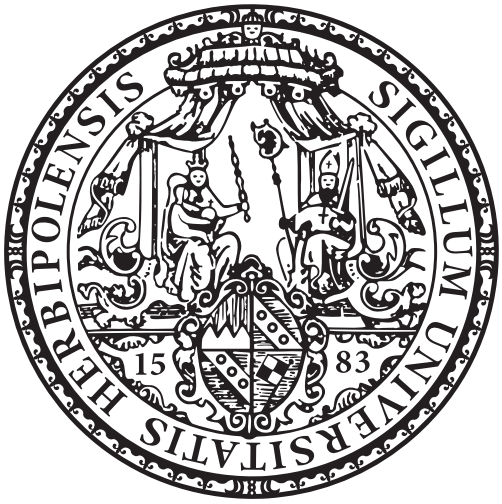
UNIVERSITÄT WÜRZBURG

INSTITUT FÜR INFORMATIK

LEHRSTUHL FÜR KÜNSTLICHE INTELLIGENZ
UND ANGEWANDTE INFORMATIK



LDA-HMMs for Domain Adaptation of Supervised Part-of-Speech Classifiers

Master-Arbeit
zur Erlangung des akademischen Grades
Master of Science (Informatik)

Vorgelegt von: Martin Felix Toepfer
Georg-Sittig-Str. 1
97074 Würzburg

Datum der Abgabe: 26.10.2012

Betreuer: Prof. Dr. Frank Puppe
Dipl. Inf. Peter Klügl

Erklärung der Selbstständigkeit

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die diesen Quellen und Hilfsmitteln wörtlich oder sinngemäß entnommenen Ausführungen als solche kenntlich gemacht habe. Die Arbeit habe ich bisher oder gleichzeitig keiner anderen Prüfungsbehörde vorgelegt.

Würzburg, 26.10.2012

Martin Felix Toepfer

Abstract

The classification of words into their Parts-of-Speech is an import step of Natural Language Processing (NLP) pipelines. While systems perform sufficiently well on this task when labeled training data from the target domain is available, Part-of-Speech tagging, and machine learning in general, is challenging when the input distribution of the target domain differs from the training data; a problem setting that often occurs in practice, for instance, when the training data comes from the financial domain and the model should be applied on medical texts.
In this thesis, we approach domain adaptation through a change of representation and representation learning (Blitzer et al., 2006; Huang and Yates, 2009; Huang and Yates, 2010): we project the data of both domains into a shared, low-dimensional space and provide a supervised discriminative sequence labeler with this representation as a feature. Especially, we concentrate our investigations on a comparison of features from certain generative models: Hidden Markov Models (HMMs) and different configurations of Latent Dirichlet Allocation Hidden Markov Models (LDAHMMs) (Griffiths et al., 2004). The former models focus on a pure sequential representation of the data, whereas the latter models additionally incorporate long-range dependencies between content words. They consider syntax, and simultaneously take topics and their term correlations into account.
Empirically, we find that all tested parameterizations of generative features result in significant improvements against a traditional supervised baseline Conditional Random Field (Lafferty et al., 2001) on domains taken from the Brown corpus. This conforms to previous work on domain adaptation and highlights the importance of feature representation as well as the feasibility and benefit of unsupervised feature creation. Interestingly, our results suggest that Hidden Markov Models may be preferred to LDAHMMs for representation learning, in particular for small corpora. For instance, for a tiny corpus with only 500 labeled training sentences we detect significant improvements of HMM features compared to LDAHMM features.

Contents

Abbreviations

CRF	Conditional Random Field
EM	Expectation Maximization
HMM	Hidden Markov Model
LDA	Latent Dirichlet Allocation
LDAHMM	Latent Dirichlet Allocation Hidden Markov Model
MAP	Maximum Aposteriori
MCMC	Markov Chain Monte Carlo
MLE	Maximum Likelihood
NER	Named Entity Recognition
NLP	Natural Language Processing
OOV	Out-of-Vocabulary
POS	Part-of-Speech
WFSM	Weighted Finite State Machine

List of Figures

List of Tables

1 Introduction

Over the last decades, research in Natural Language Processing (NLP) has made considerable progress, and more and more applications in our every day life make use of NLP techniques, for example, statistical machine translation services like Google translate[1]. The most fundamental building block of NLP pipelines is still the categorization of words into morpho-syntactic classes – the Part-of-Speech (POS) classes. Hence, it is hardly surprising that the performance of POS tagging is crucial for the accuracy of all other components. Poor POS tag assignments propagate through the successive processing units, induce false assumptions, which in turn provoke cascading errors. State-of-the-art taggers report approximately 97% word accuracy (Brants, 2000; Toutanova et al., 2003) for standard English corpora, which may seem to be sound at first glance. However, when taking a closer look, we observe at least two central problems. First, sentences typically have lengths between 10 and 20 words. This means that even with 95% word accuracy it is quite likely that a sentence contains at least one error. Indeed, Toutanova et al. (2003) measure 56.34% sentence accuracy for their tagger, and consequently components that get POS tags as input have to deal with errors in every second sentence even on data that is similar to the training data. Secondly, the errors are not equally distributed. They concentrate on Out-of-Vocabulary (OOV) words, that is, words that have not been observed in the training data. Accuracy for such words drops down to approximately 90% (Brants, 2000; Toutanova et al., 2003). This issue becomes a severe problem for real world applications when the target domain differs from the source domain. In this case, the sentences contain much more new words than the data that was used for the tagger evaluation. Besides, some words occur with different POS tags in different domains. As a consequence, not only the accuracy for Out-of-Vocabulary terms but even the word accuracy often only reaches 90% (Huang and Yates, 2009). For example, imagine we wanted to process medical text documents. Since most available POS taggers are trained on news wire texts, we expect a high amount of previously unseen words in the medical documents, and in turn a decrease in performance. Ritter et al. (2011) applied a state-of-the-art tagger on short messages from Twitter[2]. The vocabulary and the sentence structures differ a lot from the tagger's source domain since it was trained on articles of the Wall Street Journal. Without adaptation to the new domain, Ritter et al. (2011) report an accuracy of only 80% for this tagger. To re-establish the tagger's performance on the new domain, usual machine learning approaches require labeled data from the target domain for training a new model. The creation process of such data sets, however,

[1] `http://translate.google.de`
[2] `hhttp://www.twitter.com`

is typically laborious or expensive. Methods to adapt taggers from existing labeled data to new domains without costly preparing human annotated data of the target domain are highly desired.

Researchers and practitioners often encounter decreasing performance when applying their models to new data. This is a general machine learning problem which encounters in text processing, audio processing, computer vision and other tasks. As described above, the cause often lies in differences between the distributions of the source domain, where the model has initially been trained and evaluated, and the target domain, where the new data comes from. This issue is well known; the development of *domain adaptation* techniques is an active area of research (Ben-David et al., 2006; Blitzer et al., 2006; Huang and Yates, 2010). A common approach for domain adaptation is to automatically create a shared representation of both the source and the target domain with unsupervised learning. A supervised learning algorithm eventually finds out how to map from the unsupervised representation to the output labels. Huang and Yates evaluated several unsupervised learning algorithms in combination with a Conditional Random Field (CRF) for domain adaptation of a POS tagger (Huang and Yates, 2009; Huang and Yates, 2010). In this thesis, we continue the work of Huang and Yates (2009) and contribute results for domain adaptation with a Latent Dirichlet Allocation Hidden Markov Model (LDAHMM)[3] (Griffiths et al., 2004); a setting that has recently successfully been applied for semi-supervised learning (Li and McCallum, 2005).

LDAHMMs have much in common with conventional Hidden Markov Models (HMMs) but unite models of syntax and semantics. *Semantics* denotes the meaning, that is, the topic of a document in this case. Such representations can be obtained, for example, with probabilistic topic models and Latent Dirichlet Allocation (LDA) (Blei et al., 2003b). While topic models typically make a bag-of-words assumption and disregard the word order in the documents, *syntax models* focus on the word order since it is an essential part of sentence structure. A popular approach to POS tagging, for example, is the HMM that learns parameters for sequential dependencies between adjacent word classes and word emission probabilities for each class. The LDAHMM (Griffiths et al., 2004) combines LDA topic modeling and the HMM syntax model into a joint model. Remarkably, this model can be learned from unlabeled data and therefore create shared features for domain adaptation. It is, however, not clear if LDAHMMs or pure HMMs provide better representations for a subsequent discriminative POS classifier. The former approach tries to combine syntax and semantics and therefore provides more information but also needs to fit more parameters. The latter approach ignores long-range dependencies. This assumption can be oversimplifying but reduces the number of parameters. In this thesis, we conduct experiments to compare HMMs and LDAHMMs with different configuration parameters. Interestingly, our findings suggest that at least for small corpora with up to 9000 labeled training sentences Hidden Markov Models may be favored against LDAHMMs for shared feature representations of POS taggers in domain

[3] Actually, Griffiths et al. (2004) suggested no name for that model and use HMM-LDA in their software toolbox. In this thesis, we focus on the HMM aspect and therefore re-ordered the terms.

adaptation settings. We detected significant improvements of both approaches against a baseline POS tagger with only lexical features. However, HMM features performed significantly better than LDAHMM features when only 500 labeled sentences were provided. Both representations performed almost equally when more than 8000 sentences were given.

In the following chapters, we first introduce generative probabilistic graphical models (Chapter 2) before we discuss some special approaches to topic modeling (Chapter 3), shallow syntax representations (Chapter 4) and combined models (Chapter 5). In Chapter 6, we review domain adaptation and transfer learning approaches, and describe a change of representation approach with representation learning. Chapter 7 contains our experimental settings, results, a discussion and a comparison to related work. Finally, we point to future work in Chapter 8 and conclude in Chapter 9.

2 Generative Graphical Models

Topic Models and Hidden Markov Models originate from the same statistical framework. Both can be described in terms of generative stochastic processes and directed graphical models, a general approach to probabilistic reasoning present in standard artificial intelligence literature. They cover a broad range of applications, such as machine learning in vision, natural language processing and computational biology or uncertain databases. There are a number of good publications to learn about generative statistical graphical models, for detailed information we refer to existing literature, (e.g., Koller and Friedman, 2009; Darwiche, 2008). In this section, we give a brief introduction into *generative statistical graphical models*, which are also known as *Bayes Nets*, or *Belief Nets*. We focus on some special aspects that we will need later to better understand topic models and HMMs. We first take a look at the basic structure and properties of generative statistical graphical models. Then, we review typical questions that can be answered with these models. We will explain the plate notation, which will be convenient in the rest of the thesis. Lastly, we outline an approximate inference technique, namely Gibbs sampling, and give an overview of parameter estimation approaches.

2.1 Motivation

Many real world processes, and especially many aspects of natural language, can be seen as outcomes of random processes. In a way, the creation of text documents, for example, can be regarded as drawing its words as outcomes of a discrete probability distribution. If we disregard the dependencies in between words, drawing the word *the* has in general a high chance while drawing *tardigrada* has a low chance. It is obvious that this simplistic model violates grammatical and semantic rules. However, we have to make at least some simplifying assumptions to make models tractable. Modeling stochastic processes that have many random variables and possible outcomes is costly and sometimes impossible; too many parameters have to be fit. The general description of a discrete distribution, that is, a model without independence assumptions, grows exponentially with the number of variables. This makes full statistical descriptions intractable in practice; we have to introduce independence assumptions among the random variables. Independence assumptions reduce the computational complexity and the number of parameters we have to fit to describe the system. As a very simple example, let us consider that we repeatedly throw three distinguishable dice with 6 possible outcomes for each of the dice. If we make a naive approach, we need to describe the full joint distribution with all $6 \cdot 6 \cdot 6 = 216$

parameters. However, if we assume the outcomes of the three dice to be independent of each other, we only need to know the distribution of each dice to describe the joint distribution. This reduces the number of parameters to $6 + 6 + 6 = 18$, which is only $8,\bar{3}\%$ of the number of parameters before.

Even in the simple example above, independence assumptions have compressed the number of parameters dramatically. Indeed, independence assumptions become crucial when we have to deal with many random variables and possible outcomes. Compared to the 6 possible outcomes of each dice in the example above, consider, for example, language modeling where we draw words from a vocabulary with thousands of different terms. Actually, we face infinitely many variables and possible outcomes in natural language since language is a never ending stream of words, and every day we encounter new word types. Besides, we change the way in which we make use of the words over time. For example, consider the following sentence.

(1) Mike tweeted that his PC had crashed down.

One hundred years ago, *PC* was not part of the English vocabulary, and *tweeted* recently changed its contextual meaning. Hence, in order to process the large numbers of variables and possibles outcomes that we encounter in natural language texts, we need efficient ways to represent and exploit independence assumptions.

2.2 Data Structure

Generative graphical models describe the independence assumptions of a distribution and provide a data structure which enables more efficient inference algorithms and human understandable visual depictions of a domain. The fundamental property of a generative graphical model is the correspondence between a *factorization* of a full joint distribution of random variables X_i into conditional probability distributions $P(X_i|\{X_j\}_{j\in I_i})$, where each index set I_i indicates a subset of all variables X_i, and a *directed acyclic graph* (DAG)[1].

This matching can be summarized in only two statements:

- *The random variables of the domain map one-to-one to the nodes in the directed acyclic graph.* This property allows us to simplify explanations. We can talk about variables and implicitly refer to their representatives in the graph and vice versa. For instance, if we talk about the parents of a random variable, we refer to the parents of the node which corresponds to that variable in the graph.
- *The parents of each node X_i in the graph are the variables that X_i is dependent of in the factorization of the full joint probability.* Put in different words, the index

[1] A directed acyclic graph is a graph that has directed edges and that contains no cycles.

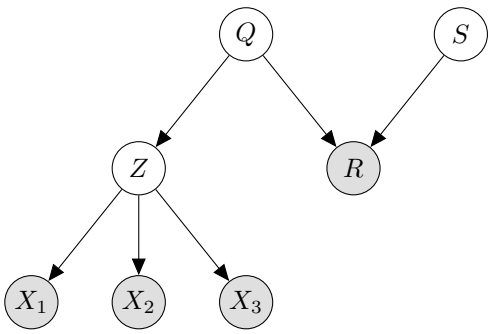
sets I_i of the factorization and the sets of the parents of the corresponding nodes parents(X_i) map one-to-one.

With the second property we can rewrite the factorization by

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | \text{parents}(X_i)), \tag{2.1}$$

where parents(X_i) denotes the parents of X_i: the set of variables whose nodes X_j are connected to X_i, that is, there is an edge from X_j to X_i in the graph.

In general, generative graphical models can handle both variables with continuous and discrete outcomes. For simplicity, we will focus on discrete variables in this section although the extension to continuous variables is straight forward. We will use variables with continuous values later, for example, in Section 3.1. In the discrete case, the conditional probability distributions $P(X_i|\{X_j\}_{j\in I_i})$ can be represented by *conditional probability tables* (CPT). For some variable X_i, its conditional probability table contains the probabilities of all combinations of values that the variables parents(X_i) and X_i can take. Each row contains such a configuration plus the corresponding probability.



Q	p
0	0.7
1	0.3

S	p
0	0.2
1	0.8

Q	**Z**	p
0	1	0.6
1	1	0.7

Z	$\mathbf{X_i}$	p
0	1	0.1
1	1	0.8

Q	S	**R**	p
0	0	1	0.3
0	1	1	0.1
1	0	1	0.8
1	1	1	0.9

Figure 2.1: Generative statistical graphical model (Bayes Net) with conditional probability tables.

Figure 2.1 shows a small example with seven variables $Q, S, Z, R, X_1, X_2, X_3$. For simplicity, all variables in this example are binary variables; they can only take the values zero or one. On the left, we can see the graph structure of the domain. On the right side, we can see the conditional probability tables for the variables where the main variable is printed in boldface. The table at the top left-hand corner of the right side shows the CPT of the variable Q. It has only two rows because the node of Q has no parents in the graph. With probability 0.7 Q takes the value zero, with probability 0.3 it is one. Below this table, we see the CPT of S. It is very similar to the table of Q since both have no parents in the graph. The table at the top right-hand side shows Z's

CPT. This variable has one parent in the graph, namely the variable Q. Thus, the CPT should have four rows for the four possible configurations of these two binary variables. However, in this case it is possible to omit the probabilities for the settings where $Z = 0$ since they can be derived by $P(Z = 0|Q = q) = 1 - P(Z = 1|Q = q)$ where q can be zero or one. In this example, we assume, that all variables X_i follow the same distribution. Therefore, we only need one table for all of these variables. This table is depicted in the middle of the right-hand side. The table at the bottom shows the conditional probabilities for the variable R. Again, we skip probabilities for $R = 0$ since they can be computed from the given values. R has two parents, and accordingly this table is a compressed representation of a table with 8 rows. Finally, for illustration purpose, we compute the probability of a certain configuration:

$$\begin{aligned} &P(Q = 1,S = 0, R = 1, Z = 0, X_1 = 0, X_2 = 0, X_3 = 1) \\ &\quad = P(Q = 1)P(S = 0)P(R = 1|Q = 1, S = 0)P(Z = 0|Q = 1) \\ &\qquad \cdot P(X_1 = 0|Z = 0)P(X_2 = 0|Z = 0)P(X_3 = 1|Z = 0) \\ &\quad = 0.3 \cdot 0.2 \cdot 0.8 \cdot (1 - 0.7) \cdot (1 - 0.1) \cdot (1 - 0.1) \cdot 0.1 \approx 1.16 \cdot 10^{-3}. \end{aligned}$$

A first useful qualitative result of this data structure addresses the conditional dependence between the variables of the domain; we explain the meaning of the edges in the graph by a notion of direct influence. At first, we introduce the term *conditional independence*. Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ be three distinct sets of random variables of a generative graphical model. If and only if

$$P(\mathbf{x}|\mathbf{y}, \mathbf{z}) = P(\mathbf{x}|\mathbf{z}) \qquad \text{or} \qquad P(\mathbf{y}, \mathbf{z}) = 0 \tag{2.2}$$

for all possible configurations $\mathbf{x}, \mathbf{y}, \mathbf{z}$ of $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$, we say that $\mathbf{X}$ is *conditionally independent* of $\mathbf{Y}$ given $\mathbf{Z}$ (cf. Darwiche, 2008, pg. 470); we write $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$. Crucially, it can be shown that each variable in a generative graphical model is conditionally independent of all of its non-descendants given its parents. Finally, we arrive at the term *Markov blanket*. Let X_i be an arbitrary variable in the graph, $\mathbf{C}_{X_i} \coloneqq \text{children}(X_i)$, and $\mathbf{T}_{X_i} \coloneqq \bigcup_{X_j \in \mathbf{C}_{X_i}} \text{parents}(X_j) \setminus X_i$. The *Markov blanket* $\mathbf{M}_{X_i}$ of X_i is the union of the parents and the child nodes of X_i and the parents of the child nodes (without X_i), that is, $\mathbf{M}_{X_i} \coloneqq \text{parents}(X_i) \cup \mathbf{C}_{X_i} \cup \mathbf{T}_{X_i}$. The Markov blanket has a nice property that can be exploited for efficient approximate inference algorithms: X_i is conditionally independent of all variables in the network given the values of the variables in its Markov blanket $\mathbf{M}_{\mathbf{X_i}}$. That is, $X_i \perp\!\!\!\perp X_j|\mathbf{M}_{X_i}$ for arbitrary $X_j \notin \mathbf{M}_{\mathbf{X_i}} \cup \{X_i\}$. Figure 2.2 illustrates a Markov blanket of a variable of a graphical model. In this example, X_4 and X_5 are part of the Markov blanket of X_7 because they are parents of X_7. The children of X_7, namely X_9 and X_{11}, also belong to the Markov blanket of X_7. The last node that is missing is X_8 because it is a parent of a child (X_{11}) of X_7.

Given a full joint probability distribution, there are several possible ways to structure the distribution into a generative graphical model. However, they may vary crucially in their efficiency; an inexpedient formulation can have as many parameters as the full

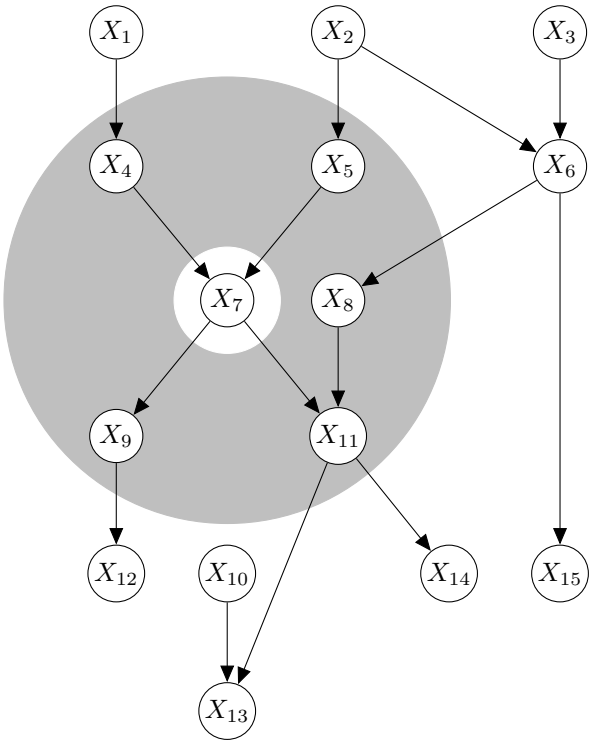


Figure 2.2: Markov blanket of variable X_7

joint probability, whereas a proper formulation, however, may only need to fit a small fraction of that. Especially, it can be beneficial to construct a generative graphical model according to dependencies that have *causal* instead of *diagnostic* meanings since diagnostic models need to model additional dependencies between the diagnostic variables (cf. Russell and Norvig, 2010, p. 517).

2.3 Plate Notation and Types of Variables

Visualising generative graphical models can further benefit from two common extensions.

Firstly, we often have two different kinds of random variables: random variables whose outcomes are observed and therefore known, and random variables whose outcomes are hidden. The former are called *observations*, the latter are called *latent* variables, *hidden* variables, or *labels* in some special cases. We can highlight this partitioning in the graph by giving nodes of observed variables a grey shade and leaving nodes of hidden variables blank. In the rest of this thesis, we will always assume that this partitioning into observed and latent variables is present. However, note that even if latent variables

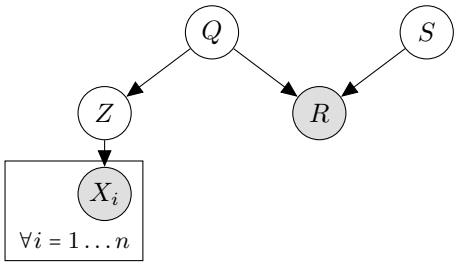


Figure 2.3: Plate notation of a generative process which generalizes the graphical model that is shown in Figure 2.1.

are hidden and observable variables are observed, we can assume to know the value of latent variables during computation, as well as we can derive the probability that an observable variable takes a certain value if its state is not given. Besides, there may be a third type of variables, called *constants*. They do not change their value during the generative process, nevertheless, they are variables and we can infer their values. This notion is commonly used, however, it can be misleading at first glance. We will encounter constants in the following sections, for instance, Figure 3.2 on page 18 contains the two constants α, β. We depict constants by their name without a circle around it.

Secondly, we can compress graphs with recurring structure. If some subgraph is repeated several times, we plot it only once and draw a frame around it. The number of draws and an index variable that may be used inside the plate are depicted in the lower right corner of the frame.

Figure 2.3 shows the plate notation of a generative graphical model. The model is the same as in Figure 2.1 when we set $n = 3$. The variables Q, S, Z are assumed to be hidden, whereas the variables X_i, R are assumed to be observed. Note that we have a short representation for the variables X_i through the use of a plate. A plate can be thought of as a template which is iteratively used to *unroll* the graph. Plates can also be nested, which will encounter in the following sections.

2.4 Tasks

The first task when starting to work with a generative graphical model involves two steps. We have to formulate which random variables are involved, this addresses the nodes of the graph, and how the full joint probability factorizes into conditional distributions. The latter addresses the edges of the graph. Although there is ongoing research to automatically induce random variables and the factorization of the joint distribution with observed data, which is called *structure learning*, the factorization is typically done manually. We will also focus on handcrafted model structure.

For a fixed structure, the quality of generative graphical model depends on its parameterization. Assume that we have already formulated which random variables are involved and where we expect causal dependencies. To actually compute probabilities with Equation (2.1) for a certain assignment to the variables, we have to specify the conditional probabilities $P(X_i|\,\text{parents}(X_i))$. This time, we need to apply automated reasoning, which is known as *parameter learning*, since handcrafted specifications of the conditional probability distributions are both error prone, due to subjectivity, and laborious, because we often have to estimate thousands to millions of parameters.

The parameter learning task is closely related to *inference*. Inference algorithms compute the probabilities of assignments to some of the random variables, the *query*, when the values of some different set of random variables is already known as *evidence*.

Similar to Darwiche (2008), we differentiate tasks for a given generative statistical graphical model:

probability of evidence Compute $P(\mathbf{e})$ for some given evidence $\mathbf{e}$.

maximum a posteriori hypothesis Given some evidence $\mathbf{e}$, determine the most likely joint assignment $\hat{\mathbf{x}}_{\text{mpe}} = \arg\max_{\mathbf{x}} P(\mathbf{x}|\mathbf{e})$ for a set of query variables X.

posterior marginals For some set of query variables $X = \{X_i\}$ and evidence $\mathbf{e}$, estimate the distributions $P(X_i|\mathbf{e})$ for all $X_i \in X$.

2.5 Gibbs Sampling

Assume we have given a generative statistical graphical model with structure and parameters, and evidence as a subset of variables whose outcomes are known. Now, we face the posterior marginals task, which means that we would like to compute the marginal probabilities for the outcomes of each query variable conditioned on the evidence. In the general case, this is a challenging problem, which makes exact inference not tractable. Sampling is an approximate inference approach which draws variables at random and provides frequencies from which we can in turn estimate the distribution.

A special type of sampling algorithms are Markov Chain Monte Carlo (MCMC) algorithms. They successively alter full assignments, which are called *states*, with changes that are only dependent on the directly preceding assignments. This is a *Markov chain* of first order. Besides, the changes occur stochastically, hence the name Markov Chain Monte Carlo. Especially, MCMC algorithms design the sampling process in such a way that its stationary distribution is the target distribution. Thus, samples from the MCMC process can be used to estimate properties of the target distribution. A common member of the MCMC family is Gibbs sampling (Geman and Geman, 1984). In the following, we assume that the set of all variables partitions into query and evidence variables; the algorithm can naturally be extended to work with a tripartite set of variables: evidence, query, and all other variables.

Input : evidence $\mathbf{e}$, set of query variables X, structure and parameters of a graphical model, number of samples k, burn-in period b
Output: $\forall X_i \in X : p(X_i|\mathbf{e})$ (approximately)

```
// random initialization
for i ← 1 to |X| do
    x_i ~ unif(dom(X_i))
end
// burn in
for i ← 1 to b do
    for i ← 1 to |X| do
        x_i ~ p(X_i|X_-i, e) ;          // depends only on markov blanket of X_i
    end
end
// take samples
for j ← 1 to k do
    for i ← 1 to |X| do
        x_i ~ p(X_i|X_-i, e) ;          // depends only on markov blanket of X_i
        n_{i,x_i} ← n_{i,x_i} + 1;
    end
end
// normalize counts to probabilities
N ← N/k ;
// return probabilities
return N ;                                          // p(X_i = j) = n_ij
```

Algorithm 1: Simplified Gibbs sampling procedure (cf. Russell and Norvig, 2010, p. 537)

The Gibbs sampling procedure is depicted in Algorithm 1. It works as follows. After random initialization of all non-evidence variables, we iterate over all the query variables and draw each X_i conditioned on the fixed assignment of all other variables, which we denote by X_{-i}. Note that for graphical models the Markov property simplifies the computations in most cases; we do not need to consider the assignments of *all* other variables, but only the assignments of variables in the neighborhood of the currently sampled variable in the graph. More precisely, we only need to regard the Markov blanket $\mathbf{M}_{X_i}$ of X_i since sampling X_i is independent of all variables outside the Markov blanket given $\mathbf{M}_{X_i}$. Such computations are typically light-weight, and are feasible even in graphical models with many variables. During sampling, we keep statistics over the assignments to the query variables. We yield final estimates by normalizing these statistics after gathering enough samples. It is common practice to reject a predefined number of samples in the beginning, which is called *burn-in*, to avoid biases from the random initialization. Note, however, that we only need to be sure that we have reached convergence before we start collecting statistics. The burn-in period can be omitted when the initialization procedure provides estimates that are close to convergence. Actually, diagnosing convergence is not

trivial. In practice, the burn-in period approach and likelihood analysis often yield sufficient results. Practitioners additionally often reject a certain amount of draws between collected samples, where this number is called *lag*. Under specific circumstances, it can be shown that Gibbs sampling provides consistent estimates. We skip this proof and refer to standard artificial intelligence literature (e.g. Russell and Norvig, 2010, p. 536). We will make use of Gibbs sampling in both Topic Models and HMMs.

The Gibbs sampling algorithm as described above divides the variables into evidence variables (their values are fixed) and query variables (their values are sampled). In some cases, we can skip sampling some subset of the query variables by integrating them out, which is known as *collapsed Gibbs sampling*. We will revisit collapsed Gibbs sampling in Section 3.1 for posterior inference in LDA topic models.

2.6 Parameter Estimation

In practice, it is often possible to define the model structure of Bayesian Networks by experts. However, there is too much uncertainty in human estimates of the model parameters θ, that is, the entries of the probability tables. Automated estimation methods differ in several aspects and can be categorized along different dimensions. Methods that depend on annotated training data are called *supervised learning* methods. The terms *unsupervised learning* and *semi-supervised learning* are used when we have unlabeled data, but either no annotated training data is given at all, or just a little amount of labeled examples is available respectively. In the following, we take a look at different aspects of inference (c.f., MacKay, 2005).

A common approach to parameter estimation directly maximizes the likelihood of the parameters θ given the data D, that is,

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} P(D|\theta) \tag{2.3}$$

which is the *Maximum Likelihood (MLE)* estimation. We use the hat notation $\hat{\theta}$ for the estimated values.

When we have additional knowledge about the prior distribution on θ, we can instead set

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} P(D|\theta)P(\theta) \tag{2.4}$$

which is known as the *Maximum Aposteriori (MAP)* estimation. In Equation (2.4), $P(\theta)$ should be a non-uniform prior; a uniform prior on θ would again yield the MLE estimate. The additional knowledge that we put into parameter estimation with the MAP approach can especially *constrain* the parameters in certain aspects. For instance, assume that the length of the parameter vector would represent some kind of complexity of the model. According to a fundamental principal that is called *Occam's razor*, we would like to have a simple solution. In that case, we have to avoid parameter vectors that have a great

vector length. This can be achieved by MAP parameter estimation with a prior that favors short parameter vectors.

The likelihood of the parameters in Equation (2.3)) and Equation (2.4) usually reflects a product over many data instances such that $P(D|\theta) = \prod_i P(x_i|\theta)$. As a consequence, we have to deal with very small numbers which can cause problems with internal number representations. To avoid this issue, we often employ the log of the likelihood, which is called *log-likelihood*.

MLE and MAP can also be applied to models like CRFs that disregard the dependencies between the observed variables. They seek to maximize the conditional probability of the hidden variables given the observed variables instead of the joint distributions that are used in generative models. Hence, they belong to the class of classifiers that are called *discriminative*, or *conditional* models. We will describe CRFs for POS tagging in Section 4.3.

In Equation (2.3) and Equation 2.4, D can refer to unlabeled or labeled data, which addresses unsupervised learning or supervised learning respectively. If the data is labeled, the MLE approach sometimes reduces to simply acquiring count statistics. In some cases, we can analytically determine a derivative of the likelihood or the log-likelihood and then use efficient mathematical optimization routines. For example, we will use parameter learning based on quasi-Newton optimization in Section 4.3 to tune the parameters of a CRF. If, however, the data is unlabeled, we need to integrate over all possible assignments to the hidden variables. A popular approach to unsupervised learning with MLE is *Expectation Maximization (EM)* (cf. Dempster et al., 1977). This algorithm consists of two steps which are iteratively applied: the Expectation-step uses the current model parameters to find estimates for the hidden states; in the Maximization-step, this assignment is used to find the parameter set that best fits to this data. Although EM often finds good approximations of the parameters, Johnson (2007) states that it has problems with parameter estimation for HMMs to do Part-of-Speech tagging.

A possible alternative to learning fixed parameters is the Bayesian approach. Parameters gathered with MLE or MAP are typically used to infer the states of hidden variables. *Bayesian inference* instead integrates over all possible parameter values and assigns the hidden variables to values without a fixed parameter estimation. Priors on the parameters can be used to integrate background knowledge into the model. For example, we can choose appropriate priors to enforce sparse, skewed distributions as they typically appear in Natural Language Processing. We will make use of Bayesian inference for Topic Models, HMMs, and LDAHMMs in Sections 3, 4 and 5, respectively.

3 Probabilistic Topic Models

Considering morphology, analysing the sequential and hierarchical structure of sentences, and other deep text analysis computations are laborious. The good news is that a shallow and less expensive view on documents suffices for some applications. Especially for the common task of text document retrieval, we can ignore the word order and instead consider just word frequencies. Even humans often break the strict sequential order of language when processing texts. Imagine, for example, that we should find an article about some specific group of diseases in a large, unorganized collection of text documents covering a broad range of topics. Certainly we would avoid to fully read every document word by word. Instead, we would first try to capture the topic of the document; we would skim over the sentences and words to find terms that let us categorize the document. We would only proceed to deeply analyse the text if we assume that our search terms fit to the topic of the document. Thus, it should be possible to build shallow representations of the *meaning* of documents, simply based on counts of word occurrences. This *semantic* representation of the documents accompanies a reduction of dimensionality. We can project documents into the low-dimensional *topic space* where many operations can be carried out more efficiently than with the plain, high-dimensional term-document counts.

The next question is, how we can describe the interrelationship between words, their contexts in the documents, that is, their surrounding words, and topics. Furthermore, we want the representation to be small and we want to spent as less manual effort for supervision as possible. There has been extensive research to address these questions, especially since the rise of search engines. Practitioners can choose between several alternatives, for example, multinomial Bayes, Latent Semantic Indexing (LSI) (Deerwester et al., 1990), or Latent Dirichlet Allocation (LDA) (Blei et al., 2003b). Each of these approaches has its strengths and weaknesses; some approaches are simpler to compute, others have a better compression rate, or provide results that are better interpretable or more comprehensive.

In the following, we will take a closer look at topic modeling with LDA which approaches the stated problem with a generative statistical graphical model. Although LDA can be applied to domains other than text documents, we will keep text document generation and topic modeling as a running example for better understanding. Due to its statistical background, LDA provides interpretable results. Moreover, we can easily vary and extend it, as we will see in Section 5.2 where we combine LDA with a sequential model. Efficient inference algorithms for LDA topic modeling have been found and the quality of the models is competitive with the state-of-the-art.

Similar to the creation process of this thesis, which deals with certain aspects of computer science and linguistics, LDA topic models assume that the words of a document are generated according to some latent topics Documents originate from *mixtures* of topics, where a *topic* is a specific distribution over words. If, for example, a document is closely related to a certain topic, the words of this document mostly come from the distribution of that topic. A news article of the sports channel of an online news portal is dominated by words like penalty, shoot, scores, referee, etc., which we would describe as the sports topic compared to other articles that, for example, tell us about politics, science, or lifestyle. If documents were, however, restricted to belong to just one category, we would not match the semantics of real world text documents. For this reason, LDA follows a mixture modeling approach that also allows for documents that are influenced by different topics. Hence, it is possible to represent documents that are related to, for instance, sports *and* politics, like a news article about funding a hockey arena by the local township, or sports and fashion, like an article about the latest tennis fashion. An additional feature of LDA topic modeling is that its output provides quantitative membership information for each document and each topic. We can compute the grade of membership of each topic for a certain document, or rank documents according to their relation to a specific topic. Moreover, LDA topic models give insight into the vocabulary of each topic, for example, we can list the most frequent terms for each topic (cf. Table 3.2 on page 20).

In the next sections, we first describe Latent Dirichlet Allocation. Afterwards we explain how LDA topic models can be visualized and how inference by collapsed Gibbs sampling works. Finally, we shortly discuss model selection and evaluation of topic models.

3.1 Latent Dirichlet Allocation

According to Blei et al. (2001) and Griffiths and Steyvers (2002)[1], assume we have D documents and T topics, and we observed V different word types, or terms synonymously. Each document $d \in \{1, \dots, D\}$ has a finite number $N_d \in \mathbb{N}$ of words, which we summarize as $\mathbf{w}^{(d)} = (w_1^{(d)}, \dots, w_{N_d}^{(d)})$ with $w_i^{(d)} \in \{1, \dots, V\}$ for all $i \in \{1, \dots, N_d\}$. The concatenation of all words of all documents in the corpus is $\mathbf{w}$. Each word $w_i^{(d)}$ is mapped one-to-one with a topic state $z_i^{(d)} \in \{1, \dots, T\}$ and many-to-one with its document state $d_i \in \{1, \dots, D\}$. The vector $\mathbf{z}$ contains the topic states that correspond to the words in $\mathbf{w}$. We represent the probabilities $P(w_i^{(d)} | z_i^{(d)})$ of observed word types given their topic states with a T dimensional vector ϕ of multinomial distributions. Each component $\phi^{(j)}, j \in T$ of ϕ is the word distribution of a topic, such that $P(w_i^{(d)} = v | z_i^{(d)} = j) = \phi^{(j)}(v)$. Hence, ϕ can be thought of as a $V \times T$-dimensional matrix: the topic matrix. We also introduce

[1] LDA was contributed by Blei et al. (2001). However, we use the variant proposed by Griffiths and Steyvers (2002) who place an additional Dirichlet prior on ϕ.

a D dimensional vector θ of multinomial distributions which represent the probabilities $\theta^{(d)}(j)$ of sampling a certain topic j for each of the documents. We can regard θ as a $T \times D$-dimensional matrix: the document matrix. The matrix factorization view of LDA is depicted in Figure 3.1, however, it would probably be better that we first introduce the formal aspects of LDA and than have a look at this figure again.

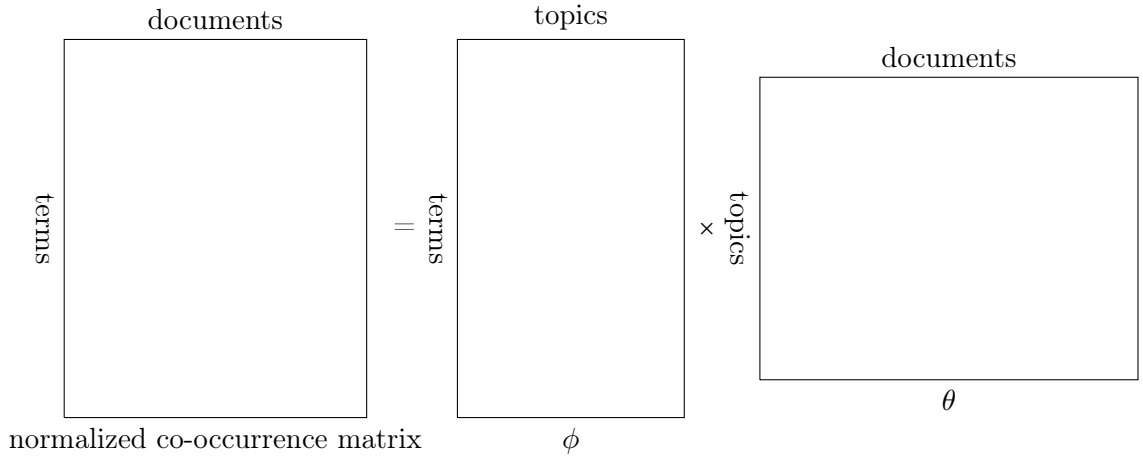


Figure 3.1: Matrix factorization of Latent Dirichlet Allocation (cf. Steyvers and Griffiths, 2007).

With two constant hyperparameters $\boldsymbol{\alpha} \in \mathbb{R}_+^T, \boldsymbol{\beta} \in \mathbb{R}_+^V$, the creation of each document d in the corpus follows a simple algorithmic scheme:

- Initially, determine a continuous topic mixture vector $\theta^{(d)}$ for that document by sampling from Dirichlet$(\cdot|\boldsymbol{\alpha})$.
- Sample N_d times[2] from Mult$(\cdot|\theta^{(d)})$ where N_d is the number of words in the document. We obtain N_d topic states z_1 to z_{N_d} with each $z_i \in \{1, \ldots, T\}$.
- Sample N_d times a word $w_i \in \{1, \ldots, V\}$ from Mult$(\cdot|\phi^{(j)})$ where j is the value of z_i, that is, the ith topic state determines from which word distribution (topic) we actually draw the word.

The corpus creation process given above assumes that we already know the topic distributions. From the generative process perspective, they are determined before the documents of the corpus are created:

- for each $j \in \{1, \ldots, T\}$, sample topic $\phi^{(j)}$ from Dirichlet$(\cdot|\boldsymbol{\beta})$.

[2]Some descriptions of LDA let the number of words per document to be sampled from a Poisson distribution (Blei et al., 2003b; Heinrich, 2009). Since we always operate on given corpora in this thesis, we assume N_d to be known for each document.

Table 3.1: Notation for the LDA topic model

Symbol	Domain	Description
$\boldsymbol{\alpha}$ / α	$\mathbb{R}_+^T$ / $\mathbb{R}_+$ (symmetric)	hyperparameter for the topic mixture vectors of the documents
$\boldsymbol{\beta}$ / β	$\mathbb{R}_+^V$ / $\mathbb{R}_+$ (symmetric)	hyperparameter for the vocabulary mixture vectors of the topics
D	$\mathbb{N}$	number of documents
N_d	$\mathbb{N}$	number of words in document d
N	$\mathbb{N}$	sum over all N_d, i.e., $N = \sum_{d=1}^{D} N_d$
T	$\mathbb{N}$	number of topics
V	$\mathbb{N}$	vocabulary size
$\theta^{(d)}$	$\mathbb{R}_+^T, \sum_{j=1}^{T} \theta^{(d)}(j) = 1$	topic mixture of document d
$\phi^{(j)}$	$\mathbb{R}_+^V, \sum_{v=1}^{V} \phi^{(j)}(v) = 1$	word distribution of topic j
$z_i^{(d)}$	$\{1, 2, \ldots, T\}$	topic of the ith word of document d
$\mathbf{z}$	$\{1, 2, \ldots, Z\}^N$	concatenation of all $\mathbf{z}^{(d)}$
$w_i^{(d)}$	$\{1, 2, \ldots, V\}$	word type of the ith word of document d
$\mathbf{w}$	$\{1, 2, \ldots, T\}^N$	concatenation of all $\mathbf{w}^{(d)}$
d_i	$\{1, 2, \ldots, D\}$	document index of the ith word
$\mathbf{d}$	$\{1, 2, \ldots, D\}^N$	document indices vector

The generative process is depicted in Figure 3.2. The hyperparemters α and β determine how the documents' topic mixture vectors $\theta^{(d)}$ and the topics $\phi^{(j)}$ are sampled from the corresponding Dirichlet distributions. An actual word $w_i^{(d)}$ depends on its topic state $z_i^{(d)}$, which is drawn from the multinomial distribution $\theta^{(d)}$ of the document d, and the topic $\phi^{(j)}$, where $j = z_i^{(d)}$. Table 3.1 summarizes all involved notation.

Dirichlet Distribution

Before we take a look at different aspects of LDA topic modeling, we first recap the Dirichlet distribution. The probability density of the Dirichlet distribution Dirichlet$(\cdot)$ over multinomials $\mathbf{p} = (p_1, \ldots, p_K) \in [0, 1]^K$, $\sum_{k=1}^{K} p_k = 1$ and parameterized by $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K) \in \mathbb{R}_+^K$ is given by

$$\text{Dirichlet}(\mathbf{p}|\boldsymbol{\alpha}) = \frac{\Gamma\left(\sum_k \alpha_k\right)}{\prod_k \Gamma(\alpha_k)} \prod_{k=1}^{K} p_k^{\alpha_k - 1}. \tag{3.1}$$

Like most publications on LDA topic modeling, we will always assume symmetry for all Dirichlet distributions in this thesis. In this case, we just need one single parameter for

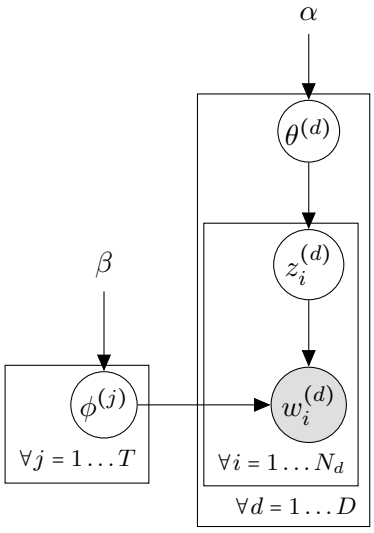


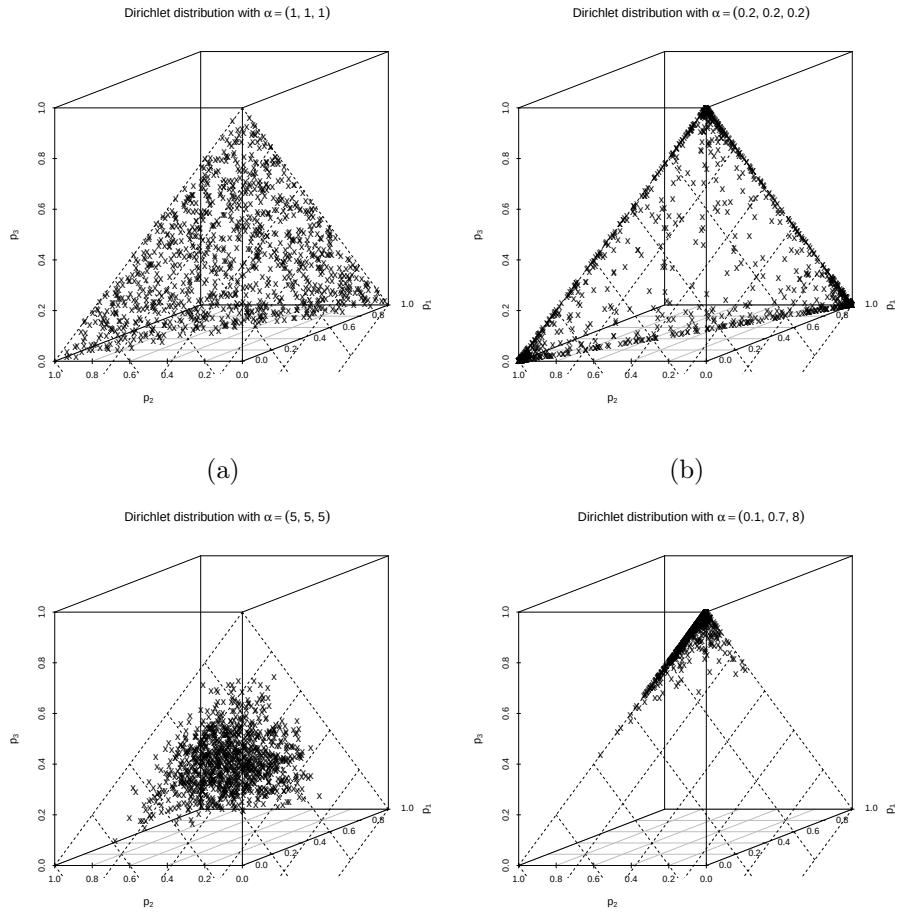
Figure 3.2: Plate notation for corpus generation according to the Latent Dirichlet Allocation topic model.

each Dirichlet distribution, and we can rewrite Equation (3.1) into

$$\text{Dirichlet}\left(\mathbf{p}|\alpha\right) = \frac{\Gamma\left(\alpha T\right)}{\Gamma\left(\alpha\right)^T} \prod_{j=1}^{T} p_j^{\alpha-1}. \tag{3.2}$$

Note that we can think of the elements of $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_K) \in \mathbb{R}_+^K$ as counts that were initially observed from a multinomial distribution, and we try to estimate the parameters $\mathbf{p}$ of the multinomial distribution with $\text{Dirichlet}\left(\mathbf{p}|\boldsymbol{\alpha}\right)$. Hence, the components of $\boldsymbol{\alpha}$ are also called pseudo-counts.

Figure 3.3 depicts a 3-dimensional Dirichlet distribution with different choices of *symmetric*, that is, $\alpha_1 = \alpha_2 = \ldots = \alpha_K$, and *asymmetric* hyperparameters. Fig. 3.3a, Fig. 3.3b, Fig. 3.3c have symmetric hyperparameters and Fig. 3.3d has asymmetric hyperparameters. Figure 3.3a shows samples of a symmetric Dirichlet distribution with $\alpha = 1$. The samples are widely distributed over all possible outcomes $\mathbf{p} \in [0,1]^3, \sum_{i=1}^3 p_i = 1$, which is called the 2-simplex. By comparison, Figure 3.3c also has symmetric hyperparameters. However, we can see that the distribution of the samples now concentrates in the middle of the simplex. This makes sense when we interpret α with pseudo-counts as mentioned above. Since we have not much evidence to estimate $\mathbf{p}$ in the case when $\alpha = 1$, the sample in Figure 3.3a are spread across the simplex. On the contrary, we can be more confident that $\mathbf{p}$ is close to $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ with increased pseudo-counts like in Figure 3.3c. Parameters less than one sharpen the distribution, which means that the distribution concentrates on the edges of the simplex, as we can see in Figure 3.3b and 3.3d. Such parameterizations can be used to favor *sparse* mixtures, where most of the components are zero,



(a) (b)

(c) (d)

Figure 3.3: Sampling $n = 1000$ times from a Dirichlet distribution with varying hyperparameters.

and just a few components are active. For example, consider Fig. 3.3b to represent the distribution over topic mixtures for documents in a corpus. Most of the documents would mainly originate from just one main active topic; the other documents would exhibit a mixture of two, or three topics, but they would tend to avoid an equal composition of all three topics. With a parameterization like in Fig. 3.3c, however, we would expect all documents to contain aspects of nearly all topics since the density in the middle of the simplex is high.

3.2 Visualization

To better understand LDA topic models and their outcomes we need visual depictions. We will refer to some available visualisation approaches in this section. Since the plain parameter results of an estimated topic model are hard to interpret by humans, there have been several suggestions for visualisation. Depending on the interests and aims of the instructor, they focus on topics, documents, or words. We will study the visualisations on abstracts of scientific papers that were crawled from ArXiv[3]. The corpus consists of 6550 documents: stop-words, punctuations, and numbers have been removed before the topic model was applied. The words have been stripped from whitespace, and converted to lower-case.

Table 3.2: The 15 most frequent terms for the topics of our ArXiv corpus.

°	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
1	field	image	model	network	system
2	magnetic	method	population	networks	paper
3	solar	algorithm	models	model	language
4	model	images	species	data	using
5	plasma	data	evolution	neurons	information
6	energy	based	dynamics	neural	based
7	electron	paper	evolutionary	brain	results
8	results	proposed	results	using	design
9	space	using	time	information	systems
10	system	methods	genetic	activity	performance
11	using	results	populations	time	processing
12	time	model	distribution	patterns	words
13	waves	approach	process	models	proposed
14	study	features	size	results	approach
15	observed	algorithms	gene	analysis	method

[3] `http://www.arxiv.org`

A straight forward approach that gives an insight into the topic-word distributions directly focuses on the corresponding vectors $\phi^{(j)}$. We only sort these vectors and then list the n most frequent terms of each topic. For example, Table 3.2 gives the top 15 terms of our ArXiv corpus for each of the 5 topics. Although the topic model cannot assign labels to the learned topics, we can clearly recognize different categories. Topic 1, for example, seems to belong to the physics domain, whereas Topic 4 deals with neural networks or cognitive neuroscience.

In some cases, we would like to focus our attention on documents and their content. For example, consider that we browse through a document collection that has been sorted with respect to some given user specified keywords. In this case, we can guide the user by colored and numbered annotations showing the topics of the words in the documents. The documents are presented in their original composition, that is, leaving the bag-of-words approach and going back to the sequential representation. Each word is mapped to its most probable topic state, which, for example, can be determined by sampling from the posterior distribution (c.f., e.g., Griffiths and Steyvers, 2004; Steyvers and Griffiths, 2007). Especially, we can use the annotations for further processing. Blei and Lafferty (2009) use nested permutation tests on the annotated documents to find multi-word expressions and relate them to the topics. Their *turbo topics* approach extracts significant n-grams of arbitrary length. Subsequently, we can exploit the extracted multi-word expressions to improve the n-best lists for the topic-term distributions as described above (Blei and Lafferty, 2009).

Early visualization approaches were developed as a side-effect of topic modeling research. Recently, their have been contributions that explicitly focus on topic model visualization. For example, Chuang et al. (2012) proposed "Termite: Visualization Techniques for Assessing Textual Topic Models". They use a tabular layout of terms and topics to let the user compare the use of terms both inside and across topics. Additionally, they propose a *saliency measure* to rank and filter terms, and a *seriation method* to sort terms and detect clusters in the data. Despite these new suggestions, Blei (2012) notes that visualization techniques and user interfaces are still one of the main points for future directions in topic modeling research.

3.3 Inference with Collapsed Gibbs Sampling

Exact inference in topic models is in general not feasible. However, good approximations are possible and we can choose between several kinds of such approaches, for instance, when Blei et al. published their paper "Latent Dirichlet Allocation" (Blei et al., 2001), they suggested a variational approach. We will not discuss variational inference methods here; instead, we will perform parameter learning as a side-effect of inference with collapsed Gibbs sampling for LDA, as contributed by Griffiths and Steyvers (2004). This approach does not estimate ϕ and θ directly; rather, it employs posterior inference focusing on $P(\mathbf{z}|\mathbf{w})$ which enables to predict ϕ and θ afterwards. Our derivation mainly

follows the thorough explanations of Gibbs sampling for LDA given by Heinrich (2009), Carpenter (2010) and Darling (2011).

As discussed in Section 2.5, we construct a Markov chain which converges to the target distribution. We iteratively sample from the chain and get assignments to the variables, which lets us estimate the parameters of the model by integrating over a few samples or just using the last sample. In the case of LDA, we want to estimate the posterior, $P(\mathbf{z}|\mathbf{w})$. We can derive the model parameters ϕ, θ for any single sample of $\mathbf{z}$ afterwards. Our Markov chain iteratively samples the hidden topic states z_i given the data $\mathbf{w}$ and all other topic states from the previous sample $\mathbf{z_{-i}}$ with

$$P(z_i = j|\mathbf{z_{-i}}, \mathbf{w}) = \frac{P(z_i = j, \mathbf{z_{-i}}, \mathbf{w})}{\sum_{k=1}^{T} P(z_i = k, \mathbf{z_{-i}}, \mathbf{w})}. \tag{3.3}$$

Remembering that we only want to sample z_i, we can disregard the denominator which stays the same for all choices $z_i = j$:

$$P(z_i = j|\mathbf{z_{-i}}, \mathbf{w}) \propto P(z_i = j, \mathbf{z_{-i}}, \mathbf{w}). \tag{3.4}$$

Hence, our focus of attention lies on the joint distribution of words and their topic states $P(\mathbf{w}, \mathbf{z}|\alpha, \beta)$ since $z_i = j, \mathbf{z_{-i}}$ is just some full assignment $\mathbf{z}$. We can simplify this term with the independence assumptions of the LDA model; according to the structure of the graphical model that is depicted in Figure 3.2, we can see that the inner plate on the right side, which represents the joint distribution of $\mathbf{w}$ and $\mathbf{z}$, can be factored into two separate terms by

$$P(\mathbf{w}, \mathbf{z}|\alpha, \beta) = P(\mathbf{w}|\mathbf{z}, \beta) \cdot P(\mathbf{z}|\alpha). \tag{3.5}$$

The first term ($P(\mathbf{w}|\mathbf{z}, \beta)$) handles the generation of the words $\mathbf{w}$ with dependence only on the current draw of topic assignments $\mathbf{z}$, the topic word distributions ϕ, and the hyperparameters β. The second term ($P(\mathbf{z}|\alpha)$) handles the generation of the topic states $\mathbf{z}$ with dependence only on θ and α. Note that we omitted the parameters ϕ, θ in Equation (3.5); we can integrate them out as we will see next.

First, we define $n_v^{(j)}$ as the number of times that the vocabulary item v is assigned to topic j, and apply this shortcut to rewrite

$$P(\mathbf{w}|\mathbf{z}, \phi) = \prod_{i=1}^{N} P(w_i|z_i, \phi) = \prod_{i=1}^{N} \phi^{(z_i)}(w_i) \tag{3.6}$$

into

$$P(\mathbf{w}|\mathbf{z}, \phi) = \prod_{j=1}^{T} \prod_{v=1}^{V} \left(\phi^{(j)}(v)\right)^{n_v^{(j)}}. \tag{3.7}$$

This step intuitively reflects the bag-of-words assumption: the generation of the words does not depend on the particular sequence as given in Equation (3.6) and can be rewritten disregarding the word order as in Equation (3.7).

Before we proceed further, we introduce a term that simplifies the upcoming equations. Similar to Darling (2011), we use the multinomial beta function

$$B(\boldsymbol{\alpha}) := \frac{\prod_{i=1}^{T} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{T} \alpha_i)} \tag{3.8}$$

for a hyperparameter vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_T)$ with length T. It reduces to

$$B(\alpha) = \frac{\Gamma(\alpha)^T}{\Gamma(\alpha T)} \tag{3.9}$$

for symmetric hyperparameters of dimension T.

Finally, we also need to know the type 1 Dirichlet integrals:[4]

$$\begin{aligned}\int\cdots\int f(\sum_{i=1}^{n} t_i) \prod_{i=1}^{n} t_i^{\alpha_i - 1} \,\mathrm{d}t_1 \,\mathrm{d}t_2 \ldots \,\mathrm{d}t_n &= \frac{\prod_{i=1}^{n} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{n} \alpha_i)} \int_0^1 f(\tau) \tau^{(\sum_{i=1}^{n} \alpha_i) - 1} \,\mathrm{d}\tau \\ &= B(\boldsymbol{\alpha}) \int_0^1 f(\tau) \tau^{(\sum_{i=1}^{n} \alpha_i) - 1} \,\mathrm{d}\tau. \end{aligned} \tag{3.10}$$

With this at hand, we are able to integrate out ϕ:

$$\begin{aligned} P(\mathbf{w}|\mathbf{z}, \beta) &= \int P(\mathbf{w}|\mathbf{z}, \phi) \cdot P(\phi|\beta) \,\mathrm{d}\phi \\ &= \int \left(\prod_{d=1}^{D} \prod_{i=1}^{N_d} P(w_i^{(d)}|\phi) \right) \left(\prod_{j=1}^{T} P(\phi^{(j)})|\beta) \right) \mathrm{d}\phi && (3.11) \\ &= \int \left(\prod_{j=1}^{T} \prod_{v=1}^{V} \left(\phi^{(j)}(v) \right)^{n_v^{(j)}} \right) \left(\prod_{j=1}^{T} \frac{\Gamma(\beta V)}{\Gamma(\beta)^V} \prod_{v=1}^{V} \left(\phi^{(j)}(v) \right)^{\beta - 1} \right) \mathrm{d}\phi && (3.12) \\ &= \prod_{j=1}^{T} \frac{\Gamma(\beta V)}{\Gamma(\beta)^V} \int \prod_{v=1}^{V} \left(\phi^{(j)}(v) \right)^{n_v^{(j)} + \beta - 1} \mathrm{d}\phi^{(j)} && (3.13) \\ &= \prod_{j=1}^{T} \frac{1}{B(\beta)} \cdot B(n_{\mathbf{v}}^{(j)} + \beta) && (3.14) \\ &= \prod_{j=1}^{T} \frac{B(n_{\mathbf{v}}^{(j)} + \beta)}{B(\beta)}, && (3.15) \end{aligned}$$

where the vector $n_{\mathbf{v}}^{(j)} = (n_v^{(j)})_{v=1}^{V}$ contains the counts for all word types restricted to topic j but across all documents. Between Equations (3.11) and (3.12) we have used Equation (3.7) and substituted the Dirichlet distribution. In the step from Equation (3.13) to (3.14), we have applied Equation (3.10).

[4] cf., for example, `http://mathworld.wolfram.com/DirichletIntegrals.html` (last checked on September 17, 2012)

Similarly, we can proceed for θ:

$$P(\mathbf{z}|\alpha) = \int P(\mathbf{z}|\theta)P(\theta|\alpha)\,\mathrm{d}\theta \tag{3.16}$$

$$= \int \left(\prod_{d=1}^{D}\prod_{i=1}^{N_d} P(z_i^{(d)}|\theta^{(d)})\right)\left(\prod_{d=1}^{D} P(\theta^{(d)}|\alpha)\right)\mathrm{d}\theta \tag{3.17}$$

$$= \int \left(\prod_{d=1}^{D}\prod_{i=1}^{N_d} \theta^{(d)}(z_i^{(d)})\right)\left(\prod_{d=1}^{D} \frac{\Gamma(\alpha T)}{\Gamma(\alpha)^T}\prod_{j=1}^{T}\left(\theta^{(d)}(j)\right)^{\alpha-1}\right)\mathrm{d}\theta \tag{3.18}$$

$$= \int \left(\prod_{d=1}^{D}\prod_{j=1}^{T} \left(\theta^{(d)}(j)\right)^{n_j^{(d)}}\right)\left(\prod_{d=1}^{D} \frac{\Gamma(\alpha T)}{\Gamma(\alpha)^T}\prod_{j=1}^{T}\left(\theta^{(d)}(j)\right)^{\alpha-1}\right)\mathrm{d}\theta \tag{3.19}$$

$$= \prod_{d=1}^{D}\frac{\Gamma(\alpha T)}{\Gamma(\alpha)^T}\int \prod_{j=1}^{T}\theta^{(d)}(j)^{n_j^{(d)}+\alpha-1}\mathrm{d}\theta^{(d)}$$

$$= \prod_{d=1}^{D}\frac{B(n_{\mathbf{j}}^{(d)}+\alpha)}{B(\alpha)}, \tag{3.20}$$

where $n_j^{(d)}$ is the number of times that a word of document d is assigned to topic j, and $n_{\mathbf{j}}^{(d)}$ is the T-dimensional vector $n_{\mathbf{j}}^{(d)} = (n_j^{(d)})_{j=1}^T$.

Equations (3.15) and (3.20) can now be used to derive

$$P(\mathbf{w},\mathbf{z}|\alpha,\beta) = \prod_{j=1}^{T}\frac{B(n_{\mathbf{v}}^{(j)}+\beta)}{B(\beta)}\prod_{d=1}^{D}\frac{B(n_{\mathbf{j}}^{(d)}+\alpha)}{B(\alpha)}. \tag{3.21}$$

At this point, we can focus on the local posterior $P(z_i = j|\mathbf{z}_{-\mathbf{i}},\mathbf{w})$ again, which is the essential part of the Gibbs sampler:

$$P(z_i = j|\mathbf{z}_{-\mathbf{i}},\mathbf{w}) = \frac{P(z_i = j,\mathbf{z}_{-\mathbf{i}},\mathbf{w})}{P(\mathbf{z}_{-\mathbf{i}},\mathbf{w})} \tag{3.22}$$

$$= \frac{P(\mathbf{w}|\mathbf{z})}{P(\mathbf{w}_{-\mathbf{i}}|\mathbf{z}_{-\mathbf{i}})P(w_i)}\cdot\frac{P(\mathbf{z})}{P(\mathbf{z}_{-\mathbf{i}})} \tag{3.23}$$

$$\propto \frac{B(n_{\mathbf{v}}^{(j)}+\beta)}{B(n_{\mathbf{v},\neg i}^{(j)}+\beta)}\cdot\frac{B(n_{\mathbf{j}}^{(d_i)}+\alpha)}{B(n_{\mathbf{j},\neg i}^{(d_i)}+\alpha)} \tag{3.24}$$

where $n_{\mathbf{v},\neg i}^{(j)}$ denotes a vector that is based on $n_{\mathbf{v}}^{(j)}$ but disregards the counts that follow from the assignment of w_i to z_i. It can be written as $n_{\mathbf{v},\neg i}^{(j)} = n_{\mathbf{v}}^{(j)} - (\underbrace{0,\ldots,0}_{1\text{ to }w_i\text{-}1},1,\underbrace{0,\ldots,0}_{w_i+1\text{ to V}})$.
The term $n_{\mathbf{j},\neg i}^{(d_i)}$ is defined with similar semantics, that is, $n_{\mathbf{j},\neg i}^{(d_i)} = n_{\mathbf{j}}^{(d_i)} - (\underbrace{0,\ldots,0}_{1\text{ to }z_i\text{-}1},1,\underbrace{0,\ldots,0}_{z_i+1\text{ to T}})$.
Remember that d_i represents the document that contains w_i, z_i; note its difference to d which is mostly used as an indexed variable in sums or products.

In the next step, we further expand Equation (3.24) with the definition of the multinomial Beta function from Equation (3.8) to get

$$P(z_i = j|\mathbf{z_{-i}}, \mathbf{w}) \propto \frac{\Gamma(n_{w_i}^{(j)} + \beta)\Gamma(\sum_{v=1}^{V} n_{v,\neg i}^{(j)} + \beta)}{\Gamma(n_{w_i,\neg i}^{(j)} + \beta)\Gamma(\sum_{v=1}^{V} n_v^{(j)} + \beta)} \cdot \frac{\Gamma(n_j^{(d_i)} + \alpha)\Gamma(\sum_{k=1}^{T} n_{k,\neg i}^{(d_i)} + \alpha)}{\Gamma(n_{j,\neg i}^{(d_i)} + \alpha)\Gamma(\sum_{k=1}^{T} n_k^{(d_i)} + \alpha)} \tag{3.25}$$

and obtain

$$= \frac{n_{w_i,\neg i}^{(j)} + \beta}{\sum_{v=1}^{V}\left(n_v^{(j)} + \beta\right)} \cdot \frac{n_{j,\neg i}^{(d_i)} + \alpha}{\sum_{k=1}^{T}\left(n_k^{(d_i)} + \alpha\right)} \tag{3.26}$$

$$\propto \frac{n_{w_i,\neg i}^{(j)} + \beta}{\sum_{v=1}^{V}\left(n_v^{(j)} + \beta\right)} \cdot \left(n_{j,\neg i}^{(d)} + \alpha\right). \tag{3.27}$$

In the step from Equation 3.25 to Equation 3.26, we have used a special property of the gamma function

$$\Gamma(x+1) = x \cdot \Gamma(x) \tag{3.28}$$

which lets us rewrite the terms in Equation (3.25) for cancellation. For example, we write $\left(n_{w_i,\neg i}^{(j)} + \beta\right) \cdot \Gamma(n_{w_i,\neg i}^{(j)} + \beta)$ for $\Gamma(n_{w_i}^{(j)} + \beta)$, which is $\Gamma(n_{w_i,\neg i}^{(j)} + 1 + \beta)$, in the nominator of the first fraction and then cancel $\Gamma(n_{w_i,\neg i}^{(j)} + \beta)$ with the identical term in the denominator. In Equation (3.27), we left out the term that sums over all topics since it remains the same for all possible topic assignments of z_i and therefore does not affect sampling.

To point this out, we have finally arrived at the equation for sampling from the posterior

$$P(z_i = j|\mathbf{z_{-i}}, \mathbf{w}) \propto \frac{n_{w_i,\neg i}^{(j)} + \beta}{\sum_{v=1}^{V}\left(n_v^{(j)} + \beta\right)} \cdot \left(n_{j,\neg i}^{(d)} + \alpha\right). \tag{3.29}$$

At each point of the Markov chain, we can retrieve estimates for the parameters ϕ and θ by

$$\hat{\phi}^{(j)}(v) = \frac{n_v^{(j)} + \beta}{\sum_{v=1}^{V}\left(n_v^{(j)} + \beta\right)} \qquad \text{and} \tag{3.30a}$$

$$\hat{\theta}^{(d)}(j) = \frac{n_j^{(d)} + \alpha}{\sum_{k=1}^{T}\left(n_k^{(d)} + \alpha\right)}. \tag{3.30b}$$

Heinrich (2009) derives these equations by applying Bayes rule to Equation (3.7) for ϕ and the corresponding formula for θ. This yields

$$P(\phi^{(j)}|\mathbf{w}, \mathbf{z}, \beta) = \frac{1}{Z_{\phi^{(j)}}} \prod_{i=1, z_i=j}^{N} P(w_i|\phi^{(j)}) P(\phi^{(j)}|\beta) = \text{Dirichlet}(\phi^{(j)}|n_{\mathbf{v}}^{(j)} + \beta), \tag{3.31a}$$

$$P(\theta^{(d)}|\mathbf{w}, \mathbf{z}, \alpha) = \frac{1}{Z_{\theta^{(d)}}} \prod_{i=1}^{N_d} P(z_i^{(d)}|\theta^{(d)}) P(\theta^{(d)}|\alpha) = \text{Dirichlet}(\theta^{(d)}|n_{\mathbf{j}}^{(d)} + \alpha) \tag{3.31b}$$

for some state $\mathbf{w}, \mathbf{z}$ of the chain. Equations (3.30a) and (3.30b) then follow by the mean of the Dirichlet distribution:

$$\mathbb{E}[\text{Dirichlet}(\cdot|\boldsymbol{\gamma})] = \frac{1}{\sum_{i=1}^{S} \gamma_i}(\gamma_1, \ldots, \gamma_S)$$

for some hyperparamter vector $\boldsymbol{\gamma}$ of dimension S.

In Section 2.6, we touched collapsed Gibbs sampling. Looking back at the Gibbs sampling procedure given above, we can see that we integrated out ϕ and θ and only sample the topic state variables z_i. Hence, we have seen an example of a collapse Gibbs sampling algorithm in this section.

Especially when we process big data, algorithms have to be efficient. Contrary to some other clustering algorithms like, for example, Brown clustering, both common inference algorithms for LDA, that is, Gibbs sampling and Variational inference, depend only linearly on the number of topics (c.f., Chrupala, 2011).

3.4 Model Selection and Evaluation

A recurring question in applying machine learning algorithms is the configuration of options and arguments. Many clustering algorithms need a predefined number of clusters that is expected to be present in the data, thresholds and biases have to be configured. LDA also has arguments that need to be specified initially and these can drastically affect the quality of the results. We have to choose the number of topics and the bias parameters α, β of the Dirichlet priors.

The simplest approach to choose parameters is data-driven. We try out a grid of different parameter values and measure the quality of the resulting model according to some evaluation measure. Finally, we take the model with the best results. The data-driven evaluation leaves some freedom for choosing an appropriate performance measure; we will briefly touch two approaches. A typical evaluation measure for topic models is the *perplexity* which has traditionally been used in the Natural Language Processing and computational linguistics research community. The motivation behind this approach is that a good model should assign a high probability to generated content. According to Blei et al. (2003b), the perplexity of an held-out text corpus D_{test} for LDA is given by

$$\text{perplexity}(D_{\text{test}}) = \exp\left\{-\frac{\sum_{d=1}^{D} \mathbf{w}_d)}{N}\right\}.$$

Computing this term for topic models is challenging and requires approximate solutions. Wallach et al. (2009), however, argue that many commonly used methods to predict the probability of held-out data for a given topic model are inaccurate. They give a detailed discussion of this issue and propose two novel methods. Another approach to assess the quality of topic models installs them into an extrinsic target system and evaluation is

then based on metrics of the target task. For example, we can use LDA topic models as a part of a recommender system. In this case, we can grade topic models according to the performance of the recommender.

Note that we should not evaluate models on the same data which has been used to adjust the parameters; we want to know how well a model generalizes on new data rather than how well it can overfit on given data. Therefore, we have to split the data into a training and a test set[5]. The training data is used to train the model, the test data is used to assess the generalization performance of the model. That is, we compute the perplexity of the models on the test data set. Figure 3.4 illustrates how the number of

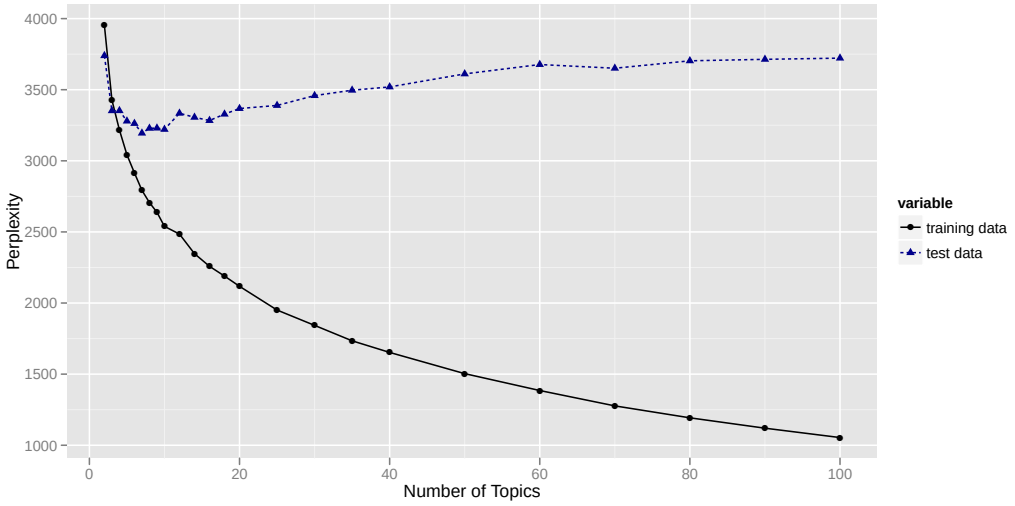


Figure 3.4: Determining the optimal number of topics for a LDA topic model with a data driven approach.

topics for a topic model can be selected with a data-driven approach. We can clearly see that the perplexity of the model on the training data is not a good indicator for the quality of the model. It consistently decreases with an increasing number of topics. The perplexity of the model on the test data, however, does not follow this direction. At first, we also observe a decrease in perplexity which means that the model gets more accurate. Approximately at $k = 8$ we observe a minimum in the perplexity on the test data. This is the optimal choice for the number of topics. More topics lead to overfitting on the training data and diminishing generalization performance.

The data driven approach has a considerable draw-back. It can be expensive to search the parameter space for appropriate values. If we cannot guess a proper range of possible

[5]Ideally, the comparison of different methods is based on a trifold setting: the *training set*, the *development test set*, and the *evaluation test set*. The training set and the development test set are used to configure the parameters. The evaluation test set is used for the final results.

values the costs can prevent practical implementations. When, however, we have background knowledge that constrains the search space, the data-driven approach provides a simple and convenient solution to the model selection problem.

Another approach is given by Blei et al. (2003a) who suggest to use the *Nested Chinese Restaurant Process* which is a *Bayesian Non-parametric* approach to hierarchical topic modeling. The term *non-parametric* states that this model automatically grows with the data and model selection is therefore a part of the inference procedure.

4 Sequential Syntax Models

In the last section, we have discussed topics and disregarded the sequential properties of words in natural language text documents. In this section, we turn to English *syntax*, which is the analysis of sentence structure. Especially, we will focus on the well-known categorisation of words into latent morpho-syntactic classes, the Part-of-Speech (POS) classes. After a short linguistically motivated discussion of the POS classes, we will study Hidden Markov Models and Conditional Random Fields as two approaches to model shallow linguistic structure.

4.1 Parts-of-Speech

The study of language has a long history, and so have the Part-of-Speech categories, especially the 8 fundamental classes: adverb, article, conjunction, noun, participle, preposition, pronoun, and verb. These classes still rank among the basic POS classes of contemporary English. Nowadays, English tag sets range from approximately 50 classes to more than 100 classes. While linguists typically agree in respect of fundamental categories, fine-grained Part-of-Speech categories vary across corpus analyses. Common tagsets for English are the tagset of the Brown corpus (Francis and Kucera, 1979), and the Penn treebank tagset (Marcus et al., 1993), which corresponds to the Penn Treebank corpus. For a better understanding of the POS tagging task, we will shortly recapitulate some important POS classes (cf. Jurafsky and Martin, 2009, pp. 158-164) before we discuss aspects of the Part-of-Speech tagging task.

Part-of-Speech Classes

At the very outset, we identify two POS classes that interrelate, the *determiner* and the *noun*. The former is a *closed class* which means that there is finite number of terms in this class. We can collect all members of a closed POS class with a large and comprehensive corpus. For instance, the class of determiners contains three articles (the, a, an) and some other determiners (e.g., this, these, etc). On the contrary, the class of nouns is an *open class*; even if we processed a huge corpus, there would still be some unobserved nouns left. Besides, new nouns emerge over time. This is mostly caused by the semantics of the noun class because nouns basically reference people, places, and objects. The noun class is, however, not defined through its meaning, and there are nouns that are more abstract than people, places, or objects. Instead, nouns are words that may be

preceded by determiners, may take possessives, and – with some exceptions – may be morphologically transformed into plural forms. Examples of nouns with their context, are *a cat, an elephant, a mouse, the cat, the elephant, the mouse, the cat's, the elephant's, the mouse's, cats, elephants, mice.* We can further distinguish two types of nouns. Nouns that can be enumerated and occur in plural form as above are called *count nouns.* Nouns that do not take plural forms, like *snow, information*, etc, are called *mass nouns.* They can appear without article in their singular form which is not possible for count nouns.

The nouns given as examples are *common nouns. Proper nouns* are a separate POS class and refer to particular entities, which can be persons, organizations or something else. They act like common nouns, however, they typically omit determiners and are mostly printed capitalized. For example, *Apple*, *Cupertino* and *iPad* are all proper nouns in the following example.

(2) *Yesterday , Apple announced the new iPad in Cupertino .*
NN , NNP VBD DT JJ NNP IN NNP .

Another open POS class is the *adjective* which is used to describe attributes of people or things. Examples from this category with their context are *the fast car, the red car*, or *new* in Example (2). Adjectives often exist in different morphological forms for *comparative* and *superlative*, for instance, *new, newer, newest.*

The *verb* category is an open POS class that literally brings action into language; verbs express that something has happened, happens or will happen, etc. or a state of being. Verbs are further distinguished and characterized by certain morphology which indicates properties such as tense and number. For example, in Example (2) we observe the past tense form (VBD) of the verb *to announce.* The forms of verbs also include *progressive* and *past participle.*

A vary flexible category is the *adverb* which summarizes words that modify something. It is an open class and its words can take diverse semantic functions which define several subcategories. *Directional adverbs* or *locative adverbs* (e.g., here, there, up, down) concern spatial relations. The abstract intensity of some action can be described by *degree adverbs* (e.g., very, totally, hardly). *Manner adverbs* can also be used to describe the extent of an action, for example, slowly or loudly. However, they have more specific semantic meanings that describe the way how something happens. Temporal aspects are expressed by *temporal adverbs* (e.g., tomorrow, Tuesday). Similar to adjectives, adverbs also have *comparative* and *superlative* forms.

Inside sentences, words, or rather sequences of POS tags, are often structured into larger groups with similar composition. These groups are called *phrases.* For the following POS classes we especially need to know the *noun phrase* concept. In the simplest case, a noun phrase can be a single pronoun, proper noun, or a mass noun. More complicated noun phrases arise from the composition of determiners, adverbs, adjectives, nouns, and other classes that we have not discussed so far.

An example of a noun phrase is

(3) *all* *the* *awfully* *funny* *jokes*
PDT DT RB JJ NNS

The last example contains a class the we have not introduced before, the *predeterminer*. Predeterminers appear at the beginning of noun phrases and often address cardinality or amount.

When we want to refer to a noun phrase we can use *pronouns*, a closed POS class. *Personal pronouns* address people or entities (e.g., I, you, he, me). They have *possessive pronoun* forms that indicate actual or abstract possession relations (e.g., my, your, his). In questions we use *Wh-pronouns* (e.g., what, who).

(4) Pronoun (PRP) and wh-pronoun (WP)

a. *I* *love* *girls* *who* *make* *me* *laugh* *.*
PRP VBD NNS WP VBP PRP IN .

b. *Who* *do* *you* *think* *you* *are* *talking* *to* *?*
WP VBP PRP VB PRP VBP VBG TO .

Wh-pronouns can also be used as *complementizers* as in Example (4a).

The semantic relation of a noun phrases can be indicated by a preceding *preposition* which often addresses spatial or temporal aspects. For instance,

(5) *Hans* *arrived* *lately* *from* *his* *journey* *to* *India* *with* *Laura* *.*
NNP VBD RB IN PRP$ NN TO NNP IN NNP .

contains the prepositions (IN) *from* and *with*.

Particles are very similar to prepositions. However, the interaction between particle and verb is stronger than between preposition and verb. Moreover, they often differ in semantic aspects. The meaning of particles is often determined by the combination of a verb and a particle whereas the meaning of prepositions is rather literally and independent of verbs. Consider the following sentence.

(6) *Finally* *,* *the* *plane* *took* *off* *.*
RB , DT NN VBD RP .

In contrast to example (5), *off* is linked to *took* here stronger than *with* to *arrived*. When verb and particle form such a strong syntactic and semantic unit as *took off* in Example (6), we call the verb a *phrasal verb*.

Sentences, phrases and clauses can be connected by words of the *conjunction* class. Either coordinating, that is, on the same level (e.g., and, or, but), or subordinating, that

is, with a distinctive direction (e.g., if, after, although). The latter are called *complementizers* when they connect the main verb to its argument. Sometimes tag sets have a special tag for coordinating conjunctions but count subordinating conjunctions to the class of prepositions. Conjunction, as well as the above-mentioned classes predeterminer, preposition, and particle are all closed POS classes.

As noted before, we focus on English grammar. All of the above can be different in other languages.

Part-of-Speech Analysis

Before we take a look on different elaborate approaches to automatically induce and assign POS tags it is reasonable to look at some simple statistical aspects of the Parts-of-Speech and think about how we can assess the quality of Part-of-Speech taggers, how we can interpret these measures, how well a naive approach can perform, and what maximum performance we can expect to reach.

A simple measure for POS tagging performance is the accuracy:

$$acc = \frac{n_{\hat{c}=c}}{n} \tag{4.1}$$

where n is the total number of words in the test set and $n_{\hat{c}=c}$ is the number of words where the predicted tag $\hat{c}$ matches the target tag c from the gold standard test set. When we interpret the accuracy we have to keep in mind that the accuracy includes all punctuation where tagging is trivial. Interestingly, there are a lot of words that can easily be tagged by a crude heuristic, the standard *baseline* for POS tagging: we simply assign each word to its most frequent class according to the training corpus. On some corpora, this simple tagger can yield up to 90% accuracy (Charniak et al., 1993) because most word types only have a few possible tags and one dominant tag. The challenges of Part-of-Speech tagging are therefore *disambiguation* between different tags, and *generalization* on new word types. For this reason, we often measure the accuracies on known word types and OOV terms separately.

Different state-of-the-art taggers with supervised approaches report accuracies of approximately 97% for simple tag sets when they are trained on a large number of labeled examples and the test data is similar to the training data. Given that human annotators also aggree on about 97% of the tags (Marcus et al., 1993) we see that these classifiers reach the human *ceiling* for this task, the best performance that we can expect to reach. It is, however, still a problem to maintain the performance of the tagger when the data distribution changes, especially when we deal with a high OOV-rate, or when just a little amount of training data is available. Both problems require learning robust models and adapting to new observations.

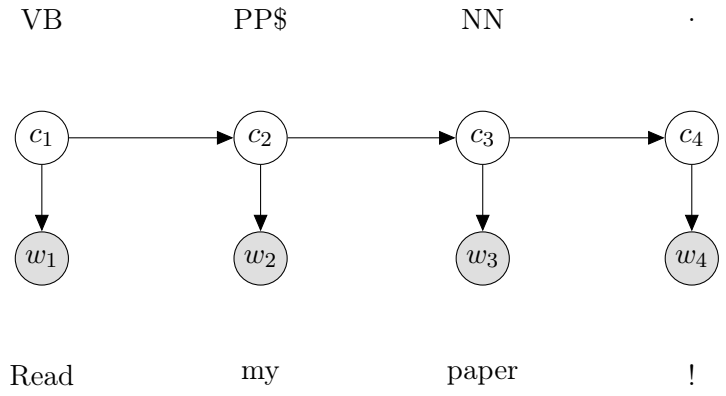


Figure 4.1: Rolled out Hidden Markov Model for POS tagging.

4.2 Hidden Markov Models for Part-of-Speech Tagging

Hidden Markov Models (HMMs) are a special type of generative graphical models (cf. Section 2). They belong to the class of *Dynamic Bayesian Networks* which model recurring structures in sequential processes. They have been widely applied to machine learning problems in several text processing tasks, speech processing, computer vision, robotics, or bioinformatics. Since the 1960s, they still yield state-of-the-art results in many sequence labeling tasks. A good tutorial on HMMs is (Rabiner, 1989).

The characteristic properties of HMMs are a two layer architecture and sequential dependencies in one layer, which is assumed to represent latent states. The variables of the other layer represent observations where each of the observation variables is independent of all other variables given its latent state. The latent, or hidden states have sequential first-order dependencies as depicted in Figure 4.1 for Part-of-Speech tagging. In that example, the nodes of the graph are divided into observed words w_i, and hidden POS tags c_i. The observed variables are assumed to be generated by the latent variables, and the latent variables are assumed to be generated by their preceding latent variable. The model parameters ϕ, π determine the probabilities $P(w_i|c_i)$ and $P(c_i|c_{i-1})$, and are called *emission* probabilities and *transition* probabilities respectively. The joint probability of a sentence of length n with words w_i and tags c_i is given by

$$P(w_1, \ldots, w_n, c_1, \ldots, c_n) = \prod_{i=1}^{n} P(w_i|c_i) \cdot P(c_i|c_{i-1}) \tag{4.2}$$

$$= \prod_{i=1}^{n} \phi^{(c_i)}(w_i) \cdot \pi^{(c_{i-1})}(c_j) \tag{4.3}$$

where c_0 denotes an artificial initial state.

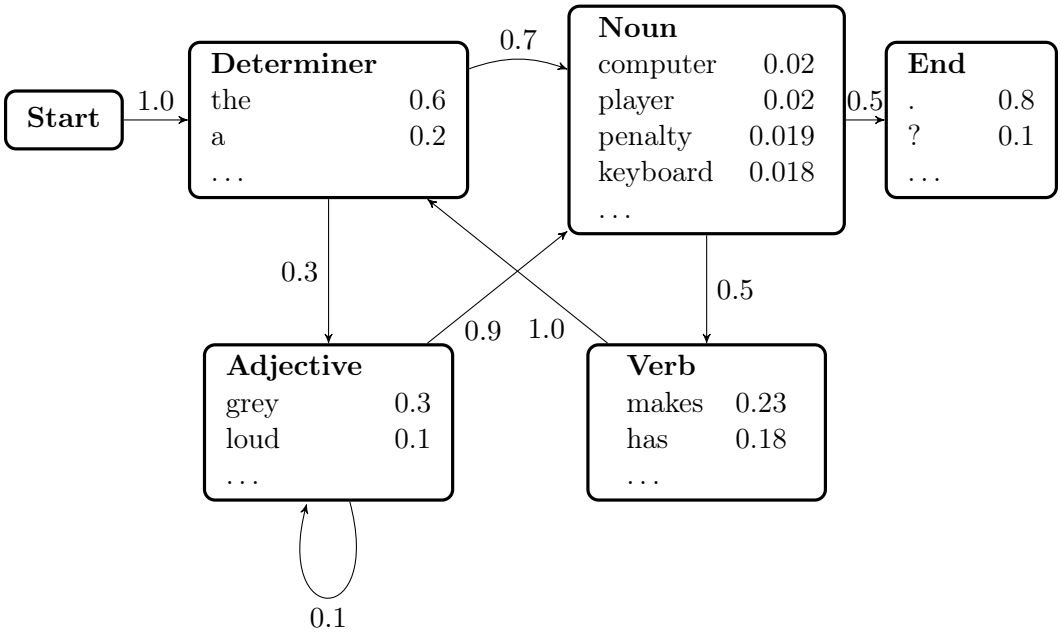


Figure 4.2: Weighted Finite State Machine of a very simple Hidden Markov Model for POS classes and words.

We can see that in contrast to models with static graph structure HMMs build their graphs dynamically. We *unroll* the graphs with a varying number of nodes and edges, dependent on the input. Figure 4.1 depicts the HMM structure for a simple sentence. Here, we discuss only the case that every hidden variable is only directly dependent on its immediate predecessor. This is called the first-order Markov assumption. HMMs can naturally be extended to support second, third, or higher order dependencies by reduction to a first-order model with an increased number of possible hidden states. First-order models are typically too simple to capture the dependencies inherent in language. They miss linguistic cues that range over more than one word. Higher-order Markov models, however, have to fit notably more parameters. That is why increasing the Markov order does not only increase computation time, but also can decrease tagging accuracy when there is not enough data to estimate the parameters of the model correctly. A common implementation strategy involves linear interpolation between first, second, and third order Markov assumptions (cf., for example, Brants, 2000).

An equivalent formulation of HMMs builds upon *Weighted Finite State Machines (WFSMs)*. In short, a weighted finite state machine consists of a set of distinct states, transition probabilities between the states, emission symbols, emission probabilities, and certain start and end states. The machine starts at the start state. Subsequently, it iteratively changes its state according to the transition probabilities. At every discrete time step, it emits a symbol according to the emission probabilities. The emission proba-

bilities depend on the current state that the machine occupies at that time step. We can clearly see the analogues to the HMM formulation given above. The states correspond to the hidden variables, the emission symbols correspond to the values of the observed variables, and the transition probabilities and the emission probabilities appear in both models. In Figure 4.2 we see a weighted finite state machine of a very simplistic model for sentence generation with POS classes and words. States are depicted as boxes; the names of the states are printed in bold. For each state, the most probable emissions are printed below the names together with their emission probabilities. Arcs indicate possible state transitions and the numbers near the arcs correspond to the transition probabilities. When two states are not connected by an arc, the corresponding transition probabilities are zero.

(7) Sentences generated by the Weighted Finite State Machine of Figure 4.2

a. *The computer has a grey keyboard .*
DT NN VB DT JJ NN .

b. **A loud keyboard makes the grey loud loud penalty .*
DT JJ NN VB DT JJ JJ JJ NN .

The sentences in Example (7) are generated by the Weighted Finite State Machine of Figure 4.2. Example (7a) shows that this Weighted Finite State Machine is able to produce correct English sentences. However, it also emits sentences that are grammatically correct but do not make sense as shown in Example (7b).

Remembering the discussion of generative graphical models in general from Section 2, we already know that there are several ways how we can fit parameters for HMMs. The standard ways to estimate the emission and transition probabilities for a Hidden Markov Model are Maximum Likelihood or Maximum Aposteriori estimation. That is, we take a fixed parameter setting that maximizes the likelihood of the parameters given the observed data. Efficient dynamic programming algorithms that exploit the linear chain structure of the model exist for this type of computation. The Baum-Welch algorithm is widely used for unsupervised learning of HMMs and the Viterbi algorithm for inference. The parameters are estimated first, then the hidden states are inferred with the fixed parameter values. In contrast to this procedure, it is also possible to leave parameters uncertain and to determine the hidden states by integrating over the parameters. Goldwater and Griffiths (2007) have shown that this fully Bayesian approach to unsupervised POS tagging can yield substantial improvements against MLE or MAP estimation. Therefore, we will take a closer look on their approach on the following pages.

Fully Bayesian HMM

According to Goldwater and Griffiths (2007), we assume the standard HMM structure as explained above, however, we now base computation on a second order Markov assump-

tion and add symmetric Dirichlet priors γ, δ over the transition distribution π and the output distribution ϕ, respectively. Essentially, the new priors are introduced to enforce *sparse* distributions as they are typically found in linguistic structures. Especially for POS tagging, we already mentioned above that most words tend to belong just to a few POS classes. Similarly, POS classes also favor sparse transition probabilities.

We arrive at a generative model that samples according to

$$\begin{array}{rll} c_i & \mid c_{i-1} = c,\, c_{i-2} = c',\, \pi & \sim \text{Mult}(\cdot|\pi^{(c,c')}) \\ w_i & \mid c_i = c,\, \phi & \sim \text{Mult}(\cdot|\phi^{(c)}) \\ \pi^{(c)} & \mid \gamma & \sim \text{Dirichlet}(\cdot|\gamma) \\ \phi^{(c)} & \mid \delta & \sim \text{Dirichlet}(\cdot|\delta). \end{array}$$

The hyperparameters γ, δ can be used to control the sparsity bias of the model. With $\gamma < 1$ we can promote sparse multinomials for the transition distributions $\pi^{(c,c')}$; with $\delta < 1$ we can achieve the same effect for the word emission distributions $\phi^{(c)}$. As above, c_i and w_i again stand for the ith tag and the ith word in the sequence. Let C denote the size of the tag set and V as before the vocabulary size.

Now we integrate over the parameters to assign the hidden variables, which yields

$$P(c_i|\mathbf{c}_{-i}, \gamma) = \frac{n_{(c_{i-2},c_{i-1},c_i)} + \gamma}{n_{(c_{i-2},c_{i-1})} + C\gamma}, \tag{4.4}$$

$$P(w_i|c_i, \mathbf{c}_{-i}, \mathbf{w}_{-\mathbf{i}}\delta) = \frac{n_{(c_i,w_i)} + \delta}{n_{(c_i)} + W_{c_i}\delta} \tag{4.5}$$

where $n_{(c_{i-2},c_{i-1},c_i)}$ and $n_{(c_i,w_i)}$ are counts for the corresponding tag sequences and emissions of the word type w_i from the tag c_i.

Goldwater and Griffiths (2007) advocate to use Gibbs sampling for inference. That is, we sample iteratively each tag from the posterior $P(\mathbf{c}|\mathbf{w}, \gamma, \delta) \propto P(\mathbf{w}|\mathbf{c}, \delta)P(\mathbf{c}|\gamma)$ as we have already discussed in Section 2. The exact sampling equations can be found in (Goldwater and Griffiths, 2007).

4.3 Conditional Random Fields for Part-of-Speech Tagging

All of the methods that we have considered so far approximate the joint distributions $P(X, Y)$[1] of the observed variables and the hidden variables; classification with conditional probabilities $P(Y|X)$ has been derived through Bayes rule. Hence, we have spent some amount of work into estimating the distribution of observed variables $P(X)$. This can be necessary sometimes, for example, if we want to generate artificial sequences from

[1] In this section we use the X, Y notation for observed variables and latent variables, which is widely used in the CRF literature. Compared to the last section, X corresponds to the words $\mathbf{w}$ and Y to the syntactic states $\mathbf{c}$.

that distribution. A trained Hidden Markov Model for Part-of-Speech tagging, or a syntax model based on a probabilistic context-free grammar can be used to create artificial sentences which follow the same distribution that the model was trained on. Thus, these models are termed *generative* models. The counterpart of generative models are *discriminative*, or *conditional*, models. Models of this type focus purely on classification, for example, by estimating the conditional distribution $P(Y|X)$. They cannot intuitively be used to generate artificial sequences that follow the input distribution. However, they often yield better classification results. Since discriminative models are more robust against dependencies in the input variables, they can incorporate more complex structures and more properties of the input. For POS tagging, such *features* can highlight if a word is capitalized, if it ends with a certain character n-gram, or if it is preceded by a special word type. Such information can be crucial for classification, and especially for the generalization performance of the classifier. Remember for example the central problem of POS tagging: the tagging of unknown word types. Sometimes we can simply guess the right tag with morphological knowledge. If the word ends with "-ly", it might be an adverb. If it ends with "-ns", it might be a noun in its plural form. Background knowledge of this type can often be easier integrated into discriminative models than into generative models since we need to add additional dependencies in the input variables for generative models, as we can see in (Brants, 2000) for POS tagging and incorporating capitalization. In this section, we briefly summarize Conditional Random Fields (CRFs) (Lafferty et al., 2001) which model conditional probabilities with undirected graphs.

Conditional Random Fields

Conditional Random Fields (CRFs) (Lafferty et al., 2001) have especially been applied to sequence labeling tasks like information extraction, named entity recognition, chunking or POS tagging where they they take a special form. Fitting to the sequential nature of these tasks, this form is shaped like a chain, similar to the HMMs in the last section, and such CRFs are called *linear chain CRFs*. More formally, we assume a sequence of tokens $\mathbf{x} = (x_1, \ldots, x_T)$ as input, binary feature functions $f_1, \ldots, f_K$, and a parameter vector $\lambda = (\lambda_1, \ldots, \lambda_K) \in \mathbb{R}^K$. We compute the conditional probability of the sequence of hidden variables $\mathbf{y} = (y_1, \ldots, y_T)$ given evidence $\mathbf{x}$ by

$$P_{\boldsymbol{\lambda}}(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_{\mathbf{x}}} \exp\left(\sum_{t=1}^{T}\sum_{i=1}^{K} \lambda_i \cdot f_i(y_{t-1}, y_t, \mathbf{x}, t)\right), \tag{4.6}$$

which is normalized by

$$Z_{\mathbf{x}} = \sum_{\mathbf{y}' \in Y} \exp\left(\sum_{t=1}^{T}\sum_{i=1}^{K} \lambda_i \cdot f_i(y'_{t-1}, y'_t, \mathbf{x}, t)\right). \tag{4.7}$$

In Equation (4.7), Y is the set of all possible label sequences $\mathbf{y}'$ for $\mathbf{x}$.

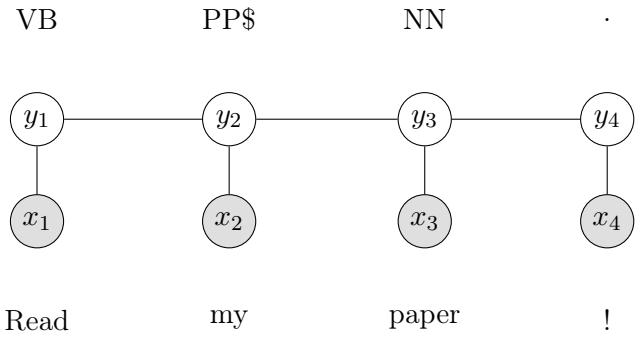


Figure 4.3: Rolled out linear-chain Conditional Random Field for Part-of-Speech tagging.

In this work, we make a further simplifying assumption that is commonly made. We assume that we just have two distinct types of feature functions: *transition feature functions* and *state feature functions*. The former are the discriminative analogues to the transition probabilities of HMMs. They all take the form

$$f_i^{(\text{trans})}(y_{t-i}, y_t) = \mathbf{1}_{\{y_{t-1}=y_a\}} \cdot \mathbf{1}_{\{y_{t-1}=y_b\}} \tag{4.8}$$

where y_a, y_b are some label types out of the label set. Thus, they indicate if a certain transition has occurred. On the contrary, *state feature functions* only depend on *one* hidden state, and possibly some context of the observation for that state. State feature functions are parametrized as $f_i^{(\text{state})}(y_t, \mathbf{x}, t)$. With this modifications the conditional probability can now be written as

$$P_{\boldsymbol{\lambda}}(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_{\mathbf{x}}} \exp\left(\sum_{t=1}^{T}\left(\sum_{i=1}^{K^{(\text{trans})}} \lambda_i^{(\text{trans})} f_i^{(\text{trans})}(y_{t-i}, y_t)\right) + \left(\sum_{i=1}^{K^{(\text{state})}} \lambda_i^{(\text{state})} \cdot f_i^{(\text{state})}(y_t, \mathbf{x}, t)\right)\right), \tag{4.9}$$

where $K^{(\text{state})}$ is the number of state feature functions and $K^{(\text{trans})}$ is the number of transition feature functions.

The state feature functions indicate if some specific event has occurred. Mostly, they only depend on the token x_t at the position t and the label that y_t takes for some specific t. In this case they factor into

$$f_i^{(\text{state})}(y_t, x_t) = \mathbf{1}_{\{y_t=y_a\}} \cdot f_i(x_t). \tag{4.10}$$

The function $f_i(x_t)$ indicates lexical properties of the token at the position t. This can be capitalization, certain n-gram suffixes, prefixes, the length, or word list membership. Together with the weights λ_i, the amount of evidence for the labels given the observed

input sequence is collected. When $f_i(y_t, \mathbf{x}, t)) = 1$ and λ_i is a strong weight, the feature function f_i supports that this label of x_t is correct. Put in different words, feature functions are neutral; they only show if some condition is satisfied for a certain configuration of input and output, or if this condition is not satisfied. The parameters λ_i then give value to this information. They determine how the information given by the feature functions affects the inference of the labels.

Figure 4.3 shows a linear-chain Conditional Random Field rolled out for a short example sentence. In contrast to the Hidden Markov Model in Figure 4.1 we now have undirected edges in the graph structure. We focus on the conditional probability of the tags given the words and certain features of the input instead of modeling the joint distribution of words and tags where the words are generated from the hidden states. The sequential structure, however, is similar between HMMs and linear-chain CRFs.

There are also parallels between linear-chain CRFs and HMMs in the way inference can be carried out to find the most likely labeling for a given observation, that is, $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x})$. In both cases, we can use dynamic programming algorithms like, for example, the Viterbi algorithm for HMMs (Rabiner, 1989). The parameters $\boldsymbol{\lambda}$ of linear-chain CRFs are typically determined by (conditional) Maximum Aposteriori estimation from data $\mathcal{D} = \left((\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\right)_{i=1,\ldots,N}$ with independently and identically distributed examples. The goal is to find

$$\hat{\boldsymbol{\lambda}}^{(\mathrm{MAP})} = \arg\max_{\boldsymbol{\lambda}} \; \mathcal{L}(\boldsymbol{\lambda}) + P(\boldsymbol{\lambda}) \tag{4.11}$$

$$= \arg\max_{\boldsymbol{\lambda}} \; \log \prod_{i=1}^{N} P_{\boldsymbol{\lambda}}(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) \quad + P(\boldsymbol{\lambda}) \tag{4.12}$$

$$= \arg\max_{\boldsymbol{\lambda}} \; \sum_{i=1}^{N} \log P_{\boldsymbol{\lambda}}(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) \quad + P(\boldsymbol{\lambda}) \tag{4.13}$$

where $P(\boldsymbol{\lambda})$ is a regularization term, for instance, a Gaussian prior. In the last equation, we use the fact that maximizing the log-likelihood yields equivalent results as optimizing against the likelihood of the parameters. The regularized log-likelihood of Equation (4.13) can be efficiently optimized by quasi-Newton methods, for example, LBFGS (Nocedal, 1980). We skip the details of inference and parameter learning for CRFs here and refer to Sutton and McCallum (2010).

5 Topic Aware Syntax Models

Most natural language processing applications can be divided into two distinct fields. On the one hand, there are applications that view documents as a bag-of-words, ignoring the word order and focusing on the semantics of whole documents. These models are applied in Information Retrieval and fit well to process huge amounts of text data. The main goal here is to reduce the dimensionality of the word distribution space but to retain similarity between documents. Documents are projected into a low dimensional space which can be seen as the topic space. Probabilistic Latent Semantic Indexing (pLSI) (Hofmann, 1999) and LDA (Blei et al., 2001) are two examples of methods applied in this field. We have been discussing LDA topic modeling in Section 3. On the other hand, there are tasks like Part-of-Speech tagging, Named Entity Recognition or Parsing that take care of the sequential and hierarchically structured word order in text. Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs) (Lafferty et al., 2001) are two examples of state-of-the-art techniques in this domain. We have introduced them in Section 4. Naturally the question arises why both fields concentrate on just one aspect of language and disregard the other when they both operate on the same language. There is at least one obvious reason to do so: the decrease in computational complexity. In general, it is beneficial to decouple tasks if possible. It prunes the search space and reduces the number of parameters to fit. However, there have also been efforts to build combined systems of topics and syntax (Griffiths et al., 2004) and empirical success in using unsupervised syntactic-semantic models as features for supervised models (Li and McCallum, 2005) motivates further research in this direction.

The structure of this chapter is as follows. We start with arguments to unite models of syntax and semantics and subsequently describe the approach of Griffiths et al. (2004).

5.1 Topics and Syntax

Let us recapitulate Section 3 where we have illustrated topics from a topic model in Figure 3.2. The terms of the topics seem expressive and informative. Such interpretable and meaningful terms, however, are only obtained when so-called *stop words* (e.g., the, of, and, to) are removed in a pre-processing step. Interestingly, stop words are typically words from closed word classes such as articles (a, the), prepositions (by, in, of), conjunctions (and, or), et cetera. Words of this type are sometimes called *function words* and characterized by three common properties: they are short, they occur frequently, and

they contribute to structure the words of the sentence. That is, they have a rather syntactic than topical nature. The last aspect is especially interesting in the context of the connection between syntax and semantics. Although the core of topic models and related algorithms disregard syntax and its sequential properties, syntax has to be considered at least in a pre-processing step; function words have to be ruled out. Besides, words that characterize topics well are mostly nouns, and to a certain degree also adjectives and adverbs, that is, words that belong to open word classes. Thus, we conclude that topic modeling may profit from syntactic knowledge. Nevertheless, note that in topic modeling practice efficient heuristics can serve as a filter to separate function and content words with considerable success. For instance, the well-known *tf-idf* approach can be used to suppress function words. It exploits the different frequency aspects of function and content words and how they are distributed across documents. Therefore, it can efficiently be applied on the term-document matrices without taking sentence structure into account. For this reason, we do not evaluate the topic modeling performance of the LDAHMM in this work and focus on the POS tagging aspect.

Considering the opposite direction, we can ask if syntax learning depends on topic models, or if it can profit from topic models. The first point that addresses this question concerns the distinction between closed and open word classes, and function and topic words again; it is similar to the idea of the tf-idf approach to stop word removal, however, in this case we turn the argumentation into the opposite direction. As we have already mentioned, words that have syntactic function are assumed to be distributed rather equally over documents independently of their topic – given that the documents are suitably long. The reason for this is that the language and its grammar are assumed to be the same across topics. Readers should be able to decode the sentences of arbitrary documents without too much specialization on a certain topic. Therefore, function words and their usage must stay the same over all topics. Topic words, however, are assumed to be distributed non-uniformly. The distribution of topic words should follow the topic distributions within certain bounds. As a consequence, simple term-document matrices, which resemble the distribution of words across documents, can provide reasonable information to differentiate between topic and function words. Therefore, it can help to learn syntax models because topic words mostly belong to open word classes and function words to closed word classes. Furthermore, topic modeling can help to generalize syntax models. If we group words into topic and function words, we can use the cluster membership information to transfer syntactic classification knowledge among words of the clusters. The information about the degree of topic specificity of a word may help to smooth syntax models such as HMMs. Hence, syntactic models can also benefit from topic modeling knowledge.

Ultimately, we see that topic models and syntax models are closely intertwined. Even if the distinction of NLP approaches into topic models and syntax models makes sense, we cannot do one of them perfectly if we totally disregard the other. While we usually use crude heuristics like removing stop words to fit one problem to solve the other, it would be desirable to have a shared model of both. The development of a structure of such a combined model is a challenging task due to computational complexity problems.

In the upcoming section, we study the approach of Griffiths et al. (2004) who combine the generative graphical models of LDA topic modeling and HMM POS tagging into one system, the LDAHMM.

5.2 The Latent Dirichlet Allocation Hidden Markov Model

In Section 3 we have shown how topic models can be learned with LDA, and in Section 4 we have discussed HMMs for learning and representing syntax; Griffiths et al. (2004) proposed the Latent Dirichlet Allocation Hidden Markov Model (LDAHMM) for unsupervised acquisition of a language model that reflects both syntax and topic model aspects by a combination of LDA topic modeling and HMM syntax representation. That is, the HMM contributes a sequential representation of syntactic states with the well established dependencies between adjacent POS tags. Topic words, however, have a rather different nature; they do not have local dependencies, but rather long-range dependencies to words in the same document from the same topic. For this type of statistical distribution, Griffiths et al. (2004) employ an LDA style model. By contrast to a traditional LDA topic model, the LDAHMM uses a designated *topic state* in the syntax model to differentiate between function and topic words and to decide when the topic model should be activated. The distinction between words in the topic state and words in the other syntactic states allows us to switch between local dependencies and long-range dependencies. Only words that belong to the topic state are assumed to follow the LDA topic model part. They are modeled with regard to the topic of the document which concerns distant words. This is computationally not feasible and not necessary for syntactic words and syntactic dependencies. Words in their syntax states are handled like in the traditional HMM approach. That is, only with local dependencies.

Consider for example two documents of a corpus where the first document belongs to the IT domain (T_1) and the second document belongs to the sports domain (T_2), and we pick one sentence of each of the two documents.

(8) Function (F) and topic (T) words

a.

The	*keyboard*	*and*	*the*	*mouse*	...
F_1	T	F_3	F_1	T	...
-	T_1	-	-	T_1	...

b.

The	*soccer*	*stadium*	*is*	...
F_1	T	T	F_2	...
-	T_2	T_2	-	...

Example (8) presents these two sentences and the intended tagging of a LDAHMM model. The first layer represents the words whereas the second layer shows the tags that indicate their syntactic state. The latter can be one of several different syntactic states, that is, either a function state F_i, which corresponds to a POS class, or a designated state T

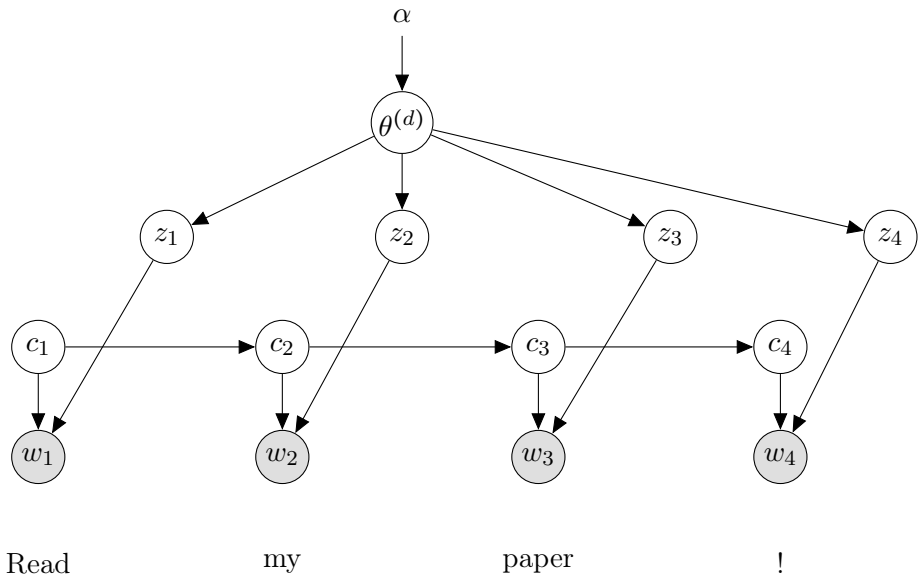


Figure 5.1: Part of a rolled out LDAHMM for POS tagging ($\beta, \gamma, \delta, \phi, \pi$ and corresponding plates and edges omitted for readability) (c.f. Griffiths et al., 2004).

for words that are topic words. When the syntax state of a word takes this special state T, the topic of the corresponding word is printed in the third layer. Note that we used unbiased identifiers F_i instead of actual POS class labels because we will train LDAHMMs in an unsupervised fashion.

Figure 5.1 demonstrates the generative graph structure of a LDAHMM rolled out on a short sentence. Compared with Figure 4.1 on page 33, we see the shared sequential syntax structure in both graphs; the Hidden Markov Model for Part-of-Speech tags represented by the variables c_i and their chain structure. However, we observe that the generation of words in the LDAHMM also depends on semantic states z_i which in turn depend on a topic vector $\theta^{(d)}$ that represents the semantic topic mixture of that document. This topic vector affects *all* words in that document that belong to the syntactic topic state and therefore incorporates long-range dependencies. The topic vectors are sampled from a Dirichlet distribution with the hyperparameter α.

Formally, the Latent Dirichlet Allocation Hidden Markov Model (LDAHMM) (Griffiths et al., 2004) is defined by a generative process. We start with document topic distributions $\theta^{(d)}$, sampled from Dirichlet$(\cdot|\alpha)$, word emission distributions $\phi^{(z)}$ and $\phi^{(c)}$ for the topics and syntactic classes respectively, and the transition probabilities π, drawn from Dirichlet$(\cdot|\gamma)$. LDAHMMs operate over a sequence of words $\mathbf{w} = (w_1, \ldots, w_n)$ where the subsequences of $\mathbf{w}$ correspond to the sentences of D documents. The vocabulary size V specifies which terms can occur, that is, $\forall w_i : 1 \leq w_i \leq V$. Besides, each word has a topic state in $\mathbf{z} = (z_1, \ldots, z_n)$ and a syntax state in $\mathbf{c} = (c_1, \ldots, c_n)$. We assume T different

topic states and C different syntactic classes so that $\forall i : 1 \leq z_i \leq T, 1 \leq c_i \leq C$. One state c_i, which we arbitrarily choose to be $c_i = 2$, has a different meaning than all other states. When $c_i = 2$, the word w_i is in the *topic state*. For each topic z there is a topic dependent word emission distribution $\phi^{(z)}$, drawn from Dirichlet$(\cdot|\beta)$. Moreover, there are $C - 1$ syntactic word emission distributions $\phi^{(c)}$, that is, for every syntactic state except the one that refers to the topic state. Each $\phi^{(c)}$ is drawn from Dirichlet$(\cdot|\delta)$. The D documents each have a topic vector $\theta^{(d)}$ as mentioned above. The transition probabilities between syntactic states c_{i-1} and c_i are determined by $\pi^{(c_{i-1})}(c_i)$. Symbols and terms used to describe the LDAHMM are recorded in Table 5.1.

Once the document topic distributions, word emission distributions, and syntactic state transition distributions have been drawn from their corresponding Dirichlets, the generative process of the LDAHMM proceeds for each document as follows. Sample the words w_i of document d according to:

1. $z_i \sim \text{Mult}(\cdot|\theta^{(d)})$
2. $c_i \sim \text{Mult}(\cdot|\pi^{c_{i-1}})$
3. $w_i \sim \begin{cases} \text{Mult}(\cdot|\phi^{(z_i)}) & \text{if } c_i = 2 \\ \text{Mult}(\cdot|\phi^{(c_i)}) & \text{else.} \end{cases}$

Table 5.1: Notation for the Latent Dirichlet Allocation Hidden Markov Model

Symbol	Domain	Description
α	$\mathbb{R}_+$	Hyperparameter for the document topic mixtures
β	$\mathbb{R}_+$	Hyperparameter for the topic word distributions
γ	$\mathbb{R}_+$	Hyperparameter for the syntactic state transition distributions
δ	$\mathbb{R}_+$	Hyperparameter for the syntactic state word distributions
$\theta^{(d)}$	$\mathbb{R}_+^T, \sum_i \theta_i^d = 1$	topic mixture of document d
$\phi^{(z)}$	$\mathbb{R}_+^V, \sum_{v=1}^V \phi^{(z)}(v) = 1$	word distribution of topic z
$\phi^{(c)}$	$\mathbb{R}_+^V, \sum_{v=1}^V \phi^{(c)}(v) = 1$	word distribution of syntactic state c
$\pi^{(c_i)}$	$\mathbb{R}_+$	transition distribution of syntactic state c_i
$z_i^{(d)}$	$\{1, 2, \ldots, T\}$	topic of the ith word of document d
$w_i^{(d)}$	$\{1, 2, \ldots, V\}$	word type of the ith word of document d
$c_i^{(d)}$	$\{1, 2, \ldots, C\}$	syntactic state of the ith word of document d
D	$\mathbb{N}$	corpus size
N_d	$\mathbb{N}$	size of document d
T	$\mathbb{N}$	number of topics
V	$\mathbb{N}$	vocabulary size
C	$\mathbb{N}$	the number of different POS classes

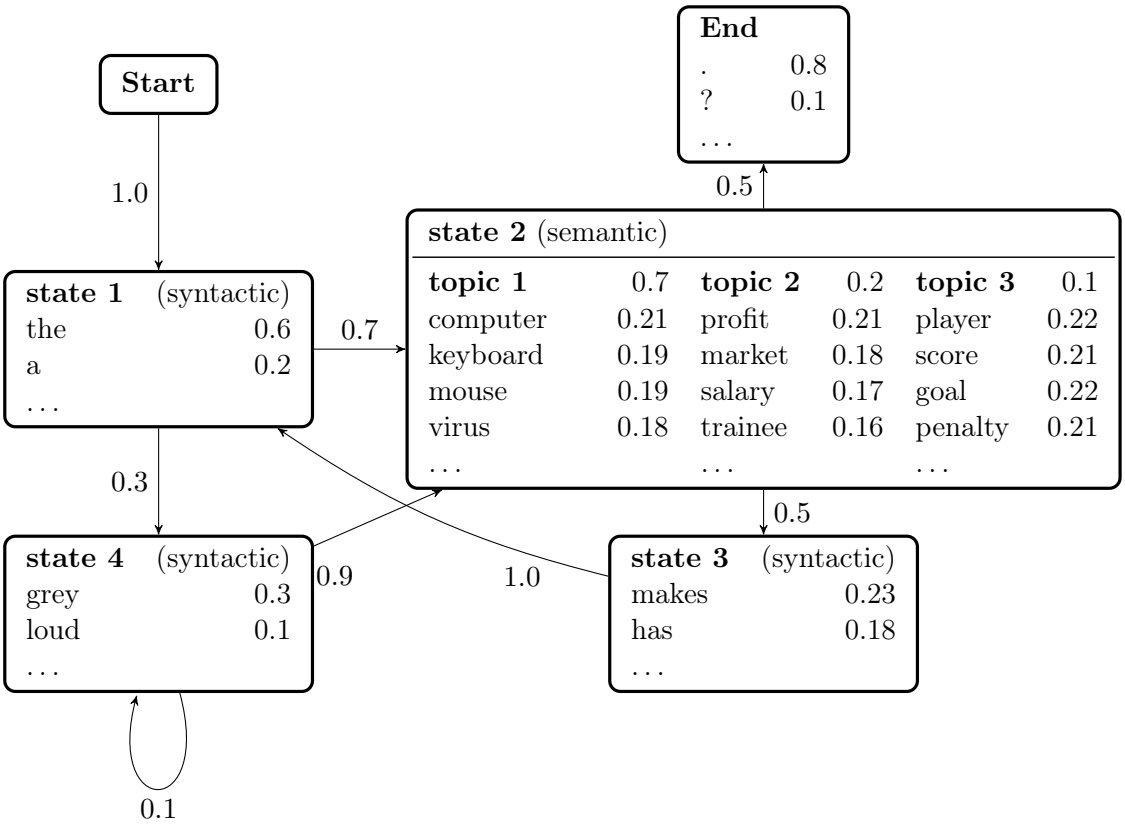


Figure 5.2: Weighted Finite State Machine of a simplified Latent Dirichlet Allocation Hidden Markov Model for POS classes and words on a specific document (c.f. Griffiths et al., 2004).

In Section 4.2, we have learned that HMMs can equivalently be regarded as Weighted Finite State Machines. If the topic vectors for the documents are given, we can also depict a LDAHMM as a WFSM. Figure 5.2 shows such a system for a specific document. This automaton has five different states plus the start state. States one to three are syntactic states and function analogous to the states of the WFSM in Section 4.2. The second state, however, denotes the semantic state, which generates the content words of the document. When the automaton is in this state, it follows a two-step procedure to choose the word that should be emitted. First, it determines the topic of the word to emit. This selection is influenced by the topic vector of that document. Subsequently, it draws the word to emit from the specific word emission distribution of that topic. Note that this modified sampling procedure enables the LDAHMM to adapt to documents and topics. In Section 4.2, the WFSM generated the sentence in Example (7a) which did not make much sense. One reason for this poor composition was the assumption of the HMM that sampling from the noun class follows the same distribution regardless of the topic of the document. The words *keyboard* and *penalty* in Example (7a), however,

belong to different topics and are not very likely to occur in the same document. The WFSM with the LDAHMM model shown in Figure 5.2 addresses this problem. Here we have an emission probability that is specialized for that document. It follows a topic mixture that avoids the composition of sentences with words from topics that do not go together. Since the probability of the first topic is 0.7 and therefore much higher than the probability of the third topic, which is 0.1, this automaton avoids mixing up terms of topics such as "keyboard" and "penalty". Nevertheless, the LDAHMM model is a simplified language model, and still creates sentences that sound strange to humans as can be seen in the following examples.

(9) Sentences generated from the WFSM shown in Figure 5.2

a.

A	*computer*	*has*	*a*	*mouse*	.
state-1	state-2	state-3	state-1	state-2	end
-	topic 1	-	-	topic 1	end

b.

**A*	*grey*	*grey*	*virus*	.
state-1	state-3	state-3	state-2	end
-	-	-	topic 1	-

In Example (9), the sentence that is not very likely in real world conversations is marked by an asterisk again.

Inference

Griffiths et al. (2004) use a fully Bayesian approach, similar to the fully Bayesian approach to POS tagging that we have already encountered in Section 4.2. That is, we integrate over the parameters of the model and sample from the posterior with Gibbs sampling.

According to Griffiths et al. (2004), the inference equations are given by

$$\begin{aligned} P(z_i|\mathbf{z_{-i}}, \mathbf{c}, \mathbf{w}) &\propto P(z_i|\mathbf{z_{-i}})\, P(w_i|\mathbf{z}, \mathbf{c}, \mathbf{w_{-i}}) \\ &\propto \begin{cases} n_{z_i}^{(d_i)} + \alpha & \text{if } c_i \neq 2 \\ \left(n_{z_i}^{(d_i)} + \alpha\right) \frac{n_{w_i}^{(z_i)} + \beta}{n_{\cdot}^{(z_i)} + W\beta} & \text{if } c_i = 2 \end{cases} \end{aligned} \tag{5.1}$$

for the topic states, and

$$P(c_i|\mathbf{c_{-i}}, \mathbf{z}, \mathbf{w_{-i}}) \propto P(w_i|\mathbf{c}, \mathbf{z}, \mathbf{w_{-i}}) P(c_i|\mathbf{c_{-i}}) \tag{5.2}$$

$$\propto \begin{cases} \frac{n_{w_i}^{(c_i)} + \delta}{n_{\cdot}^{(c_i)} + W\delta} \cdot \frac{(n_{c_i}^{(c_{i-1})} + \gamma)(n_{c_{i+1}}^{(c_i)} + \mathbf{1}_{c_{i-1}=c_i} \cdot \mathbf{1}_{c_{i+1}=c_i} + \gamma)}{n_{\cdot}^{(c_i)} + \mathbf{1}_{c_{i-1}=c_i} + C\gamma} & \text{if } c_i \neq 2 \\ \frac{n_{w_i}^{(c_i)} + \beta}{n_{\cdot}^{(c_i)} + W\beta} \cdot \frac{(n_{c_i}^{(c_{i-1})} + \gamma)(n_{c_{i+1}}^{(c_i)} + \mathbf{1}_{c_{i-1}=c_i} \cdot \mathbf{1}_{c_{i+1}=c_i} + \gamma)}{n_{\cdot}^{(c_i)} + \mathbf{1}_{c_{i-1}=c_i} + C\gamma} & \text{if } c_i = 2 \end{cases} \tag{5.3}$$

for the syntactic states. The counts $n_x^{(y)}$ are defined analogously to Section 3.1, $\mathbf{1}_{x=y}$ denotes the indicator function.

6 Adaptive Generative Features for Parts-of-Speech

In this chapter, we return to the problem of domain adaptation with a special focus on POS tagging. First, it is convenient to introduce measures of domain divergence so that we can show that the distributions in our setting differ and require domain adaptation. In the following section, we formalize the problem statement of the domain adaptation task. We follow the work of Ben-David et al. (2006) and Huang and Yates (2010) and assume that no labeled training examples of the target domain are given. Finally, we explain a certain approach to domain adaptation which Jiang (2008) calls *change of representation*, and we illustrate how HMMs and LDAHMMs can be embedded into this framework as a representation learning component. At the end of this chapter, we provide an overview of the related field of *transfer learning* approaches.

6.1 Domain Divergence

Throughout this work, we have always regarded words and their hidden states with probabilistic models and distributions. As we will formalize in the upcoming section, a domain is nothing but a distribution over the set of possible input instances. In many other scientific disciplines there have been contributions to measure the divergence between probability distributions, where divergence is a special concept of difference between two distributions. Naturally, we can use already existing divergence measures to quantify the differences between domains. Especially, we will make use of concepts from information theory.

A popular measure of divergence is the *Kullback-Leibler divergence* or relative entropy (c.f., e.g., MacKay, 2005). For two discrete probability distributions P, Q with the same alphabet, the Kullback-Leibler divergence is defined by

$$D_{\mathrm{KL}}(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}. \tag{6.1}$$

This definition is only valid if $\forall x : P(x) > 0 \Rightarrow Q(x) > 0$ holds. If $0 \cdot \ln 0$ occurs in the sum, it is treated as zero. In particular, the Kullback-Leibler divergence is always greater or equals zero, and the latter is the case if and only if P and Q are the same, and thus,

$$D_{\mathrm{KL}}(P||Q) > 0 \quad \text{if } P \neq Q, \qquad\qquad D_{\mathrm{KL}}(P||P) = 0. \tag{6.2}$$

On the one hand, this aspect of the Kullback-Leibler divergence meets our expectations on a measure of difference between domains since we desire positive values for different domains and zero for equality. On the other hand, we encounter situations where $Q(x) = 0 \wedge P(x) > 0$ for domains P, Q, and we also desire symmetry, which is not supported by the Kullback-Leibler divergence; formally, D_{KL} is not a distance measure. A symmetric and well-known extension of the Kullback-Leibler divergence is the *Jensen-Shannon divergence* (c.f., e.g., Lee, 1999), which is given by

$$D_{\mathrm{JS}}(P, Q) = \frac{1}{2} \left(D_{\mathrm{KL}}\left(P \| M\right) + D_{\mathrm{KL}}\left(Q \| M\right) \right), \tag{6.3}$$

where $M = \frac{P+Q}{2}$. Obviously,

$$D_{\mathrm{JS}}(P, Q) \geq 0 \tag{6.4}$$

for any P, Q since

$$D_{\mathrm{KL}}(P \| M) = D_{\mathrm{KL}}(P \| \overbrace{\frac{P+Q}{2}}^{=P}) \overset{(6.2)}{=} 0 \qquad \text{if } P = Q,$$

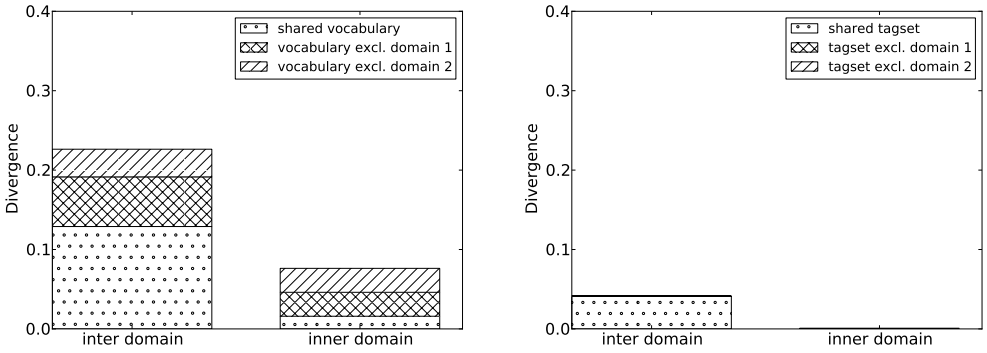
$$D_{\mathrm{KL}}(P \| M) = D_{\mathrm{KL}}(P \| \underbrace{\frac{P+Q}{2}}_{\neq P}) \overset{(6.2)}{>} 0 \qquad \text{if } P \neq Q,$$

and analogous equations hold for $D_{\mathrm{KL}}(Q \| M)$. In contrast to the Kullback-Leibler divergence, the Jensen-Shannon divergence is also defined if $P(x) > 0$ and $Q(x) = 0$ for some x. This is a nice property which allows us to apply the Jensen-Shannon divergence to measure the difference between word occurrence distributions.

In order to gain insight into domain adaptation and to show that it is necessary in our experimental setting, we will take a look on the data of our case study. We construct two different domains from the Brown Corpus[1]. The first domain contains documents from the categories *learned*, *editorial*, *news*, and *reviews* (Informative Prose). We compare these documents to documents from *fiction*, *science fiction*, *mystery*, *romance*, *adventure*, and *humor* (Imaginative Prose). At first, we consider simple word occurrence statistics. We compute inter domain divergence, and inner domain divergence as a reference value. In the latter case, we split up the words of the first domain into two pieces. Figure 6.1a depicts how domain membership affects divergence in word distributions. First of all, we can see that the divergence between the domains is substantially greater than the inner domain divergence. We can further notice that a salient part of the divergence stems from vocabulary that is domain specific. This contingent also includes OOV terms. Besides, the divergence in the use of shared vocabulary terms between domains is more than the whole inner domain divergence. Taken together, there is an evident variation in the usage of words between our domains.

In contrast to domain divergence in the distributions of observed states like words, it is also possible that the distribution of the hidden states changes. For example, romance

[1] Further explanation of the Brown corpus follows in Section 7.



(a) Words (b) POS tags

Figure 6.1: Jensen-Shannon divergence of word distributions for interdomain (domain one vs. domain two) and innerdomain (one half of domain one vs. the other half) where domain one contains categories "learned", "editorial", "news", and "reviews" (Informative Prose) and domain two contains categories "fiction", "science fiction", "mystery", "romance", "adventure", and "humor" (Imaginative Prose) from the Brown corpus text.

literature can be expected to contain more adverbs than scientific writing. Figure 6.1b depicts the divergence in POS tags rather than words for the same interdomain and innerdomain data as regarded for the divergence in word distributions. We can clearly see that there is a difference in the use of POS tags in between domains. However, the differences in tag distributions are almost negligible compared to the differences in the word distributions. This conforms to the claim of syntax representations to reflect the general properties and structures of language, independent of any domain.

6.2 Domain Adaptation and Change of Representation

The domain adaptation task is an extension of the common supervised classification task. Classification can briefly be summarized as follows: we seek to find a function $\hat{f}$ that maps input instances $x \in \mathcal{X}$ to discrete outputs $y \in \mathcal{Y}$ and approximates a hidden target function f. Estimation is based on given training examples $\{(x_i, y_i)\}_{i=1...N}$ in the supervised case. The function $\hat{f}$ is applied and evaluated on previously unknown test instances $\{x_i\}$. Crucially, traditional classification approaches assume that training and test data follow the same distribution.

Domain adaptation, however, relaxes this assumption. It distinguishes between the *source* domain $\mathcal{D}_S$, from which the training data originated, and the *target* domain $\mathcal{D}_T$,

which belongs to the test data. In this context, we define a *domain*[2] $\mathcal{D}$ as a distribution over some space $\mathcal{X}$ of possible input instances, following the works of Ben-David et al. (2006) and Pan and Yang (2010). According to Pan and Yang (2010), we define a *task* $\mathcal{T} = (\mathcal{Y}, f)$ for a certain domain $D = (\mathcal{X}, P(X))$ as a pair of a label space $\mathcal{Y}$ and an objective predictive function $f(\cdot)$. *Learning* aims to approximate the hidden target function f of the task so that we can predict the labels of new examples x by $\hat{y} = \hat{f}(x)$. If we deal with probabilities, we use $P(y|x)$ instead of $f(x)$.

Domain Adaptation Formally, let $\mathcal{D}_S, \mathcal{D}_T$ and $\mathcal{T}_S, \mathcal{T}_T$ denote the source domain and the target domain, and the source task and the target task, respectively. Especially, we assume that $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S = \mathcal{T}_T$. Given some labeled source domain data $D_S = \{(x_{S_i}, y_{S_i})\}_{i=1...N}$ and some unlabeled target domain data $D_T^U = \{x_{T_i}\}_{i=1...M}$, *domain adaptation* seeks to solve the target domain task $\mathcal{T}_T$.

That is, we have given only unlabeled input instances from the target domain, and some input-output-pairs that, however, represent the source domain and the source task. With this information, we try to best approximate the objective function of the target task. On the one hand, the labeled data alone can only be used to learn a function that suffices to solve the source task. Applying such a model for prediction on the target task would heavily rely on the similarity between the domains and the tasks. On the other hand, the unlabeled data from the target task alone provides no information on how input and output spaces relate to each other and therefore depends on additional knowledge sources. With a proper combination of both data sources, however, we can try to use the labeled data of the source domain to learn the mapping between inputs and outputs, and simultaneously exploit the unlabeled data from the target domain to *adapt* the learned model to the target task. For example, Blitzer et al. (2006) use the POS-tagged documents of the Wall Street Journal part of the Penn Treebank (Marcus et al., 1993) (Financial domain) and plain-text PubMed[3] abstracts (biomedical domain) to build a POS tagger for biomedical documents. Put in different words, the goal of domain adaptation is to bridge the gap that arises through the divergence between the source and the target domain.

Note that we have already mentioned the names for slightly different settings than the domain adaptation environment. For example, domain adaptation assumes that $\mathcal{D}_S \neq \mathcal{D}_T$. If we relax this assumption, that is, if $\mathcal{D}_S \approx \mathcal{D}_T$ we arrive at the *semi-supervised learning* setting that we have touched in Section 2.6. Thus, the difference between semi-supervised learning and domain adaptation lies mainly in the distribution of the unlabeled training data that is used to increase the performance of the classification system. Semi-supervised learning has the simpler setting where the unlabeled target data stems from the same distribution as the labeled source data. Domain adaptation, however, has to compensate a shift in the input distribution. Nonetheless, due to the similarity of semi-supervision and domain adaptation, semi-supervised approaches can

[2] Do not confuse the usage of the term "domain" in this context with the domain of a function which means something completely different.

[3] `http://www.ncbi.nlm.nih.gov/pubmed`

serve as a starting point for domain adaptation (c.f., e.g., Blitzer et al., 2006). Vice versa, there are domain adaptation procedures that should also help in certain semi-supervision settings. We could principally imagine domain adaptation techniques that could be simplified or restricted to more efficient semi-supervised methods. Note, however, that domain adaptation and semi-supervision fundamentally differ in their focuses. The former tries to compensate a shift in the input distributions using unlabeled data from the target domain, whereas semi-supervision exploits unlabeled data of the same domain. The latter especially entails completely different assumptions and enables differing strategies. Although not explicitly stated, some algorithms of these domains are rather a combination of semi-supervision *and* domain adaptation, where both issues are addressed. Relaxing another assumption, that is, if only $\mathcal{D}_S$ is given and we simply need to solve the source task, we come back to the *traditional classification* task. Besides, if we are restricted to learn from unlabeled target data and we seek to solve the target task, domain adaptation reduces to *unsupervised learning*, which we have alluded to in Section 2.6. We provide a further embedding of domain adaptation into the transfer learning landscape at the end of this chapter.

Change of Representation

In this work, we will use a *change of representation* approach to domain adaptation. The fundamental idea of this approach is to generate a different view on the input data in both the source and the target domain that is similar across domains (cf. Ben-David et al., 2006). We thereby reduce the divergence between source and target domain in the input representation. As a consequence, the classifier is less irritated by instances from sparse regions of the input space. In the context of discrete inputs, this may be instances that have never or just rarely been seen in the training data. However, it is clear that equalizing the input instances to a similar distribution is not beneficial per se. At the same time, we have to assure that the new data structure is still useful for the classification task. Assume, for instance, the extreme case when we represent all instances by just one symbol. This would lead to equal distributions in both domains. Hence, the divergence between source and target domain would be zero. This representation, however, would not provide any margin between the classes and therefore would not allow to separate them. Classification would not be possible although we would have equally distributed data in both the source and the target domain. Consequently, representation learning must keep, or increase classification margin.

To recap, we desire a representation that

- has little divergence between source and target domain,
- provides powerful features for the classification task, and
- can be induced without labeled data in the target domain.

6.3 Adaptive Generative Features for Part-of-Speech

In the last section, we have introduced a general framework for domain adaptation. Now, we want to apply this approach to POS tagging and seek a special representation of words and their contexts that is convenient to classify them into their Parts-of-Speech. Therefore, we make a simple but intuitive and effective move. We combine unsupervised and supervised classification approaches: an *unsupervised* learner that has been trained on both the source *and* the target domain, and a *supervised* learner that has seen labeled examples *only from the target domain*. The categories provided by the unsupervised learner are handled over to the supervised learner as an additional feature for classification; this approach has previously been termed *representation learning* (Huang and Yates, 2010). Note that this setting is a legal configuration. In general, one has to be careful that the project setting does not make use of information from the test data that would not be provided in practice. Thus, it is essential that we only make use of the *unlabeled* target data. That means that we can always repeat our procedure for arbitrary new unlabeled data in a practical application environment. A procedure that would transfer information from the labeled test data into the inference process would be cheating. Our assumption that a batch of unlabeled test examples is given reflects the setting in many practical applications. Typically, acquiring unlabeled test data from the target domain is easy. For example, if we want to learn a POS tagger for medical texts, we naturally have unlabeled medical documents at hand.

As we have seen in Section 4.2 and Section 5.2, POS categories can be learned in an unsupervised fashion. Such tags that are induced without labeled data do not precisely fit the desired POS classes, nevertheless, they are a good first approximation of the target classes and can be used to get a first impression that stems from both the target *and* the source domain. Therefore, this representation is stable across domains. It has little divergence between the source and the target domain and thereby satisfies the first requirement for a proper data representation. At the same time, it resembles the POS classes. It projects words to a lower dimensional representation that is not any representation; essentially, it is a representation that reflects the POS distribution to a certain degree. We cluster exactly in the same direction that we seek to classify afterwards. Hence, it increases the margin of the classes in the training data and reduces the classification error such that the second requirement for an appropriate representation is also met. Thus, we assume that a generatively induced POS tagging model should be a good representation for a subsequent discriminatively trained POS classifier. The new representation is assumed to better generalize the classifier to the target domain and increase the tagging performance, especially on OOV words.

In Table 6.1, we can see the difference in the representation of the data for the CRF. The first column contains the POS classes of the words in the sentence. This information is hidden during test time and should be inferred by the CRF. The following 5 columns show lexical features of the words. Notably, the last two columns are the hidden states of a LDAHMM which was trained on all data. These features assign words to clusters

Table 6.1: Generative features for the POS tagging CRF

						LDA-HMM	
POS	word	suffix2	cap	suffix3	len	c-state	z-state
DET	"the"	"he"	"title"	"The"	"3"	"7"	-
N	"terms"	"ms"	"lower"	"rms"	"5"	"2"	"4"
V	"are"	"re"	"lower"	"are"	"3"	"11"	-
ADV	"generally"	"ly"	"lower"	"lly"	"9"	"2"	"4"
VN	"taken"	"en"	"lower"	"ken"	"5"	"2"	"4"
P	"for"	"or"	"lower"	"for"	"3"	"4"	-
VN	"granted"	"ed"	"lower"	"ted"	"7"	"2"	"4"
CNJ	"as"	"as"	"lower"	"as"	"2"	"12"	-
CNJ	"though"	"gh"	"lower"	"ugh"	"6"	"12"	-
PRO	"they"	"ey"	"lower"	"hey"	"4"	"3"	-
VD	"referred"	"ed"	"lower"	"red"	"8"	"11"	-
P	"to"	"to"	"lower"	"to"	"2"	"8"	-
ADJ	"direct"	"ct"	"lower"	"ect"	"6"	"2"	"4"
CNJ	"and"	"nd"	"lower"	"and"	"3"	"12"	-
ADJ	"axiomatic"	"ic"	"lower"	"tic"	"9"	"2"	"2"
N	"elements"	"ts"	"lower"	"nts"	"8"	"2"	"4"
P	"in"	"in"	"lower"	"in"	"2"	"4"	-
DET	"the"	"he"	"lower"	"the"	"3"	"5"	-
ADJ	"common"	"on"	"lower"	"mon"	"6"	"2"	"4"
N	"experience"	"ce"	"lower"	"nce"	"10"	"10"	-
P	"of"	"of"	"lower"	"of"	"2"	"4"	-
DET	"all"	"ll"	"lower"	"all"	"3"	"2"	"5"
.	"."	"."	"other"	"."	"1"	"1"	-

in the syntactic space (c-state) and in the topic space (z-state), respectively. We can see that the particles *for*, *in*, and *off* have been assigned to the same class (c-state=4) by the LDAHMM which makes classification almost trivial for that terms. However, there are also states that encode diverse POS classes, for example, the syntactic topic state (c-state=2) which contains adverbs, verbs, and nouns.

6.4 Transfer Learning

In this section, we embed domain adaptation into the bigger landscape of machine learning tasks that aim to transfer knowledge, which are subsumed by the term *transfer learning*. Throughout this section, we will mostly follow the terminology of Pan and

Yang (2010). We will reuse the notation and terms described above, especially the definitions of domain and task. In fact, one direction for characterization of transfer learning approaches is based exactly on these two terms. As we can see in Table 6.2, traditional machine learning assumes that both the source and the target domain are the same. In contrast, all transfer learning approaches have to tackle differences in either the domains or the tasks. In *inductive transfer learning*, the domains are the same but the tasks differ. The *transductive transfer learning* setting is the opposite: different but related domains and the same tasks in the source and the target. According to Pan and Yang (2010), *unsupervised transfer learning* has neither the same domains nor the same tasks. The problem addressed in this thesis belongs to transductive transfer learning regarding this categorization since our domains differ but the task remains the same.

Table 6.2: Transfer learning tasks and traditional machine learning (confer Pan and Yang, 2010)

	Source vs. Target	
Setting	Domains	Tasks
Traditional Machine Learning	the same	the same
Inductive Transfer Learning	the same	different but related
Transductive Transfer Learning	different but related	the same
Unsupervised Transfer Learning	different but related	different but related

Table 6.3: Categorization of Transfer Learning tasks (confer Pan and Yang, 2010)

	Presence of Labeled Examples		
Setting	Source Domain	Target Domain	Related
Inductive	x	x	Multi-task Learning
		x	Self-taught Learning
Transductive	x		Domain Adaptation, Sample Selection Bias, Co-variate Shift
Unsupervised			Clustering, Dimensionality Reduction

The next characterization of transfer learning approaches concerns the presence of labeled examples in the source or in the target domain as depicted in Table 6.3. Similar to traditional unsupervised learning, unsupervised transfer learning addresses the general case when neither in the source nor in the target domain labeled data is provided.

In the opposite direction, inductive transfer learning is related to traditional supervised and semi-supervised machine learning. It requires labeled data at least from the target domain although this data set is assumed to be too small to acquire the target prediction function directly; the additional knowledge must either be gathered from unlabeled or labeled data from a source domain. If it is possible to retrieve labeled data from a similar source domain, the setting is similar to *Multi-Task Learning*. If only unlabeled data from a close-by source domain can be obtained, the problem resembles *Self-taught Learning*. As noted above, domain adaptation belongs to the class of transductive transfer learning approaches. This category assumes that labeled data can be accessed in the source domain, however, the target domain has a lack of labeled examples. Nonetheless, transductive transfer learning can exploit unlabeled target domain data to gain knowledge about the target task. Figure 6.2 sums up this classification of transfer learning approaches by a diagram.

A third dimension for the categorization of transfer learning approaches concerns the subject of transfer. Some transfer learning approaches transfer knowledge of instances, other approaches focus on the feature representation, parameters, or relational knowledge. *Instance transfer* approaches take into account that the instances of the source domain do not equally help to learn the target task. They select and weight instances according to their relatedness and usefulness for the target domain and the target task. As discussed above, approaches that focus on the *feature representation* mostly try to encode the data by a shared representation that is beneficial for the target task. In some cases, we can transfer knowledge about *model parameters*, for example, when source and target task share the same prior distributions of some model aspect. Transfer learning

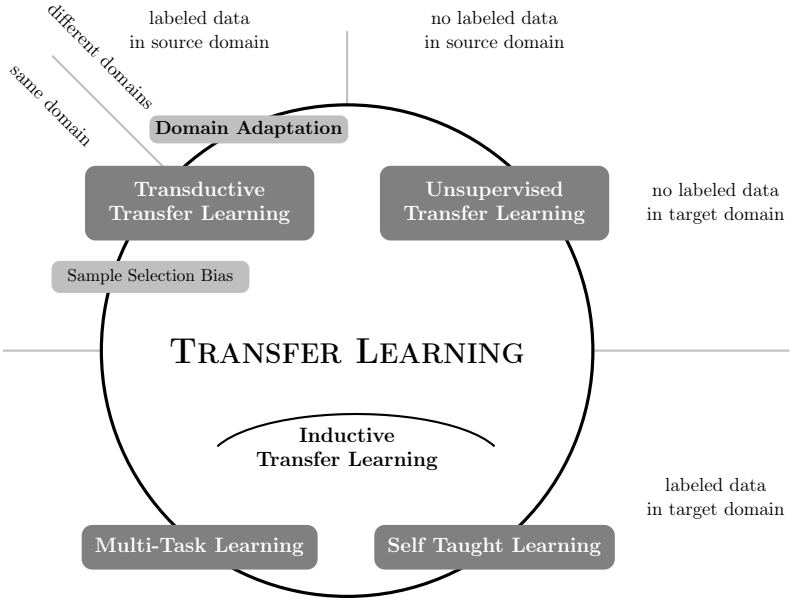


Figure 6.2: Landscape of Transfer Learning settings

has also been applied for relational data. *Relational knowledge* can be transferred when data of the source and the target domains have similar relationships. Table 6.4 shows which type of approach has been made to which type of transfer learning setting.

Table 6.4: Possible approaches for different transfer learning types (confer Pan and Yang, 2010)

	Focus of Transfer			
Setting	Instances	Feature-representation	Parameters	Relational-knowledge
Inductive	x	x	x	x
Transductive	x	x		
Unsupervised		x		

After this short overview, we take a closer look at transductive transfer learning and the special aspect of negative transfer due to their relevance and relatedness to domain adaptation. We will skip unsupervised transfer learning, self-taught learning, and multi-task learning, and we will also leave out details about relational knowledge transfer and parameter transfer approaches.

Transductive Transfer Learning

Imagine that we want to build a classifier for images of cats, especially including Siamese cats, and we already have some annotated images of different kinds of cats, for example, Siberian cats or English shorthair cats. We now assume that the images that we want to label (target domain) contain Siamese cats, while the images of our training data (source domain) contain different types of other cats but no Siamese cats at all. Thus, we seek to adapt our model of the cat images in the source domain to the new domain of Siamese cats.

Transductive Transfer Learning Similar to Pan and Yang (2010), let $\mathcal{D}_S, \mathcal{D}_T, \mathcal{T}_S, \mathcal{T}_T$ be the source domain, the target domain, the source task and the target task. If $\mathcal{D}_S \neq \mathcal{D}_T$ but $\mathcal{T}_S = \mathcal{T}_T$, and we aim to learn the objective function of the target domain exploiting knowledge from $\mathcal{D}_S, \mathcal{T}_T$, and $\mathcal{D}_T$, we call this setting *Transductive Transfer Learning*. Labeled source domain data is available, and also unlabeled target domain data.

This notion of transfer learning is equivalent to that of domain adaptation; considerably related but slightly different are sample selection bias and covariate shift. Sample selection bias (cf., e.g., Heckman, 1977) states equivalence between source and target domain and describes errors caused by training data that is not representative for that

domain. Put in different words, it assumes that the source and the target domain are the same but our training data samples are poor. In contrast, domain adaptation assumes that the source domain and the target domain are inherently different. Even more sampling data would not reduce the divergence between the domains. For example, texts from the science fiction domain are in fact different from scientific texts. They have inherently different word distributions. Covariate shift is sometimes used synonymously with domain adaptation (c.f., e.g.., Bickel et al., 2009).

Negative Transfer

Naturally the question arises if knowledge transfer can also hurt the performance of a classifier. Indeed, the *negative transfer* effect has been shown in experiments by Rosenstein et al. (2005). This may, for example, occur when the assumption of the similarity between the tasks does not hold. In that case, it would be better to learn an unbiased classifier by scratch, possibly in an unsupervised fashion on the target domain alone. For example, if an English native speaker tries to learn Chinese, any grammar transfer may disturb the initial learning process; it may be easier for that person to learn the new language when she starts unbiased. It is clear that there is a grey area between positive and negative transfer. For example, positive transfer may be possible for some instances or features, while negative transfer occurs for others. In this work, we have disregarded negative transfer effects because we have used background knowledge to choose a representation learning approach that clusters in the same direction as the classifier. This avoids some aspects of negative transfer for this domain. Nevertheless, recognizing and avoiding negative transfer can be import in several environments; this issue is a direction for future research.

7 Experimental Evaluation

In this chapter, we investigate the usefulness of automatically induced syntactic states and topic information for domain adaptation. Therefore, we address several questions:

1. quantitative questions:
   a) Do LDAHMMs or HMMs generate the better features for POS tagging?
   b) How much does the improvement depend on the amount of labeled training data?
2. qualitative questions:
   a) Are the LDA-states interpretable?
   b) Are the syntactic states interpretable?

We first describe metrics and then the general settings of the experiments: the data, the parameters and hyperparameters, the features, preprocessing and the workflow. Then we describe the outcomes, discuss and evaluate the results. Finally, we compare this thesis to related work.

7.1 Metrics

As described in Section 4, we measure the *word accuracy* of a tagger ζ by

$$acc(\zeta) = \frac{n_{\hat{c}=c}(\zeta)}{n} \tag{7.1}$$

where n is the total number of words in the test set and $n_{\hat{c}=c}(\zeta)$ is the number of words where the predicted tag $\hat{c}$ of the tagger matches the target tag c from the gold standard test set. We also evaluate the *sentence accuracy* of the models, which compares the fraction of perfectly tagged sentences, that is, sentences without any erroneous prediction, to the total number of all sentences.

The relative error reduction between two classifiers ζ_1, ζ_2 is computed by

$$\Delta_{\text{rel}}(\zeta_1, \zeta_2) = \frac{acc(\zeta_1) - acc(\zeta_2)}{1 - acc(\zeta_2)}. \tag{7.2}$$

To test significance between different classifiers, we apply a *k-fold cross-validated t-test.*

7.2 Experiment A: HMM vs. LDAHMM

In the first experiment, we study the effect of the number of syntactic and topic states of the LDAHMM on the performance of the CRF; in particular when $T = 1$, configurations conform to Bayesian HMMs.

7.2.1 Setting

The evaluation of domain adaptation methods requires corpora that not only provide POS tags but also give domain information and assign sentences to their document origin. If no such information is provided, we could generate artificial domain and document information by clustering documents with similar distributions. However, in our experiments, we use real documents and domains from the Brown corpus (Francis and Kucera, 1979) which provides this kind of information. It was collected by W. N. Francis and H. Kucera at the Brown University in 1964 and contains 500 documents with a total of approximately one million tokens. The Brown corpus has a large tag set compared to other corpora like the Penn Treebank. We, however, use a reduced tag set which can be obtained, for example, through a special option in NLTK[1]. The documents of the Brown corpus belong to two different domains: *Informative Prose* and *Imaginative Prose*. From the first domain, we take the categories *learned*, *editorial*, *news*, *reviews*, and from the second domain we only regard the categories *fiction*, *science fiction*, *mystery*, *romance*, *adventure* and *humor*.

We hold the number of sentences used for training the CRFs fixed at 9000 sentences from the source domain. The LDAHMM, however, is trained on these sentences and all the data from the target domain. We select the source domain training sentences in a four-fold manner; for each category of the source domain we create a data set that only contains instances from the other three categories. We evaluate the performance of models with generative feature where $T \in \{1, 3, 5, 7, 10, 15, 20\}$ and $C \in \{10, 15, 20, 25, 30\}$, and compare the results to a linear-chain CRF with a traditional feature set – CRFs are one of the current state-of-the-art techniques for supervised Part-of-Speech tagging.

To train the LDAHMMs, we replace rare words, that is, words with less than 6 occurrences, by special symbols; we differentiate between upper case and lower case rare words. For all experiments, we choose the hyperparemeters to be $\alpha = 50.0, \beta = 0.01, \gamma = 0.1$, which are the default values of the Matlab Topic Modeling Toolbox 1.4[2]. We run 600 Gibbs sampling steps for burn-in before we take 50 samples of the chain with a lag of 10 samples.

For all configurations, the base features for the CRF classifiers always stay the same. In addition to the label transition features that form the linear chain structure, we use different kinds of lexical features such as the lowercase form of the word, suffixes, the

[1] `http://nltk.org/`

[2] `http://psiexp.ss.uci.edu/research/programs_data/toolbox.htm`

Table 7.1: Base state features of all CRFs and the special LDAHMM state features.

Feature		
Name	Additional Explanation	Example
word	lowercase of w_i	running
bigram suffix	$w_i[-2:]$	ng
trigram suffix	$w_i[-3:]$	ing
case	upper, lower, or title case	lower_case
length	length of w_i	7
syntactic state	c_i of LDAHMM	19
topic state	z_i of LDAHMM	3

length of the word, and its case. Following the experimental setup of Huang and Yates (2009), we leave out context features, such as preceding and succeeding words. Table 7.1 lists the full set of state features: the shared base state features and the state features that indicate the states of the LDAHMM. We compare two different settings (C vs. CZ) of using the LDAHMM state features: using only the syntactic states (c_i), and using the syntactic and the topic states (c_i, z_i), respectively. Providing the CRF with topic states may help to disambiguate tags when word senses differ. The feature set of the base CRF configuration contains neither of the LDAHMM state features.

We run the linear-chain CRF implementation of Okazaki (2007)[3] with LBFGS (Nocedal, 1980) optimisation and train the models until convergence with an L2 regularisation with $c = 1.0$.

7.2.2 Results

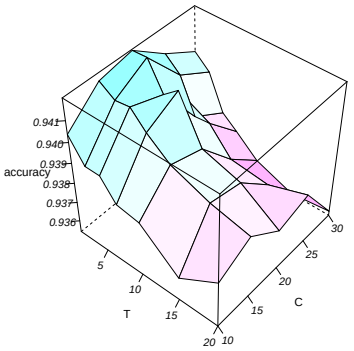
The accuracies of the different parameterizations are listed in Table 7.2; Figure 7.1a gives an overview of the data. The corresponding relative error reductions are visualized in Figure 7.1b.

Our first observation is that all tested combinations improve the baseline CRF. Even the worst model ($T = 15, C = 30$, C) still achieves 6.8% relative error reduction. The best models ($T = 1, C = 20$ and $T = 3, C = 20$, CZ) use syntactic and topic features, and on average they label 94.20% of the tokens correctly. Their average relative error reduction is 18.25%. Our second observation is that for each choice of T, C, we detect small but consistent improvements through the use of topic features and syntax features (CZ models) against only syntax features (C models).

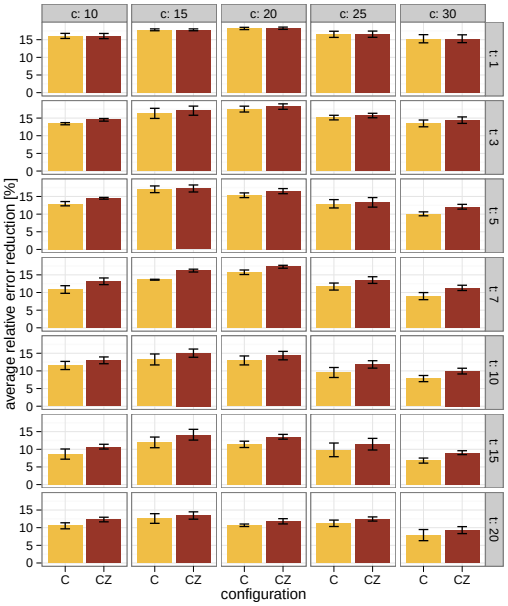
[3] `http://www.chokkan.org/software/crfsuite/`

Table 7.2: Accuracies of CRFs with generative features of LDAHMMs with different choices of T (number of topics) and C (number of syntax states), and configurations with topic features and syntax features (CZ models) against only syntax features (C models). Values are averages over four folds of overlapping training data on the same test set.

	C=10		C=15		C=20		C=25		C=30	
T	C	CZ	C	CZ	C	CZ	C	CZ	C	CZ
1	0.9404	0.9404	0.9416	0.9416	0.9419	0.9420	0.9408	0.9408	0.9399	0.9399
3	0.9385	0.9393	0.9406	0.9412	0.9415	0.9420	0.9397	0.9402	0.9386	0.9393
5	0.9382	0.9393	0.9411	0.9413	0.9399	0.9407	0.9383	0.9385	0.9362	0.9376
7	0.9366	0.9384	0.9387	0.9405	0.9402	0.9413	0.9373	0.9386	0.9354	0.9370
10	0.9371	0.9382	0.9384	0.9397	0.9382	0.9392	0.9358	0.9374	0.9346	0.9361
15	0.9351	0.9366	0.9375	0.9390	0.9371	0.9386	0.9361	0.9372	0.9338	0.9354
20	0.9365	0.9378	0.9380	0.9386	0.9366	0.9374	0.9370	0.9379	0.9346	0.9356



(a) Average accuracies for configurations with syntactic and topic state features (CZ)



(b) Relative error reductions for configurations with syntactic and topic state features (CZ) vs. only syntactic state features (C).

Figure 7.1: Results of Experiment A: different parameterizations of T (number of topics) and C (number of syntax states)

.

Moreover, we identify that for each choice of C, the configuration with only one topic state (HMM alike) performs better or at least equal to the corresponding ones with $T \geq 1$.

The accuracies for the HMM alike configurations are mostly the same whether they use only syntactic state features, or syntactic and topic state features. Only for $T = 1, C = 20$ there is a difference of 0.01% in accuracy that may have been caused by the stochastic nature of the algorithm.

7.3 Experiment B: Learning Curve

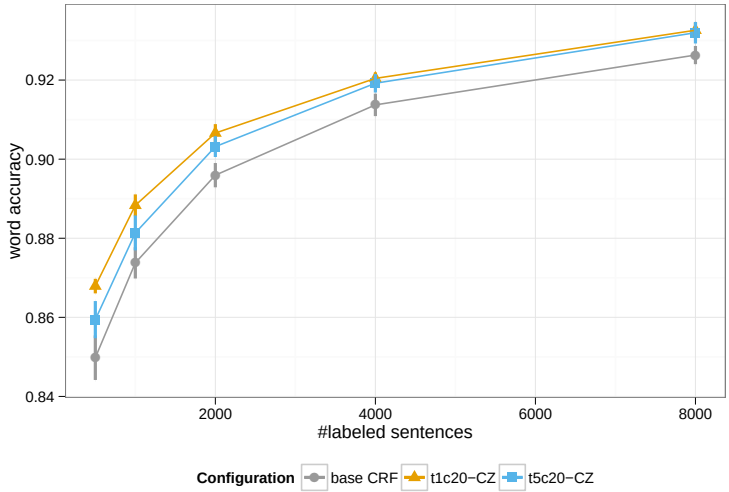
In the second experiment, we investigate how the labeled training set size influences the error reduction obtained from HMM or LDAHMM features respectively.

7.3.1 Setting

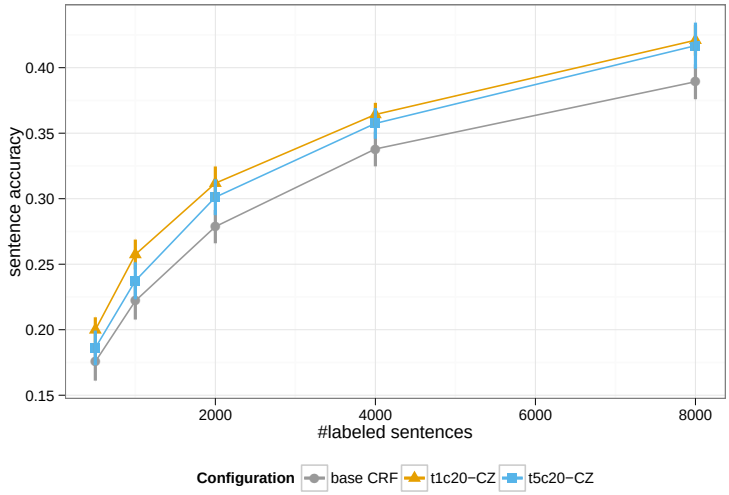
The setting of this experiment is similar to that of the first one. We have the same categories of the Brown corpus as source and target domains. The CRF parameters stay the same and also the LDAHMM configuration and data preprocessing. We create data sets with 500, 1000, 2000, 4000, and 8000 sentences of the target domain. For each training set size we build four folds. Similar to k-fold cross validation settings, our source domain sentences, that is, the training data of the folds, may overlap; they are chosen randomly from the source domain leaving out one of the four source domain categories for each fold. The test data of the folds was created by randomly splitting the target domain into 4 disjoint parts of equal size. In this trial, we only compare three different types of configurations: base CRF, $T = 1, C = 20$ (HMM) and $T = 5, C = 20$ (LDAHMM with topic and syntax features). These parameter values have been chosen as they represent a plausible background knowledge assumption. We assume approximately 20 syntactic states that are easy to learn in an unsupervised manner, and there should be at least 5 topics in the data because we initially know about the new target domain and four categories in the training data.

7.3.2 Results

The results of the second experiment are depicted in Figure 7.2, and listed in Table 7.3 and Table 7.4. Both settings with generative features (t1-c20, t5-c20) consistently surpass the baseline CRF where the results with the HMM-features are better than the results of the LDAHMM setting. The relative error reduction (concerning the word accuracy) of the HMM-setting is greater, when little labeled training data is available: 12.1% for 500 instances versus 8.5% for 8000 training instances. In contrast, the average relative error reduction of the LDAHMM fluctuates between 6% and 7%, roughly independently of the



(a) Word accuracy



(b) Sentence accuracy

Figure 7.2: Learning curves (averages over 4 folds) for three different configurations: base CRF, a CRF with HMM alike features where $T = 1, C = 20$ (t1c20-CZ), and a CRF with LDAHMM syntactic and topic features where $T = 5, C = 20$ (t5c20-CRF).

number of training instances. It achieves its highest average relative error reduction at 8000 labeled training instances with 7.7%; the corresponding accuracy is 93.20%, which is 0.06% under the accuracy of the best model. The best performance comes from the HMM-feature configuration (t1-c20-CRF) with 8000 labeled trainig instances).

Table 7.3: Learning curves (word accuracy): averages over four disjoint test sets; Δ_{rel} computed in comparison to the base CRF

# instances	base CRF	t1-c20-CRF (HMM)		t5-c20-CRF (LDA-HMM)	
(labeled)	w. acc.	w. acc.	Δ_{rel}[%]	w. acc.	Δ_{rel}[%]
500	0.8498	0.8679	(12.1)	0.8594	(6.4)
1000	0.8738	0.8883	(11.5)	0.8813	(6.0)
2000	0.8960	0.9066	(10.2)	0.9032	(6.9)
4000	0.9137	0.9204	(7.8)	0.9192	(6.3)
8000	0.9263	0.9326	(8.5)	0.9320	(7.7)

Table 7.4: Learning curves (sentence accuracy): averages over four disjoint test sets; Δ_{rel} computed in comparison to the base CRF

# instances	base CRF	t1-c20-CRF (HMM)		t5-c20-CRF (LDA-HMM)	
(labeled)	s. acc.	s. acc.	Δ_{rel}[%]	s. acc.	Δ_{rel}[%]
500	0.1756	0.1999	(2.9)	0.1861	(1.3)
1000	0.2220	0.2574	(4.6)	0.2374	(2.0)
2000	0.2787	0.3118	(4.6)	0.3012	(3.1)
4000	0.3379	0.3643	(4.0)	0.3575	(2.9)
8000	0.3893	0.4209	(5.2)	0.4168	(4.5)

When we consider the sentence accuracy instead of the word accuracy, we find similar results. As shown in Table 7.4 and Figure 7.2b, both configurations with generative features lie above the base CRF, and the Hidden Markov Model alike feature setting consistently exceeds the t5-c20-CRF. A difference to the word accuracy results regards the effect of more labeled training instances on the relative error reduction: for the sentence accuracy, it increases. The average sentence accuracies of the best models (t1-c20-CRF) range from 20.0% for 500 labeled training instances to 42.1% for 8000 labeled training instances. The corresponding absolute error reductions are 2.4% and 3.2% respectively.

Table 7.5: Significance Tests (paired one-sided t-test)

# instances	Null Hypothesis Setting			p-value
8000	(**T** = **1**, **C** = **20**)**-CRF**	vs.	base-CRF	0.0004
8000	(**T** = **5**, **C** = **20**)**-CRF**	vs.	base-CRF	0.0008
8000	$(T = 1, C = 20)$-CRF	vs.	$(T = 5, C = 20)$-CRF	0.2693
500	(**T** = **1**, **C** = **20**)**-CRF**	vs.	$(T = 5, C = 20)$-CRF	0.0318

Table 7.5 reports p-values of single-sided paired t-tests and prints models with significant improvements in boldface. We find that both the HMM alike configuration and the LDAHMM-CRF with $T = 5, C = 20$ perform significantly better than the base-CRF when all models are trained with 8000 labeled instances ($p < 0.001$). In this case, the difference between the former two models is not significant ($p > 0.1$). However, when less labeled training data is given, that is, only 500 instances, we detect that the HMM alike model is significantly better than the t5-c20-CRF setting.

7.4 Experiment C: Qualitative Analysis of States

In this section, we take a qualitative look at the words that form the syntactic states and topics of an LDAHMM model. As usual and already described in Section 3.2, we inspect these distributions by printing lists of the n most frequent terms in each syntactic state or topic respectively, along with the corresponding frequencies.

Table 7.6: The seven most frequent words and probabilities for the three topics of an unstable LDAHMM clustering (Experiment C). The heading also contains the marginal probabilities of the topics.

°	Topic 1	0.31210	Topic 2	0.34992	Topic 3	0.33798
1	CAP***CAP	0.11357	***	0.27693	all	0.06663
2	***	0.05380	not	0.06276	been	0.04148
3	also	0.01469	so	0.03801	too	0.03462
4	is	0.01409	more	0.02818	now	0.03036
5	more	0.01298	even	0.02513	just	0.02713
6	Mr.	0.01221	only	0.02498	as	0.02484
7	last	0.00965	no	0.02354	one	0.02413

7.4.1 Setting

We train one LDAHMM with the same hyperparameters, preprocessing, etc. as in Experiment A, take the first fold of its data, and evaluate the parameters of the LDAHMM after

Table 7.7: The seven most frequent words for the 20 syntax states of an unstable LDAHMM clustering (Experiment C). The headings also contain the marginal probabilities of the states.

	Syntax State 1	0.05734	Syntax State 2	0.11568	Syntax State 3	0.01707	Syntax State 4	0.04281
1	man	0.0177	the	0.4895	said	0.2163	way	0.0122
2	eyes	0.0121	a	0.1930	***	0.0717	man	0.0116
3	head	0.0109	his	0.0732	thought	0.0401	life	0.0107
4	room	0.0103	her	0.0273	knew	0.0360	week	0.0088
5	time	0.0102	an	0.0263	asked	0.0338	program	0.0078
6	door	0.0101	their	0.0193	told	0.0280	part	0.0078
7	day	0.0101	this	0.0191	know	0.0261	thing	0.0077
	Syntax State 5	0.06596	Syntax State 6	0.05614	Syntax State 7	0.02477	Syntax State 8	0.04698
1	,	0.9045	was	0.2220	CAP***CAP	0.0982	of	0.5705
2	–	0.0430	had	0.1305	time	0.0322	in	0.1261
3	:	0.0139	is	0.0909	one	0.0289	for	0.0751
4	(	0.0104	were	0.0587	people	0.0186	with	0.0439
5	)	0.0036	would	0.0505	course	0.0179	and	0.0238
6	than	0.0032	are	0.0384	years	0.0168	by	0.0220
7	States	0.0014	could	0.0345	fact	0.0110	from	0.0162
	Syntax State 9	0.05525	Syntax State 10	0.06265	Syntax State 11	0.02436	Syntax State 12	0.04491
1	and	0.4587	in	0.1709	to	0.7409	him	0.0897
2	that	0.1130	on	0.1089	not	0.0614	it	0.0829
3	but	0.0570	to	0.1031	will	0.0312	out	0.0581
4	as	0.0517	at	0.0807	would	0.0269	up	0.0440
5	when	0.0347	with	0.0764	never	0.0114	them	0.0438
6	or	0.0326	for	0.0587	might	0.0088	me	0.0381
7	which	0.0306	from	0.0478	or	0.0075	her	0.0373
	Syntax State 13	0.07603	Syntax State 14	0.03724	Syntax State 15	0.05020	Syntax State 16	0.02478
1	***	0.7203	be	0.1360	CAP***CAP	0.3909		0.5049
2	CAP***CAP	0.0169	have	0.0614	“	0.2523	?	0.1492
3	water	0.0034	been	0.0314	Mrs.	0.0177	;	0.0883
4	top	0.0021	do	0.0305	Mr.	0.0126	!	0.0574
5	light	0.0021	get	0.0301	–	0.0076	)	0.0163
6	paper	0.0017	see	0.0244	(	0.0068	again	0.0127
7	coffee	0.0015	go	0.0220	Miss	0.0062	now	0.0116
	Syntax State 17	0.04647	Syntax State 18	0.05031	Syntax State 19	0.05718	Syntax State 20	0.04388
1	he	0.2036	CAP***CAP	0.0338	The	0.1192	***	0.2182
2	I	0.1465	new	0.0177	He	0.0910	made	0.0208
3	it	0.0777	other	0.0175	CAP***CAP	0.0879	came	0.0184
4	she	0.0720	first	0.0173	It	0.0379	went	0.0160
5	they	0.0622	little	0.0171	But	0.0314	looked	0.0124
6	you	0.0578	old	0.0150	She	0.0306	took	0.0122
7	we	0.0328	own	0.0134	And	0.0216	got	0.0114

800 iterations with 3 topic states. We choose $n = 7$ and employ the integrated function of the above-mentioned Matlab Topic Modeling Toolbox for tabulating the seven most frequent terms along with their frequencies inside the topics or syntax states respectively. We also print the marginal probabilities for the topics and syntax states.

7.4.2 Results

The most frequent words and their probabilities are shown in Table 7.6 and Table 7.7 for the topic states and the syntax states respectively. While the syntax states mostly meet our expectations, and certain states can be matched with definite POS classes as

discussed in the following section, we failed to interpret and label the topics in Table 7.6. As mentioned before, the topic states of the LDAHMM were expected to comprise mostly nouns. However, the third topic state in Table 7.6, for example, looks completely different.

7.5 Discussion

At the very beginning, we see evidence that all the generative features of our settings considerably improve supervised POS tagging for domain adaptation. Even when the parameters of the LDAHMM are not chosen perfectly we still observe reliable gains in accuracy through the shared representation. The knowledge transfer between the source and the target domain seems to help classifying words into their syntactic categories. However, we must admit that our current evaluation does not compare the domain adaptation setting to the related task of semi-supervised learning. Some amount of the error reduction may also have been obtained by exploiting unlabeled data from the same domain. It will be left for future investigations to find evaluation settings that allow for comparison of the contributions of domain adaptation and semi-supervision.

When we compare the results of Experiment A and Experiment B, we find a much higher relative error reduction in the former. We expect that this is due to the smaller amount of unlabeled data in the settings of Experiment B; these configurations only use one fourth of the number of target domain sentences of Experiment A. A next step for future work would be to further investigate the effect of the amount of unlabeled training data on the error reduction.

Interestingly, we showed that for small corpora with up to 9000 sentences of English, features of pure syntax clustering with a Bayesian-HMM may be preferred for POS tagging. They performed equally or more effective than clustering with an LDAHMM. While the difference between the two approaches is significant for very small corpora (500 labeled instances), it is not significant for training a CRF with 8000 labeled instances; for the setting with 9000 labeled training instances in Experiment A, we observed equal performance of the HMM alike setting and the t3-c20-CRF. Hence, it may be that more labeled, or more unlabeled data is necessary to learn an appropriate LDAHMM representation. This may be caused due to the increased number of parameters to fit in LDAHMM models. Nevertheless, even if the POS tagging performance of an LDAHMM-feature representation is not as good as a HMM-feature representation, the LDAHMM may be preferred if the topic knowledge is desired, for example, in the case of Named Entity Recognition (NER) or word sense disambiguation. The qualitative results of our third experiment, however, suggest that this setting has to be carefully configured. The topic states in Experiment C do not seem to reflect topics in the sense that we expected, and that were reported by Griffiths et al. (2004) or Li and McCallum (2005); they find topics that are mostly made of nouns. Maybe this comes because our domains consist of imaginative and informative prose or because our corpora are too small. Note that

Experiment C reflects only the outcome of one single run, and therefore should not be overemphasized; the unexpected clustering could have been caused due to the stochastic nature of the algorithm. We also inspected some feature files on a random basis (c.f. Table 6.1) that agreed more with our expectations than the distributions shown in Table 7.6. Still, it shows that LDAHMM clusterings may be unstable for small corpora. By contrast, the learned syntax states of the LDAHMM seem to match certain POS classes and thus may be useful for classification. For example, syntax state 17 represents personal pronouns, syntax state nine contains coordinating conjunctions, and syntax state one is mostly made up of singular nouns.

Another interesting behaviour can be identified in the sentence accuracy learning curves: the relative error reductions increase with the number of labeled training instances, with the only exception at 4000 labeled training instances. We speculate that the differences in word accuracies cause this phenomenon; when the word accuracy is low, the same amount of word accuracy improvement should have less effect compared to a setting where the word accuracy is high. In the latter case, there are more sentences that are close to a perfect tagging. As a result, even small improvements can yield a gain in sentence accuracy. In the former case, a lot of error corrections do not contribute to the sentence accuracy.

Our general POS tagging performance is below the 97% of application-ready POS taggers which has three reasons. First, we only use a small fraction of training data compared to mature Part-of-Speech taggers. Second, we designed our experiments to reflect aspects of the domain adaptation task which is harder than typical POS tagger evaluation. Third, we did not spend any time into hand-crafted features. Instead, our results show that generative features can serve as a surrogate for some amount of hand-crafted features, and that they can increase the performance considerably.

7.6 Related Work

In this section, we discuss several prior work that has addressed domain adaptation, or areas that are related to our work, especially semi-supervised learning approaches. We also take a look at unsupervised POS tagging, and ways to integrate syntax and semantics.

Domain adaptation, in particular for Part-of-Speech tagging, has been addressed previously, for example, by *Structural Correspondence Learning* (SCL) (Blitzer et al., 2006) which extends the semi-supervised structure learning approach of Ando and Zhang (2005). SCL searches for correspondences between features from different domains and produces shared representations for discriminative classification. The feature correlation analysis differentiates between so-called *pivot features* which play the same role for classification across domains, and non-pivot features whose function is guessed by their relation to pivot features. Blitzer et al. (2006) cite significant improvements over traditional supervised and semi-supervised. Nonetheless, (Huang and Yates, 2009; Huang

and Yates, 2010) compared SCL to different kinds of *representation learning* settings, and their empirical investigations suggest that the latter is superior to SCL. Huang and Yates (2009) find that among several distributional representations, that is, TF, TF-IDF, LSA and HMMs, the generative features of Hidden Markov Models mostly provide the best information. Huang and Yates (2010) follow up this direction by proposing a combination of multiple independent HMMs (iHMMs) which performed better than SCL, features from Contrastive Estimation (Smith and Eisner, 2005), and their previous system. In our work, we use Bayesian HMMs instead of Expectation Maximization, which should be preferred referring to Goldwater and Griffiths (2007), Johnson (2007), and Toutanova and Johnson (2007). Besides, we compare HMM features to LDAHMM features for different configurations and parameterizations.

As noted in Section 6.4, domain adaptation has much in common with semi-supervised learning approaches. For example, Nigam et al. (1998) suggest an approach for text classification with labeled and unlabeled documents. They combine a Naive Bayes classifier with Expectation Maximization: an initial model from the labeled training data classifies all available instances before several iterations of learning and inference are performed until it reaches a stable state. Although the simplicity of this purely generative approach is appealing, we believe that a combination of generative and discriminative models has several advantages because it can focus on classification performance and jointly use distributional clusters of the unlabeled data. However, it would be interesting to use an approach similar to (Nigam et al., 1998) to direct the LDAHMM clustering into the desired dimensions in future work. A different semi-supervised learning approach comes from Li and McCallum (2005) who use a setting that is close to our work. They compare features generated by LDAHMMs and Mutual Information clustering for Part-of-Speech tagging and Chinese word segmentation together with linear-chain Conditional Random Fields. Li and McCallum (2005) arbitrarily set the numbers of topic and syntactic states and do not compare against HMM clustering. In this thesis we have conducted several runs with different configurations of T, C and found that for small corpora HMM alike models seem to be more appropriate. In the POS tagging task, Li and McCallum (2005) report error reductions that range from 14.74% (10k tokens training data) to 10.30% (50k tokens training data) on the Wall Street Journal collection of the Penn Treebank with similar baselines (89.96% accuracy for 10k tokens, 94.66% accuracy for 50k tokens). These results are difficult to compare to our investigations since the corpora and tagsets are different. Nevetheless, the directions are in line. Toutanova and Johnson (2007) proposed another semi-supervised POS tagging approach with a generative distributional clustering technique, where similar to the work of (Schütze, 1995) the focus lies on word context features rather than word and tag sequences. They obtain superior results compared to unsupervised methods, like EM-HMM, Bayesian-HMM, or Contrastive Estimation (Smith and Eisner, 2005). By contrast to their approach, we believe that a combination of supervised discriminative classifiers and generative features should be preferred for final tagging since for classification tasks, for example, CRFs or maximum entropy models have been shown to surpass their generative equivalents, that is, HMMs or naive Bayes, respectively. Nonetheless, it would be interesting to include

features obtained from a Toutanova and Johnson (2007)-model into a domain adaptation framework with a stacked CRF.

Presumably because joint models of syntax and semantics tend to run into complexity problems, most practitioners are either interested in a good syntax model or a topic model, and mostly sufficient results can be achieved with heuristics, the field of topic-aware syntax models has been rather sparsely addressed compared to research on each area. Despite the work of Griffiths et al. (2004), that has gained some attention and which we have used in our setting, there have also been contributions by Darling et al. (2012) and Boyd-Graber and Blei (2008). The former, called *Part-of-Speech LDA* (POSLDA), is a recent generalization of LDAHMMs (Griffiths et al., 2004) to incorporate more than one Part-of-Speech class with a topic distribution role. Their results indicate improvements of POSLDA against a Bayesian HMM in an unsupervised POS tagging setting. Knowledge from labeled data should, however, be exploited when possible, for instance, in semi-supervised or domain adaptation settings similar to that of this thesis. In this case, it remains unclear what happened if we integrated POSLDA states as features into the change of representation framework. Our results state that such a conclusion is not trivial; clustering directly into the POS class dimension with a HMM could be more appropriate than a more complicated model with topic dependencies, at least for small corpora. Integrating syntax and semantics has also been studied by Boyd-Graber and Blei (2008) with a Bayesian nonparametric model. The so-called syntactic topic model (STM) leaves the linear dimension of HMM alike models and operates with parse trees, although the authors note that it is not a full parsing model itself. Especially, the STM builds upon a given tree structure for the sentences, an assumption that we did not want to make for the setting in this thesis.

8 Future Work

In this chapter, we collect and develop ideas for future research, which can be divided into work with focus on model aspects, and work considering data and evaluation.

The LDAHMM of Griffiths et al. (2004) is an interesting approach to combine syntax and topics, however, there are obvious shortcomings of this model that should be addressed in future work. For example, the POS classes of nouns, adjectives and verbs all have topic aspects but either share the same syntax state, or are handled without topic adaptation in LDAHMMs. Darling et al. (2012) recently proposed POSLDA to overcome this issue, but the model has not yet been evaluated inside of an change of representation setting for domain adaptation. Future studies could also compare this setting to other distributional clustering approaches such as the work of Toutanova and Johnson (2007), or Chrupala (2011).

In our setting, we train HMMs or LDAHMMs respectively on the whole corpus to generate the feature representations, and subsequently we learn a CRFs on all labeled instances. This procedure is acceptable if we wish to process large batches of labeled and unlabeled documents. By contrast, if we encounter a stream of instances, that is, if documents arrive piece by piece, the current operational sequence would be obstructive. In this case, we would desire online learning algorithms, that is, algorithms that train models with only one single run over the data and update the model when new data arrives. Altering the current framework for online adaptive inference is an important future research direction.

A different direction for follow-up studies could be a combination of semi-supervision and domain adaptation. Maybe it would be useful to constrain the generative feature generation to the label distribution of the annotated training data, in a manner similar similar to Nigam et al. (1998) or Toutanova and Johnson (2007).

Obviously, our findings need to be verified on different data sets. For example, there is a recent trend to process short messages and text from social media, for example, from Twitter (Li and McCallum, 2005; Gimpel et al., 2011; Ritter et al., 2011). Social media has an inherently dynamic and diverse character which demands domain adaptation. It would also be interesting to carry out a multilingual evaluation of the approach, for instance, with German data sets.

Besides, we left out consideration of negative transfer in this work. Therefore, a next step would be to investigate negative transfer effects in this kind of domain adaptation with generative features.

9 Conclusion

In this work, we have addressed domain adaptation for Part-of-Speech tagging. The classification of words into their Parts-of-Speech is an import step of Natural Language Processing pipelines, and challenging when the input distribution of the target domain differs from the training data; a problem setting that often occurs in practice, for instance, when the training data comes from the financial domain and the model should be applied on medical texts.

We have approached domain adaptation through a change of representation and representation learning (Blitzer et al., 2006; Huang and Yates, 2009; Huang and Yates, 2010); we projected the data of both domains into a shared, low-dimensional space and provided a supervised discriminative sequence labeler with this new representation as a feature. Especially, we concentrated our investigations on a comparison of features from certain generative models: Hidden Markov Models (HMMs) and different configurations of Latent Dirichlet Allocation Hidden Markov Models (LDAHMMs) (Griffiths et al., 2004). The former models focus on a pure sequential representation of the data while the latter models additionally incorporate long-range dependencies between content words; they consider syntax and take topics and their term correlations into account.

Empirically, we found that all tested parameterizations of generative features yielded significant improvements against a traditional supervised baseline Conditional Random Field (Lafferty et al., 2001) on domains taken from the Brown corpus. This conforms to previous work on domain adaptation and highlights the importance of feature representation as well as the feasibility and benefit of unsupervised feature generation. Interestingly, our results suggest that Hidden Markov Models may be preferred to LDAHMMs for representation learning, in particular for small corpora. For a tiny corpus with only 500 labeled training sentences we detected significant improvements of HMM features compared to LDAHMM features.

Future work should apply experiments on other languages, and larger data sets. It would be interesting to investigate effects of negative transfer in this field and how it could be avoided. Another direction for upcoming studies is a combination of semi-supervision and domain adaptation.

Bibliography

Ando, Rie Kubota and Tong Zhang (Dec. 2005). "A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data". In: *Journal of Machine Learning Research* 6, pp. 1817–1853.

Ben-David, Shai, John Blitzer, Koby Crammer, and Fernando Pereira (2006). "Analysis of Representations for Domain Adaptation". In: *Advances in Neural Information Processing Systems 19*. Ed. by Bernhard Schölkopf, John C. Platt, and Thomas Hoffman. Cambridge, MA: MIT Press, pp. 137–144.

Bickel, Steffen, Michael Brückner, and Tobias Scheffer (Dec. 2009). "Discriminative Learning Under Covariate Shift". In: *Journal of Machine Learning Research* 10, pp. 2137–2155.

Blei, David M. (Apr. 2012). "Probabilistic topic models". In: *Commun. ACM* 55.4, pp. 77–84.

Blei, David M. and John D. Lafferty (July 2009). *Visualizing Topics with Multi-Word Expressions*. arXiv: `0907.1013 [(stat.ML]`.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan (2001). "Latent Dirichlet Allocation". In: *Advances in Neural Information Processing Systems 14*. Ed. by Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani. Cambridge, MA: MIT Press, pp. 601–608.

Blei, David M., Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum (2003a). "Hierarchical Topic Models and the Nested Chinese Restaurant Process". In: *Advances in Neural Information Processing Systems 16*. Ed. by Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf. Cambridge, MA: MIT Press.

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

Blitzer, John, Ryan McDonald, and Fernando Pereira (2006). "Domain adaptation with structural correspondence learning". In: *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. EMNLP '06. Sydney, Australia: Association for Computational Linguistics, pp. 120–128.

Boyd-Graber, Jordan and David M. Blei (2008). "Syntactic Topic Models". In: *Advances in Neural Information Processing Systems 21*. Ed. by Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou. Curran Associates, Inc., pp. 185–192.

Brants, Thorsten (Apr. 2000). "TnT – A Statistical Part-of-Speech Tagger". In: *Proceedings of the Sixth Conference on Applied Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, pp. 224–231.

Carpenter, Bob (Sept. 2010). *Integrating out multinomial parameters in latent Dirichlet allocation and naive Bayes for collapsed Gibbs sampling*. Tech. rep. Version 1.4. LingPipe. URL: http://lingpipe.files.wordpress.com/2010/07/lda3.pdf.

Charniak, Eugene, Curtis Hendrickson, Neil Jacobson, and Mike Perkowitz (1993). "Equations for Part-of-Speech Tagging". In: *Proceedings of the 11th National Conference on Artificial Intelligence*. Ed. by Richard Fikes and Wendy G. Lehnert. AAAI Press / The MIT Press, pp. 784–789.

Chrupala, Grzegorz (2011). "Efficient induction of probabilistic word classes with LDA". In: *Proceedings of 5th International Joint Conference on Natural Language Processing*. Chiang Mai, Thailand: Asian Federation of Natural Language Processing, pp. 363–372.

Chuang, Jason, Christopher D. Manning, and Jeffrey Heer (2012). "Termite: Visualization Techniques for Assessing Textual Topic Models". In: *AVI '12: Proceedings of the International Working Conference on Advanced Visual Interfaces*. Ed. by Genny Tortora, Stefano Levialdi, and Maurizio Tucci. Capri Island, Italy: ACM, pp. 74–77.

Darling, William M. (2011). *A Theoretical and Practical Implementation Tutorial on Topic Modeling and Gibbs Sampling*. Tech. rep. University of Guelph. URL: http://www.uoguelph.ca/~wdarling/research/papers/TM.pdf.

Darling, William M., Michael J. Paul, and Fei Song (2012). "Unsupervised Part-of-Speech Tagging in Noisy and Esoteric Domains With a Syntactic-Semantic Bayesian HMM". In: *Proceedings of the Workshop on Semantic Analysis in Social Media*. Avignon, France: Association for Computational Linguistics, pp. 1–9.

Darwiche, Adnan (2008). "Bayesian Networks". In: *Handbook of knowledge representation*. Ed. by Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter. Foundations of Artificial Intelligence. Amsterdam: Elsevier. Chap. 11, pp. 476–509.

Deerwester, Scott C., Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman (1990). "Indexing by Latent Semantic Analysis". In: *Journal of the American Society for Information Science (JASIS)* 41.6, pp. 391–407.

Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1, pp. 1–38.

Francis, W. N. and H. Kucera (1979). *Brown Corpus Manual*. Tech. rep. Department of Linguistics, Brown University, Providence, Rhode Island, US. URL: http://icame.uib.no/brown/bcm.html.

Geman, Stuart and Donald Geman (1984). "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 6.6, pp. 721–741.

Gimpel, Kevin, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith (2011). "Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (Short Papers)*. The Association for Computer Linguistics, pp. 42–47.

Goldwater, Sharon and Thomas L. Griffiths (2007). "A fully Bayesian approach to unsupervised part-of-speech tagging". In: *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*. Ed. by John A. Carroll, Antal van den Bosch, and Annie Zaenen. The Association for Computational Linguistics.

Griffiths, Thomas L. and M. Steyvers (2002). "A probabilistic approach to semantic representation". In: *Proceedings of the 24th Annual Conference of the Cognitive Science Society*.

Griffiths, Thomas L. and Mark Steyvers (2004). "Finding scientific topics". In: *Proceedings of the National Academy of Sciences of the United States of America* 101.Suppl 1, pp. 5228–5235. eprint: `http://www.pnas.org/content/101/suppl.1/5228.full.pdf+html`.

Griffiths, Thomas L., Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum (2004). "Integrating Topics and Syntax". In: *Advances in Neural Information Processing Systems 17*. Ed. by Lawrence K. Saul, Yair Weiss, and Léon Bottou. Cambridge, MA: MIT Press, pp. 537–544.

Heckman, James J. (1977). *Sample Selection Bias As a Specification Error (with an Application to the Estimation of Labor Supply Functions)*. Working Paper 172. National Bureau of Economic Research. URL: `http://www.nber.org/papers/w0172`.

Heinrich, Gregor (2009). *Parameter estimation for text analysis*. Tech. rep. Fraunhofer IGD. URL: `http://www.arbylon.net/publications.html`.

Hofmann, Thomas (1999). "Probabilistic latent semantic indexing". In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '99. Berkeley, California, United States: ACM, pp. 50–57.

Huang, Fei and Alexander Yates (2009). "Distributional Representations for Handling Sparsity in Supervised Sequence-Labeling". In: *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Ed. by Keh-Yih Su, Jian Su, and Janyce Wiebe. The Association for Computer Linguistics, pp. 495–503.

Huang, Fei and Alexander Yates (2010). "Exploring Representation-Learning Approaches to Domain Adaptation". In: *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*. Association for Computational Linguistics, pp. 23–30.

Jiang, Jing (2008). "Domain Adaptation in Natural Language Processing". PhD thesis. University of Illinois at Urbana-Champaign.

Johnson, Mark (2007). "Why Doesn't EM Find Good HMM POS-Taggers?" In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: Association for Computational Linguistics, pp. 296–305.

Jurafsky, Daniel and James H. Martin (2009). *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. 2nd ed. Prentice-Hall-series in artificial intelligence. Upper Saddle River, NJ: Prentice Hall, p. 1024.

Koller, Daphne and Nir Friedman (2009). *Probabilistic graphical models : principles and techniques*. Cambridge, MA: MIT Press.

Lafferty, John D., Andrew McCallum, and Fernando C. N. Pereira (2001). "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *Proc. 18th International Conf. on Machine Learning*, pp. 282–289.

Lee, Lillian (1999). "Measures of distributional similarity". In: *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. College Park, Maryland: Association for Computational Linguistics, pp. 25–32.

Li, Wei and Andrew McCallum (2005). "Semi-supervised sequence modeling with syntactic topic models". In: *Proceedings of the 20th national conference on Artificial intelligence - Volume 2*. AAAI'05. Pittsburgh, Pennsylvania: AAAI Press, pp. 813–818.

MacKay, David J. C. (2005). *Information Theory, Inference, and Learning Algorithms*. 4. printing. Available from `http://www.inference.phy.cam.ac.uk/mackay/itila/`. Cambridge: Cambridge University Press.

Marcus, Mitchell P., Mary Ann Marcinkiewicz, and Beatrice Santorini (June 1993). "Building a large annotated corpus of English: the penn treebank". In: *Comput. Linguist.* 19.2, pp. 313–330.

Nigam, Kamal, Andrew McCallum, Sebastian Thrun, and Tom M. Mitchell (1998). "Learning to Classify Text from Labeled and Unlabeled Documents". In: *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference*. Ed. by Jack Mostow and Chuck Rich. AAAI Press / The MIT Press, pp. 792–799.

Nocedal, Jorge (1980). "Updating Quasi-Newton Matrices with Limited Storage". English. In: *Mathematics of Computation* 35.151, pp. 773–782.

Okazaki, Naoaki (2007). *CRFsuite: a fast implementation of Conditional Random Fields (CRFs)*.

Pan, Sinno Jialin and Qiang Yang (2010). "A Survey on Transfer Learning". In: *IEEE Trans. Knowl. Data Eng.* 22.10, pp. 1345–1359.

Rabiner, Lawrence R. (1989). "A tutorial on hidden markov models and selected applications in speech recognition". In: *Proceedings of the IEEE*. Vol. 77. 2, pp. 257–286.

Ritter, Alan, Sam Clark, Mausam, and Oren Etzioni (2011). "Named Entity Recognition in Tweets: An Experimental Study". In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. ACL, pp. 1524–1534.

Rosenstein, Michael T., Zvika Marx, Leslie Pack Kaelbling, and Thomas G. Dietterich (2005). "To transfer or not to transfer". In: *NIPS 2005 Workshop on Inductive Transfer: 10 Years Later*.

Russell, Stuart J. and Peter Norvig (2010). *Artificial intelligence : a modern approach*. 3. ed., internat. ed. Prentice Hall series in artificial intelligence. Boston: Pearson.

Schütze, Hinrich (1995). "Distributional Part-of-Speech Tagging". In: *CoRR* cmp-lg/9503009.

Smith, Noah A. and Jason Eisner (2005). "Contrastive Estimation: Training Log-Linear Models on Unlabeled Data". In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*. Ann Arbor, Michigan, pp. 354–362.

Steyvers, Mark and Thomas L. Griffiths (2007). "Probabilistic topic models". In: *Latent Semantic Analysis: A Road to Meaning*. Ed. by T. Landauer, S. Dennis McNamara, and W. Kintsch. Mahwah, NJ: Erlbaum. Chap. 21.

Sutton, C. and A. McCallum (Nov. 2010). *An Introduction to Conditional Random Fields*. arXiv: `1011.4088 [stat.ML]`.

Toutanova, Kristina and Mark Johnson (2007). "A Bayesian LDA-based model for semi-supervised part-of-speech tagging". In: *Advances in Neural Information Processing Systems 20*. Ed. by John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis. Curran Associates, Inc.

Toutanova, Kristina, Dan Klein, Christopher D. Manning, and Yoram Singer (2003). "Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network". In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. Association for Computational Linguistics, pp. 173–180.

Wallach, Hanna M., Iain Murray, Ruslan Salakhutdinov, and David M. Mimno (2009). "Evaluation methods for topic models". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. Ed. by Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman. Vol. 382. ACM International Conference Proceeding Series. ACM, p. 139.