude, this chapter shows how topic modeling may be applied to the problem of analyzing the news over time. As such it is the only unsupervised model in this thesis. Nevertheless, extensions in the spirit of the multi-label topic models of the previous chapters are possible, for example by assigning labels to some of the topics and keeping them fixed. This would result in a semi-supervised model where only some of the topics are updated.

Chapter 7

Conclusion

This thesis introduced four different generative topic models. All four are particularly efficient and scalable to large datasets. Three of them are multi-label topic models, three are online models and two are nonparametric.

7.1 Online Multi-label Topic Model

The first online multi-label topic model was introduced in Chapter 3 and is more efficient than previously existing multi-label topic models. It learns label dependencies online and achieves better classification results than online methods that do not consider dependencies. To enable this, the key factor for the success of an existing multi-label topic model was identified and utilized to develop an improved topic model. This model was inspired by the idea of greedy layer-wise training procedures as are used in neural networks. The greedy approach proves to be effective in training multi-label topic models as compared to more complex existing methods.

7.2 Nonparametric Multi-label Topic Model

The first nonparametric multi-label topic model was introduced in Chapter 4. This model has asymmetric priors for label and topic frequencies and is therefore more flexible in modeling datasets with many labels that have varying label frequencies. Efficient training is made possible by a new sampling method that reduces the sampling complexity for nonparametric topic models in general. The model shows improved results, especially on micro-averaged F-measure, which is indicative of the performance on frequent labels.

7.3 Hybrid Nonparametric Topic Model

Chapter 5 introduced a hybrid Variational-Gibbs topic model. By applying this model in the supervised setting, it was shown for the first time that HDP topic models can be effective multi-label classifiers even though the correct parameter estimation remains a problem. A doubly sparse training method employs sparsity on the updates of global parameters as well as on local document updates.

7.4 Tracking News Topics

Chapter 6 introduced a topic model to track topics over time. In contrast to previously existing models, the topics stay consistent over time allowing to analyze the evolution of specific topics depending on events such as elections or party conventions. The model was applied on a new dataset related to the refugee crisis in Germany. It makes it possible to extract insights and react to newly emerging actors such as populist political parties. Applicable in an increasingly quickly changing world, this method can efficiently be updated without having to store data from the past.

7.5 Overview of Key Results

- *New online methods:* Two different kinds of online methods were introduced in this thesis.
  - *Tracking topics over time:* This model is an example for a method that is able to model the change in the topic distributions over time. It does not assume that there is one static underlying topic distribution, but allows to model the evolution of topics.
  - *Online Fast Dep.-LLDA:* In contrast to the first method, this model assumes a single underlying distribution that is approximated by training a model online using minibatches. Here, it was demonstrated for the first time that it is possible to train multi-label classifiers online while learning label dependencies at the same time.
- *New sampling methods for HDP topic models:* This method enables time and space efficient training of HDP topic models. It was utilized in two different ways.
  - *Stacked HDP:* Here, the sampling method enables efficient training of a relatively complex hierarchical multi-label topic model with many topics and labels.

- *Hybrid Variational-Gibbs HDP:* For this model, the sampling method enables especially fast and at the same time accurate online training of HDP topic models.

7.6 Outlook

This thesis is concerned with generative models where the training and update equations are explicitly derived for each model. In the last few years, generative models have increasingly been learned using neural networks [30] that are employed as "black box" methods that can be adapted to different models without having to specify new update equations for every single case. This easy adaptability makes such models attractive. For example, it does not increase the complexity of a neural network topic model when the assumption of a mixture model, that all documents are mixtures of topics, is dropped [57]. This can make the model more expressive by allowing each document to be represented by different combinations or products of topics. Additionally, neural networks can more easily be extended to use word vectors (i.e. pre-trained vector representations of words that capture semantic and syntactic attributes of the words) or other layer types that allow to take into account word order and syntax.

Although neural networks have advantages in training topic models, they are more difficult to tune to specific datasets since they have more hyperparameters to adjust and parameters to train and generally require more training data to obtain good results. The field of neural network topic models is still in its infancy, but many promising research directions continue to emerge as the progress in the field of deep learning is fast.

Another line of future work is to train topic models using active learning. In the case of text data streams it is often difficult to label all incoming new documents by hand. Active learning could help to actively select documents that differ from previously viewed documents or where the algorithm has the least confidence during labeling and automatically infer labels for the rest. Semi-supervised extensions are also related to this field and could help to train better models with less labeled training data.

In conclusion, this thesis demonstrates the manifold possibilities and flexibility of the topic model framework for complex settings such as multi-label classification by exploring different learning and sampling strategies.

List of Figures

1.1 Overview of the methods introduced in this thesis. Chapter 6 proposes a parametric online model, Chapter 3 a parametric online multi-label model, Chapter 4 puts forward a nonparametric multi-label model, whereas Chapter 5 introduces a nonparametric online model that can also be made multi-label. Apart from the parametric online model, all models represent novel combinations of these subfields of topic model research. 3

2.1 The graphical model of LDA. 10
2.2 The graphical model of the variational distribution used to approximate the posterior of LDA. 14
2.3 Illustration of a hierarchical Chinese restaurant process. For each table at a local restaurant one customer is sent to the global restaurant. The numbers represent different topics. . 21
2.4 This figure shows schematic diagrams of CC and BCC. X denotes the input features that are used for the prediction of all labels L_i. In CC, the second classifier predicts L_2 using X and L_1, the third classifier predicts L_3 using X, L_1, and L_2 etc. In BCC, the whole first block is predicted from the input features X by the first b classifiers, the following b labels are predicted using X and the labels from the first block etc. L_{ij} denotes the jth label in the ith block. 35
2.5 The percentage of chains being better or worse than BR for different block sizes on the `medical` and `birds` datasets for BCC and BCC-IO. The performance measures were AUC and Hamming loss, respectively. The shaded area indicates the presence of the block effect. 42
2.6 The distribution of performance on the `CAL500` dataset for AUC and F-measure over randomly sampled block sizes. The performance of BCC is better than that of CC for AUC (2.6a), but worse for F-measure (2.6b). 43
2.7 Dep.-LLDA from Rubin *et al.* [54] 47

3.1 The graphical models of the original Dep.-LLDA by Rubin *et al.* [54], PAM [37], and Fast-Dep.-LDA. 53
3.2 Greedy Algorithm 56
3.3 Non-Greedy Algorithm 56
3.4 The pseudo code for batch Fast-Dep.-LLDA 56
3.5 Testset log-likelihood on the EUR Lex dataset for the greedy and the non-greedy version of Fast-Dep.-LLDA. For the non-greedy version $\beta_Y = 0.01$ is chosen. 57
3.6 Word clouds for the `Ohsumed` dataset after training for 100 iterations using SCVB-Dep.. The size of the words is scaled according to their frequencies. Each word cloud corresponds to one label and includes the 30 most frequent words for this label. 63
3.7 Label clouds for the `Ohsumed` dataset after training for 100 iterations using SCVB-Dep.. The size of the labels is scaled according to their frequencies. Each label cloud corresponds to one topic and includes the 30 most frequent labels for this topic. 64
3.8 Word clouds for the `Amazon` dataset after training for 100 iterations using SCVB-Dep.. The size of the words is scaled according to their frequencies. Each word cloud corresponds to one label and includes the 30 most frequent words for this label. 65
3.9 Label clouds for the `Amazon` dataset after training for 100 iterations using SCVB-Dep.. The size of the labels is scaled according to their frequencies. Each label cloud corresponds to one topic and includes the 30 most frequent labels for this topic. 66
3.10 This figure shows the parameter ρ^{ϕ} over 100 iterations for the settings mentioned in the text. 68
3.11 Performance of the online classifier SCVB-Dependency compared to SCVB (micro-/macro-averaged AUC). An initial batch is trained with 1000 instances. Classifiers are then updated with the next batch size instances and tested on the following batch size instances. Results are averaged over ten runs with random orderings of the datasets. 76
3.12 Runtime comparison, 3.12a to 3.12c: training runtime against testset performance in terms of micro-averaged AUC (the testset is fixed). Compared are the online method SCVB-Dep and the batch method Fast-Dep.-LLDA. For the batch method, samples are only taken from a single chain which is why the performance of the online method may be better after convergence. 77

3.13 Label and Word Clouds for Fast-Dep.-LLDA-Reversed on the `Ohsumed` dataset. The 30 most frequent words and 10 most frequent labels are shown for three exemplary topics. 78

4.1 Illustration of the difference between Stacked HDP (left) and Dependency-LDA (right). The labels are drawn from the document label set in both cases. Stacked HDP samples one topic for each word/label token, whereas Dependency-LDA samples one topic for each label in the document label set. The white rectangles are sampled variables. 84
4.2 The graphical model of sHDP compared to two alternative models, the coupled HDP (cHDP) model by Shimosaka *et al.* [56] and the nonparametric PAM model by Wei Li [36]. sHDP is a simplified model with a more effective sampling procedure. 85
4.3 Performance on micro-averaged AUC with respect to the number of samples used for evaluation. For sHDP all hyperparameters were estimated from the data. The convergence of Dependency-LDA is faster, but for three out of four datasets, the final evaluation performance of sHDP is better than that of Dependency-LDA. 96
4.4 The testset log-likelihood is compared for the different sampling methods for HDP. The improved sampling method is overall closest to that of the standard sampler, showing that a single global distribution improves convergence. 97
4.5 The runtime per iteration is compared for the different sampling methods for HDP. The alias samplers have a much lower runtime per iteration than the standard sampler. . . . 99

5.1 Effect of batchsize on the log-likelihood after 1000 updates. The truncation was set to 100 topics. 111
5.2 Effect of truncation on the log-likelihood after 1000 updates. The batchsize was set to 100 and 1000 documents as mentioned in the plot labels. For `Enron` only a batchsize of 100 was used since higher batchsizes lead to a large increase in runtime. The hybrid method has a higher log-likelihood for all settings. 111
5.3 Comparison of runtime with 100 and 1000 topics respectively. The hybrid methods were trained with a batchsize of 100 documents. Performance was evaluated on a separate testset. The hybrid method with the sparse sampling algorithm converges faster than the other methods. 113
5.4 Ranking performance with respect to parameter b_1. LDA is the result of standard Labeled LDA, HDP Updates is the result of Labeled HDP with hyperparameter optimization. . . 114

5.5 Performance of the online hybrid labeled HDP (Hybrid NP) classifier to the parametric hybrid LDA (Hybrid P) and SCVB on micro- and macro-averaged AUC. An initial batch is trained with 1000 instances. Classifiers are then updated with the next batch size instances and tested on the following batch size instances. 115

6.1 A word cloud for one topic related to the populist right-wing party "AfD" in Germany obtained by training a batch topic model on the corpus of news articles with 100 topics. 122
6.2 This figure illustrates the difference between online variational Bayes and the proposed method. Online variational Bayes continuously updates the word-topic distribution ϕ and keeps the prior β constant. In constrast, the proposed method updates the priors β_t using the distributions from the previous time slice. ϕ_t at different time slices t is only indirectly influenced via the prior. 122
6.3 An example of the topic evolution of a topic relevant to the AfD party. It is possible to see the influence of events such as party conventions and elections on the topic words. 124

List of Tables

2.1 This table provides an overview over topic models with different properties as compared to the models proposed in this thesis. 27
2.2 Notation for multi-label classification. 28
2.3 number of labels, attributes, and instances for the used datasets 38
2.4 The medians for CC and BCC on different multi-label datasets; the average difference between CC and BCC on the same chain: significant differences (0.5% significance level, Wilcoxon signed-rank test) are shown in bold, significant improvements of BCC in italics; the best block size according to the block size with the highest average value of BCC . . . 39
2.5 ECC compared to EBCC; an average over 100 runs was computed. Significant results (0.5% significance level, Wilcoxon signed-rank test) are shown in bold. The ensemble size is 20, the block size is chosen randomly for each chain. 40
2.6 CC compared to BCC; the best of 10 chains is determined using 10-fold cross validation; an average over 15 runs was computed. Significant results (0.5% significance level, Wilcoxon signed-rank test) are shown in bold. 44
2.7 Notation for Dep.-LLDA Gibbs sampling models 48

3.1 The generative process of Fast-Dep.-LLDA 54
3.2 Additional notation for SCVB-Dep. model 58
3.3 Number of labels, attributes, instances, cardinality and average number of tokens per document for the used datasets . . 67
3.4 Ranking measures for Binary Relevance with SVMs, Dep.-LLDA, Fast-Dep.-LLDA, and Fast-Dep.-LLDA with the more complex evaluation strategy S2. Results are produced by averaging distributions from 10 different sampling chains. S1 and S2 denote the different evaluation strategies for Fast-Dep.-LLDA. The best result for each dataset is highlighted in bold. 71

3.5 Classification measures for Binary Relevance with SVMs, Dep.-LLDA, Fast-Dep.-LLDA, and Fast-Dep.-LLDA with the more complex evaluation strategy S2. Results are produced by averaging distributions from 10 different sampling chains. S1 and S2 denote the different evaluation strategies for Fast-Dep.-LLDA. The best result for each dataset is highlighted in bold. 72
3.6 Ranking measures for Binary Relevance with SVMs, Dep.-LLDA, Fast-Dep.-LLDA, and Fast-Dep.-LLDA with the more complex evaluation strategy S2. Results are produced from only **one sampling chain**. */°Statistically significant difference to Dep.-LLDA at a level of **0.05**. **/°°Statistically significant difference to Dep.-LLDA at a level of **0.01** according to a Wilcoxon signed-rank test/T-test. The best result for each dataset is highlighted in bold. 73
3.7 Classification measures for Binary Relevance with SVMs, Dep.-LLDA, Fast-Dep.-LLDA, and Fast-Dep.-LLDA with the more complex evaluation strategy S2. Results are produced from only **one sampling chain**. */°Statistically significant difference to Dep.-LLDA at a level of **0.05**. **/°°Statistically significant difference to Dep.-LLDA at a level of **0.01** according to a Wilcoxon signed-rank test/T-test. The best result for each dataset is highlighted in bold. 74

4.1 number of labels, attributes, instances, cardinality and average number of tokens per document for the used datasets . . 92
4.2 Preliminary experiments concerning the hyperparameter settings of b_1 at the super-topic- and sub-topic-level respectively for sHDP using micro-averaged AUC. The best result for each dataset is highlighted in bold. The datasets and train-test splits are described in Section 4.4.2. 93
4.3 Results for all datasets on four different measures. The best value for each dataset is highlighted in bold. The rank is given in brackets. 94

5.1 Statistics for the datasets used in the experiments, $|D|$ train and test: the number of documents in the train- and testset, respectively, $|V|$: size of vocabulary 110

List of Algorithms

1 Batch Variational Bayes 17
2 Online Variational Bayes 18
3 Train Block Classifier Chains 36
4 Test Block Classifier Chains 36
5 Find Ensemble of Chains 37
6 Online Training/Inference of SCVB-Dependency 59
7 Online Greedy Layer-wise Training of SCVB-Dependency . . 62
8 Train Stacked HDP Topic Model 89
9 Train Hybrid HDP Topic Model 106
10 computeAliasTable(q) 107
11 Sample(A, w) 108

Bibliography

[1] Loulwah AlSumait, Daniel Barbará, and Carlotta Domeniconi. "Online LDA: Adaptive Topic Models for Mining Text Streams with Applications to Topic Detection and Tracking". In: *Eighth IEEE International Conference on Data Mining*. Dec. 2008, pp. 3–12.

[2] Everton Alvares-Cherman, Jean Metz, and Maria Carolina Monard. "Incorporating Label Dependency into the Binary Relevance Framework for Multi-label Classification". In: *Expert Systems with Applications* 39.2 (Feb. 2012), pp. 1647–1655.

[3] Charles E. Antoniak. "Mixtures of Dirichlet Processes with Applications to Bayesian Nonparametric Problems". In: *Annals of Statistics* 2.6 (Nov. 1974), pp. 1152–1174.

[4] Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. "On Smoothing and Inference for Topic Models". In: *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. UAI '09. Montreal, Quebec, Canada: AUAI Press, 2009, pp. 27–34.

[5] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. "Greedy Layer-Wise Training of Deep Networks". In: *Advances in Neural Information Processing Systems 19*. Ed. by B. Schölkopf, J. C. Platt, and T. Hoffman. MIT Press, 2007, pp. 153–160.

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

[7] David M. Blei and John D. Lafferty. "Dynamic Topic Models". In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: ACM, 2006, pp. 113–120.

[8] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent Dirichlet Allocation". In: *Journal of Machine Learning Research* 3 (Jan. 2003), pp. 993–1022.

[9] Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. "Learning multi-label scene classification". In: *Pattern Recognition* 37.9 (Mar. 2004), pp. 1757–1771.

[10] Andrew P. Bradley. "The use of the area under the ROC curve in the evaluation of machine learning algorithms". In: *Pattern Recognition* 30.7 (1997), pp. 1145–1159.

[11] Wray L. Buntine and Swapnil Mishra. "Experiments with Non-parametric Topic Models". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '14. New York, New York, USA: ACM, 2014, pp. 881–890.

[12] Wray Buntine and Marcus Hutter. "A Bayesian view of the Poisson-Dirichlet process". In: *arXiv preprint arXiv:1007.0296* (2010).

[13] Sophie Burkhardt and Stefan Kramer. "Multi-Label Classification using Stacked Hierarchical Dirichlet Processes with Reduced Sampling Complexity". In: *Knowledge and Information Systems* (2018), pp. 1–23.

[14] Kevin R. Canini, Lei Shi, and Thomas L. Griffiths. "Online inference of topics with latent Dirichlet allocation". In: *Proceedings of the International Conference on Artificial Intelligence and Statistics*. 2009, pp. 65–72.

[15] Olivier Cappé and Eric Moulines. "On-line expectation-maximization algorithm for latent data models". In: *Journal of the Royal Statistical Society Series B* 71.3 (2009), pp. 593–613.

[16] Changyou Chen, Lan Du, and Wray Buntine. "Sampling Table Configurations for the Hierarchical Poisson-Dirichlet Process". In: *Proc. of ECML-PKDD*. Ed. by Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 296–311.

[17] Weiwei Cheng, Eyke Hüllermeier, and Krzysztof J Dembczynski. "Bayes optimal multilabel classification via probabilistic classifier chains". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 279–286.

[18] Mauro Coletto, Andrea Esuli, Claudio Lucchese, Cristina Ioana Muntean, Franco Maria Nardini, Raffaele Perego, and Chiara Renso. "Sentiment-enhanced multidimensional analysis of online social networks: Perception of the mediterranean refugees crisis". In: *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. 2016, pp. 1270–1277.

[19] James Foulds, Levi Boyles, Christopher DuBois, Padhraic Smyth, and Max Welling. "Stochastic Collapsed Variational Bayesian Inference for Latent Dirichlet Allocation". In: *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '13. Chicago, Illinois, USA: ACM, 2013, pp. 446–454.

[20] Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. "Multilabel classification via calibrated label ranking". In: *Machine Learning* 73.2 (Nov. 2008), pp. 133–153.

[21] Shantanu Godbole and Sunita Sarawagi. "Discriminative Methods for Multi-labeled Classification". In: *Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference, PAKDD 2004, Sydney, Australia, May 26–28, 2004, Proceedings*. 2004, pp. 22–30.

[22] Henry Gouk, Bernhard Pfahringer, and Michael J Cree. "Learning Distance Metrics for Multi-Label Classification". In: *8th Asian Conference on Machine Learning*. Vol. 63. 2016, pp. 318–333.

[23] Thomas L Griffiths and Mark Steyvers. "Finding Scientific Topics". In: *Proc. of the National Academy of Sciences of the United States of America*. Vol. 101. Suppl 1. National Acad Sciences, 2004, pp. 5228–5235.

[24] Gregor Heinrich. "Parameter estimation for text analysis". In: *Technical report, Fraunhofer IGD* (2004).

[25] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. "A Fast Learning Algorithm for Deep Belief Nets". In: *Neural Computation* 18.7 (July 2006), pp. 1527–1554.

[26] Matt Hoffman, David M Blei, and David M Mimno. "Sparse Stochastic Inference for Latent Dirichlet Allocation". In: *Proceedings of the 29th International Conference on Machine Learning*. ICML '12. New York, NY, USA: ACM, 2012, pp. 1599–1606.

[27] Matthew D. Hoffman, David M. Blei, and Francis R. Bach. "Online Learning for Latent Dirichlet Allocation". In: *Advances in Neural Information Processing Systems 23*. Ed. by John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta. Curran Associates, Inc., 2010, pp. 856–864.

[28] Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. "Stochastic Variational Inference". In: *Journal of Machine Learning Research* 14 (May 2013), pp. 1303–1347.

[29] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. "Multilabel Text Classification for Automated Tag Suggestion". In: *ECML-PKDD discovery challenge*. Vol. 75. 2008.

[30] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *CoRR* abs/1312.6114 (2013).

[31] Bryan Klimt and Yiming Yang. "The Enron Corpus: A New Dataset for Email Classification Research". In: *Machine Learning: ECML 2004*. Ed. by Jean-Francois Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 217–226.

[32] Simon Lacoste-Julien, Fei Sha, and Michael I. Jordan. "DiscLDA: Discriminative Learning for Dimensionality Reduction and Classification". In: *Advances in Neural Information Processing Systems 21*. Ed. by D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou. Curran Associates, Inc., 2009, pp. 897–904.

[33] John D Lafferty and David M Blei. "Correlated Topic Models". In: *Advances in neural information processing systems*. 2006, pp. 147–154.

[34] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. "Rcv1: A New Benchmark Collection for Text Categorization Research". In: *Journal of Machine Learning Research* 5 (Apr. 2004), pp. 361–397.

[35] Aaron Q. Li, Amr Ahmed, Sujith Ravi, and Alexander J. Smola. "Reducing the Sampling Complexity of Topic Models". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '14. New York, New York, USA: ACM, 2014, pp. 891–900.

[36] Wei Li. "Pachinko Allocation: DAG-structured Mixture Models of Topic Correlations". PhD thesis. University of Massachusetts Amherst, Apr. 2007.

[37] Wei Li and Andrew McCallum. "Pachinko Allocation: DAG-structured Mixture Models of Topic Correlations". In: *Proceedings of the 23rd international conference on Machine learning*. ICML '06. New York, NY, USA: ACM, 2006, pp. 577–584.

[38] Ximing Li, Jihong OuYang, and Xiaotang Zhou. "Sparse Hybrid Variational-Gibbs Algorithm for Latent Dirichlet Allocation". In: *Proceedings of the 2016 SIAM International Conference on Data Mining, Miami, Florida, USA, May 5-7, 2016*. 2016, pp. 729–737.

[39] Eneldo Loza Mencıa and Johannes Fürnkranz. "Efficient Multilabel Classification Algorithms for Large-Scale Problems in the Legal Domain". In: *Semantic Processing of Legal Texts – Where the Language of Law Meets the Law of Language*. Ed. by Enrico Francesconi, Simonetta Montemagni, Wim Peters, and Daniela Tiscornia. 1st ed. Vol. 6036. Lecture Notes in Artificial Intelligence. Springer-Verlag, May 2010, pp. 192–215.

[40] Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Džeroski. "An Extensive Experimental Comparison of Methods for Multi-label Learning". In: *Pattern Recognition* 45.9 (2012). Best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'2011), pp. 3084–3104.

[41] Jon D Mcauliffe and David M Blei. "Supervised Topic Models". In: *Advances in neural information processing systems*. 2008, pp. 121–128.

[42] Elena Montañes, Robin Senge, Jose Barranquero, José Ramón Quevedo, Juan José del Coz, and Eyke Hüllermeier. "Dependent Binary Relevance Models for Multi-label Classification". In: *Pattern Recognition* 47.3 (2014). Handwriting Recognition and other PR Applications, pp. 1494–1508.

[43] Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. "Large-Scale Multi-label Text Classification — Revisiting Neural Networks". In: *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*. Ed. by Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo. Berlin Heidelberg: Springer, 2014, pp. 437–452.

[44] Yannis Papanikolaou, James R. Foulds, Timothy N. Rubin, and Grigorios Tsoumakas. "Dense Distributions from Sparse Samples: Improved Gibbs Sampling Parameter Estimators for LDA". In: *Journal of Machine Learning Research* 18.62 (2017), pp. 1–58.

[45] V. Perrone, P. A. Jenkins, D. Spano, and Y. W. Teh. "Poisson Random Fields for Dynamic Feature Models". ArXiv e-prints: 1611.07460. 2016.

[46] Yashoteja Prabhu and Manik Varma. "FastXML: A Fast, Accurate and Stable Tree-classifier for Extreme Multi-label Learning". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '14. ACM. New York, NY, USA, 2014, pp. 263–272.

[47] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. "Labeled LDA: A Supervised Topic Model for Credit Attribution in Multi-labeled Corpora". In: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*. EMNLP '09. Singapore: Association for Computational Linguistics, 2009, pp. 248–256.

[48] Daniel Ramage, Christopher D. Manning, and Susan Dumais. "Partially Labeled Topic Models for Interpretable Text Mining". In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '11. San Diego, California, USA: ACM, 2011, pp. 457–465.

[49] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. "Classifier Chains for Multi-label Classification". In: *Machine learning* 85.3 (June 2011), pp. 333–359.

[50] Lu Ren, David B Dunson, and Lawrence Carin. "The Dynamic Hierarchical Dirichlet Process". In: *Proceedings of the 25th International Conference on Machine learning*. ICML '08. New York, NY, USA: ACM, 2008, pp. 824–831.

[51] Ryan Rifkin and Aldebaro Klautau. "In Defense of One-Vs-All Classification". In: *Journal of Machine Learning Research* 5 (Dec. 2004), pp. 101–141.

[52] Abel Rodrıguez, David B Dunson, and Alan E Gelfand. "The Nested Dirichlet Process". In: *Journal of the American Statistical Association* 103.483 (2008), pp. 1131–1154.

[53] Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. "The Author-topic Model for Authors and Documents". In: *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*. UAI '04. Banff, Canada: AUAI Press, 2004, pp. 487–494.

[54] Timothy N. Rubin, America Chambers, Padhraic Smyth, and Mark Steyvers. "Statistical Topic Models for Multi-label Document Classification". English. In: *Machine Learning* 88.1-2 (July 2012), pp. 157–208.

[55] Ruslan Salakhutdinov, Joshua B Tenenbaum, and Antonio Torralba. "Learning with Hierarchical-Deep Models". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35.8 (2013), pp. 1958–1971.

[56] Masamichi Shimosaka, Takeshi Tsukiji, Shoji Tominaga, and Kota Tsubouchi. "Coupled Hierarchical Dirichlet Process Mixtures for Simultaneous Clustering and Topic Modeling". In: *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD), Lecture Notes in Computer Science, vol. 9852*. Ed. by Paolo Frasconi, Niels Landwehr, Giuseppe Manco, and Jilles Vreeken. Cham: Springer International Publishing, 2016, pp. 230–246.

[57] Akash Srivastava and Charles Sutton. "Autoencoding Variational Inference For Topic Models". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2017.

[58] Yee W. Teh, David Newman, and Max Welling. "A Collapsed Variational Bayesian Inference Algorithm for Latent Dirichlet Allocation". In: *Advances in Neural Information Processing Systems 19*. Ed. by B. Schölkopf, J. C. Platt, and T. Hoffman. MIT Press, 2007, pp. 1353–1360.

[59] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. "Hierarchical Dirichlet Processes". In: *Journal of the American Statistical Association* 101.476 (2006), pp. 1566–1581.

[60] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artieres, Axel Ngonga, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, Michael Schroeder, Ion Androutsopoulos, and Georgios Paliouras. "An Overview of the BIOASQ Large-scale Biomedical

Semantic Indexing and Question Answering Competition". In: *BMC Bioinformatics* 16 (2015), p. 138.

[61] Grigorios Tsoumakas, Anastasios Dimou, Eleftherios Spyromitros, Vasileios Mezaris, Ioannis Kompatsiaris, and Ioannis Vlahavas. "Correlation-Based Pruning of Stacked Binary Relevance Models for Multi-Label Learning". In: *Proc. ECML/PKDD 2009 Workshop on Learning from Multi-Label Data (MLD'09)*. Bled, Slovenia, 2009, pp. 101–116.

[62] Grigorios Tsoumakas and Ioannis Katakis. "Multi-label Classification: An Overview". In: *International Journal of Data Warehousing and Mining (IJDWM)* 3.3 (2007), pp. 1–13.

[63] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. "Mining Multi-label Data". English. In: *Data Mining and Knowledge Discovery Handbook*. Ed. by Oded Maimon and Lior Rokach. Springer US, 2010, pp. 667–685.

[64] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis P. Vlahavas. "Effective and Efficient Multilabel Classification in Domains with Large Number of Labels". In: *ECML/PKDD 2008 Workshop on Mining Multidimensional Data*. Antwerp, Belgium, 2008.

[65] Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas. "Mulan: A Java Library for Multi-Label Learning". In: *Journal of Machine Learning Research* 12 (July 2011), pp. 2411–2414.

[66] Alastair J. Walker. "An Efficient Method for Generating Discrete Random Variables with General Distributions". In: *ACM Trans. Math. Softw.* 3.3 (Sept. 1977), pp. 253–256.

[67] Hanna M. Wallach, David M. Mimno, and Andrew McCallum. "Rethinking LDA: Why Priors Matter". In: *Advances in Neural Information Processing Systems 22*. Ed. by Y. Bengio, D. Schuurmans, J.D. Lafferty, C.K.I. Williams, and A. Culotta. Curran Associates, Inc., 2009, pp. 1973–1981.

[68] Chong Wang and David M. Blei. "Truncation-free Online Variational Inference for Bayesian Nonparametric Models". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., 2012, pp. 413–421.

[69] Chong Wang, John William Paisley, and David M Blei. "Online Variational Inference for the Hierarchical Dirichlet Process." In: *AISTATS*. Vol. 2. 3. 2011, p. 4.

[70] Hongning Wang, Minlie Huang, and Xiaoyan Zhu. "A Generative Probabilistic Model for Multi-label Classification". In: *Eighth IEEE International Conference on Data Mining*. IEEE. Dec. 2008, pp. 628–637.

[71] Jörg Wicker, Bernhard Pfahringer, and Stefan Kramer. "Multi-label Classification Using Boolean Matrix Decomposition". In: *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. SAC '12. Trento, Italy: ACM, 2012, pp. 179–186.

[72] Jörg Wicker, Andrey Tyukin, and Stefan Kramer. "A Nonlinear Label Compression and Transformation Method for Multi-label Classification Using Autoencoders". In: *Advances in Knowledge Discovery and Data Mining: 20th Pacific-Asia Conference, PAKDD 2016, Auckland, New Zealand, April 19-22, 2016, Proceedings, Part I*. Ed. by James Bailey, Latifur Khan, Takashi Washio, Gill Dobbie, Joshua Zhexue Huang, and Ruili Wang. Cham: Springer International Publishing, 2016, pp. 328–340.

[73] Frank Wood, Cédric Archambeau, Jan Gasthaus, Lancelot James, and Yee Whye Teh. "A Stochastic Memoizer for Sequence Data". In: *Proceedings of the 26th International Conference on Machine Learning*. ICML '09. Montreal, Quebec, Canada: ACM, 2009, pp. 1129–1136.

[74] Yiming Yang. "A Study of Thresholding Strategies for Text Categorization". In: *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '01. New Orleans, Louisiana, USA: ACM, 2001, pp. 137–145.

[75] Ian En-Hsu Yen, Xiangru Huang, Pradeep Ravikumar, Kai Zhong, and Inderjit Dhillon. "PD-Sparse: A Primal and Dual Sparse Approach to Extreme Multiclass and Multilabel Classification". In: *Proceedings of the 33rd International Conference on Machine Learning*. ICML '16. New York, NY, USA: ACM, 2016, pp. 3069–3077.

[76] Lingfeng Zhang, Shishir K. Shah, and Ioannis A. Kakadiaris. "Hierarchical Multi-label Classification using Fully Associative Ensemble Learning". In: *Pattern Recognition* 70 (2017), pp. 89–103.

[77] Min-Ling Zhang and Zhi-Hua Zhou. "A Review on Multi-label Learning Algorithms". In: *IEEE transactions on knowledge and data engineering* 26.8 (2014), pp. 1819–1837.

[78] Jun Zhu, Amr Ahmed, and Eric P Xing. "MedLDA: Maximum Margin Supervised Topic Models for Regression and Classification". In: *Proceedings of the 26th International Conference on Machine Learning*. ICML '09. New York, NY, USA: ACM, 2009, pp. 1257–1264.