# Bayesian Unsupervised Labeling of Web Document Clusters

by

Ting Liu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2011

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

**Abstract**

Information technologies have recently led to a surge of electronic documents in the form of emails, webpages, blogs, news articles, etc. To help users decide which documents may be interesting to read, it is common practice to organize documents by categories/topics. A wide range of supervised and unsupervised learning techniques already exist for automated text classification and text clustering. However, supervised learning requires a training set of documents already labeled with topics/categories, which is not always readily available. In contrast, unsupervised learning techniques do not require labeled documents, but assigning a suitable category to each resulting cluster remains a difficult problem. The state of the art consists of extracting keywords based on word frequency (or related heuristics).

In this thesis, we improve the extraction of keywords for unsupervised labeling of document clusters by designing a Bayesian approach based on topic modeling. More precisely, we describe an approach that uses a large side corpus to infer a language model that implicitly encodes the semantic relatedness of different words. This language model is then used to build a generative model of the cluster in such a way that the probability of generating each word depends on its frequency in the cluster as well as the frequency of its semantically related words. The words with the highest probability of generation are then extracted to label the cluster.

In this approach, the side corpus can be thought as a source of domain knowledge or context. However, there are two potential problems: processing a large side corpus can be time consuming and if the content of this corpus is not similar enough to the cluster, the resulting language model may be biased. We deal with those issues by designing a Bayesian transfer learning framework that allows us to process the side corpus just once offline and to weigh its importance based on the degree of similarity with the cluster.

**Acknowledgements**

First and foremost I would like to offer my deepest gratitude to my supervisor, Pascal Poupart. This thesis would not have been possible without him. His guidance in the area of Machine Learning and his support from the beginning to the end has enabled me to develop this thesis. His encouragement also helped me to overcome all difficulties through the bulk of this work.

I would like extend my appreciation to Ruitong Huang. He deserves particular acknowledgement for his math expertise that helped me clear my confusion on several occasions. Moreover, I would like to thank him for his constant encouragement and support.

Many thanks go in particular to my project colleagues, Andy Chiu and Finnegan Southey, at Google Inc. They offered much valuable advice and insight throughout my work.

I gratefully thank Professor Charles Clarke and Doctor Finnegan Southey for their constructive comments regarding my thesis. I am thankful that they accepted to be members of the reading committee.

I would like to show my gratitude to my friends, Hongsen Liu, Ying Liu, Wenjie Xiao, Johan Harjono and all those who have made invaluable comments about the thesis. They have offered their support in a number of ways. Their friendship helped me to never feel alone when facing some challenges.

Lastly, I want to say thank you to my mother, Liyun Liu, my husband, Xiaodong Zhou and my son, Jason Zhou. Your unlimited support allowed me to understand the true meaning of life.

## Dedication

I would like to dedicate my thesis to my beloved grandmother.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

How many years did it take for the radio to reach 50 millions users? The answer is: 38 years. TV took 13 years, the Internet took 4 years, and the iPod took 3 years. Facebook accumulated 100 millions users in less than nine months [1]. With the advent of the World Wide Web (a.k.a. WWW), the internet acts as an accelerator to reshape human lives. More and more people use the internet for reading news, exchanging ideas, sharing pictures, shopping and so on. Today, social networks have become an important part of people's daily life. There are amazing web services supplied by so many companies. For instance, Google allows us to retrieve information quickly via her powerful search engine, Amazon created an e-commerce web application that provides an online shopping platform, and Facebook has features, such as the "like" button, to allow individuals to share what they like. The number of web users grows fast, at the same time, information on web grows fast too. How can this web data be used? Most web service vendors use a wide variety of clustering algorithms to group web pages with similar content. Clustering is an important task, however, it is not fully automatized because the developers must still look at the documents to understand the main idea of each cluster. Cluster labeling is the main obstacle that prevents the deployment of new web services that would automatically analyze and label groups of web pages. Currently, how to automatically and efficiently to label a cluster is a hot topic in Artificial Intelligence (AI) and Information Retrieval (IR). In order to analyze text data

[1] http://searchenginewatch.com/article/2066771/Social-Media-Fad-or-Revolution

and generate a label, we consider machine learning techniques. More precisely, in this thesis, we design Bayesian machine learning models for text analysis and cluster labeling.

To tackle this problem, we assume that a search engine company has a large text corpus which consists of indexed web pages. Although each web page may contain a title, some text, pictures and hyperlinks, we just focus on the basic case where the data is a sequence of words. And, in this thesis, we will call this kind of web content as a document. As new web pages are created every day, a search engine company collects new documents and groups them into several clusters. We propose an efficient method to generate a word for labeling each one of those new clusters within an acceptable amount of time. This label word should help developers to understand what is the cluster "talking" about. For example, if the new cluster contains some documents related to baseball, some documents related to swimming, and the rest related to cycling, ideally we would like to generate "sport" as a label for this new cluster. Furthermore, the word "sport" may not exist in the new cluster at all. However, we would most likely find that word in the large side corpus.

## 1.1 Research Contribution

In Artificial Intelligence, there are a lot of models for text analysis. Two new Transfer Learning models are introduced in this thesis. They are LDA-Transfer learning model (LDA-Trans) and HDP-Transfer learning model (HDP-Trans). Both of them are based on topic modeling and Bayesian data analysis. The main idea is to use a large side corpus to learn a topic distribution for each word. This topic distribution can be thought as characterizing the semantics of each word. Next, we transfer the semantics implied by the topic assignments to the new cluster. Furthermore, we automatically weigh the implied semantics based on the degree of similarity between the side corpus and the cluster to be labeled as a prior to analysis the target cluster. During this step, the LDA-Trans model uses Latent Dirichlet Allocation (LDA), which assumes a fixed number of topics to estimate topic distributions, and HDP-Trans uses Hierarchical Dirichlet Processes (HDP), which modifies the number of topics according to the data. There are three benefits to this approach. First, when the cluster to be labeled is small, the transferred topic distributions

improve the robustness of the language model of the cluster. Second, since the topic distributions can be pre-computed on the large side corpus off line, new clusters can be processed efficiently. Three, given a topic distribution for each word, we estimate the most important topics of the cluster and label the cluster according to the word that is the most likely to be generated by those topics. As a result, this word may not exist in the cluster itself, but may come from the side corpus.

## 1.2 Thesis Organization

This thesis consists of five chapters. Chapter one introduces the problem and domain. Chapter two reviews several existing language models, such as frequency based models, N-gram models, topic models and some variants of these models. Chapter three introduces the theoretical motivations and intuitions of our new models: LDA-Trans and HDP-Trans. The accuracy will be evaluated with perplexity graphs. Chapter four describes how to use LDA-Trans and HDP-Trans to label a cluster of documents by producing a list of candidate words. Chapter five concludes our research with a brief summary and some directions for future work.

# Chapter 2

# Background

## 2.1 Language Modeling

Language modeling can be used for speech recognition, handwriting recognition, spelling correction, foreign language reading/writing aid, machine translation and so on. A language model is defined as a probability distribution over word sequences. In "Foundations of Statistical Natural Language Processing" [11], Manning and Schutze point out that words in a corpus can be thought as random discrete data, which is generated according to some unknown probability distribution. For example, if we have a string "I like Chinese food" denoted as $\mathbf{w}$ = "$w_0 w_1 w_2 w_3$", the language model is $P(\mathbf{w})$. In other words, the language model attempts to reflect how frequently a string $\mathbf{w}$ occurs as a sentence.

Nowadays, a popular class of models is based on frequency. This class of models uses term occurrence to weight each word while ignoring any relationship between words. Since words do not occur in a random order, another class of language models consists of n-grams, such as uni-gram, bi-gram and tri-gram, which predict each word given the $n-1$ previous words. The parameter $n-1$ indicates how many previous words are used to model the next word. However, the n-gram models only focus on the likelihood of short sequences of words, they ignore how words are semantically related to each other. Another class of language models consists of topic models. Topic models are based on the "bag-of-word" assumption, which means that words can be thought as sampled from a set (i.e. bag) of words without

any order. For example, the vector representation of "Lucy is taller than Sam" is same with the vector representation of "Sam is taller than Lucy". Topic models also include latent components which are often referred to as topics since they tend to cluster together words related to the same topic. Such latent topics can serve as a basis to measure how closely related words are by meaning. While there are many ways of modeling topics, Bayesian approaches provide a principled statistical framework to do this. Moreover, a hierarchical non parameter data model can provide more precise result. In particular, we will review Latent Dirichlet Allocation (LDA) and Hierarchical Dirichlet Processes (HDP), which will be used in our new transfer learning models in this thesis.

Throughout this thesis, we assume that there is a corpus (a set of documents) and that each document consists of a list of words. The following notation will be used. Upper case letters will denote variables: $T$ for topics, $W$ for the sequence of words, and $D$ for documents. The calligraphic versions of the same letters denote their domain: $\mathcal{T} = dom(T)$, $\mathcal{W} = dom(W)$, and $\mathcal{D} = dom(D)$. Lower case letters indicate values for the variables: $t \in \mathcal{T}, w \in \mathcal{W}$, and $d \in \mathcal{D}$. Sets are denoted by bold letters: $\mathbf{T} = \{T_0, T_1, ..., T_{K-1}\}$ is a set of variables and $\mathbf{t} = \{t_0, t_1, ..., t_{K-1}\}$ is a joint assignment of values for $\mathbf{T}$. We reserve $K$ to indicate the number of topics and $N$ to indicate the number of words in each document. A set subscripted by a negative index denotes all items in the set except the one subscripted: $\mathbf{T}_{-k} = \{T_1, ..., T_{k-1}, T_{k+1}, ..., T_K\}$. Some specific lower case letters will be used as indices in this thesis. For example, $i$ is used as an index for words in a document (or sequence); $d$ is used as an index for the documents in a corpus or a cluster; and $k$ is used as an index for topics.

## 2.2 Bayesian Learning and Plate Notation

The essential characteristic of Bayesian methods is their explicit use of probabilities to quantify uncertainty in inferences based on data [5]. In information retrieval, when we analyze a corpus $\mathcal{D}$, it is very useful to build a language model.

Let $\theta =< \theta_0, ..., \theta_{N-1} >$ be the set of parameters of the language model. Since we don't know the values of the parameters initially, we treat them as random variables. Once we

have observed some data (i.e. a sequence of words $\mathbf{w}$), then we can compute a posterior distribution $P(\theta|\mathbf{w})$. More precisely, this posterior can be computed by Bayes' rules:

$$p(\theta|\mathbf{w}) = \frac{p(\theta, \mathbf{w})}{p(\mathbf{w})} = \frac{p(\theta)p(\mathbf{w}|\theta)}{p(\mathbf{w})}, \tag{2.1}$$

where $p(\mathbf{w}) = \sum_\theta p(\theta)p(\mathbf{w}|\theta)$, and the sum is over all possible values of $\theta$. Since $p(\mathbf{w})$ does not depend on $\theta$, we can get the unnormalized posterior density:

$$p(\theta|\mathbf{w}) \propto p(\theta)p(\mathbf{w}|\theta). \tag{2.2}$$

These equations are at the core of Bayesian inference. Most models in this thesis describe a joint distribution $p(\theta, \mathbf{w})$ and perform the necessary computations to summarize $p(\theta|\mathbf{w})$ in appropriate ways.

The plate notation, as in Figure 2.1, is a compact graphical representation to denote Bayesian networks with identical subgraphs that repeat. Instead of repeatedly drawing variables with the same dependency structure, this method uses a plate (or a rectangle) to group variables in a subgraph with an index in the bottom right corner. Each unknown (latent) variable is represented as a circle, and each observed variable, such as $w_{di}$, is represented as a shaded circle. Edges indicate probabilistic dependencies between variables. In this thesis, the plate notation is used to illustrate several n-gram models and topic models.

## 2.3 Frequency models

### 2.3.1 Tf-Idf

In information retrieval (IR), researchers developed many techniques to extract relevant keywords. A popular approach is called *Tf-Idf* [13]. There are many forms for term frequency $Tf$, the basic one is using raw term frequency of each word $w_i$ in each document $d$ as follows:

$$Tf_{w_i,d} = n_{w_i,d}, \tag{2.3}$$

where $n_{w_i,d}$ is the number of occurrences of word $w_i$ in document $d$. In some situations, it would be preferable for $Tf$ to grow at a slower rate than linear, so Salton created another from that is logarithmic:

$$Tf_{w_i,d} = \begin{cases} log(n_{w_i,d}) + 1, & \text{if } n_{w_i,d} > 0; \\ 0, & \text{otherwise.} \end{cases} \tag{2.4}$$

By this equation, if a word has higher frequency in a specific document, that word should have higher *Tf* weight.

*Idf* stands for Inverse document frequency. The classic equation of *Idf* is:

$$Idf_{w_i} = \log_{10}(|\mathcal{D}|/|\{d : w_i \in d, d \in D\}|), \tag{2.5}$$

where $|\mathcal{D}|$ is the number of documents in corpus $\mathcal{D}$ and $|\{d : w_i \in d, d \in D\}|$ is the number of documents that include the word $w_i$. For instance, the word "the" happens in every document in corpus $\mathcal{D}$. If there are 1,000,000 documents in $\mathcal{D}$, and $|\{d : w_i \in d, d \in D\}| = 1,000,000$ too, then the $Idf$ value of "the" is 0. On the other hand, if a word occurs only in some of documents, the *Idf* value of that word should be higher.

In the end, the *Tf-Idf* value is obtained as follows:

$$Tf\text{-}Idf = Tf * Idf. \tag{2.6}$$

This method uses *Tf* value to find more frequent words in a document, and *Idf* value to prune common words that are meaningless in the corpus. However, if the corpus is related to only one topic, then the most relevant word may happen in every document, but it will be pruned by *Idf*. As a result, *Tf-Idf* is not suitable to label a cluster of documents.

### 2.3.2 A metric of word relevance

A metric of word relevance is proposed by Zhou and Slater in [19]. This method estimates the relevance of uni-grams (or single words). Since meaningful words tend to cluster in certain parts of the text instead of being randomly distributed throughout the text, this method computes a measure of the tightness of the clusters of each word. Words that

tend to cluster more frequently are given a higher relevance score. The approach starts by building a list of positions $l_w = -1, t_1, t_2, ..., t_m, n$ where $t_i$ denotes the position of the $i^{th}$ occurrence of word w and n is the total number of words in the document. Then we compute the average distance $\hat{\mu}$ between successive occurrences of $w$ as follows:

$$\hat{\mu} = \frac{n+1}{m+1}, \tag{2.7}$$

where $m$ is the number of occurrences of word $w$. We can compare this global average distance to a local average distance $d(t_i)$ for each position $t_i$:

$$d(t_i) = \frac{t_{i+1} + t_{i-1}}{2}, i = 1, ..., m. \tag{2.8}$$

Occurrences of $w$ where $d(t_i) < \hat{\mu}$ are said to be part of a cluster since they are closer to their neighbours than average. More precisely, we denote by $\sigma(t_i)$ a function that verifies whether position $t_i$ is part of a cluster as follows:

$$\sigma(t_i) = \begin{cases} 1, & \text{if } d(t_i) < \hat{\mu}; \\ 0, & \text{otherwise.} \end{cases} \tag{2.9}$$

Next, we measure the reduction $\nu(t_i)$ in average distance of the cluster points with respect to the global average distance as follows:

$$\nu(t_i) = \frac{\hat{\mu} - d(t_i)}{\hat{\mu}}. \tag{2.10}$$

Finally, we estimate the relevance of a word by computing the average reduction in average distance for the cluster:

$$\Gamma(w) = 1/m * \sum_{i=1}^{m} \sigma(t_i) * \nu(t_i) \tag{2.11}$$

While this method is simple and efficient it tends to give low scores to relevant words that are frequent or rare because they tend not to cluster.

### 2.3.3 Alternative scoring functions

Ventura and Silva [17] also proposed some scoring functions to measure the relevance of uni-grams. In this section we describe their Score function, SPQ function and Islands method.

Since relevant words tend to co-occur with a small set of predecessor and successor words, the Score function estimates the relevance of a word by measuring the variation of the frequency of its immediate neighbors. This is done by first defining the successor Score function $Sc_{suc}(w)$ for the words that immediately follow $w$:

$$Sc_{suc}(w) = \sqrt{\frac{1}{\parallel \gamma \parallel -1} \sum_{y_i \in \gamma} (\frac{p(w, y_i) - p(w, \cdot)}{p(w, \cdot)})^2} \tag{2.12}$$

Here, the $\gamma$ is the set of distinct words in the corpus and $\parallel \gamma \parallel$ denotes the size of $\gamma$. Also, $p(w, y_i)$ is the probability that $y_i$ follows $w$ and $p(w, \cdot)$ is the average probability of the successor words of $w$. The joint term $p(w, y_i)$ is defined as follows:

$$p(w, y_i) = \frac{f(w, y_i)}{N} \tag{2.13}$$

and

$$p(w, \cdot) = \frac{1}{\parallel \gamma \parallel -1} \sum_{y_i \in \gamma} p(w, y_i) \tag{2.14}$$

Here, $N$ denotes the number of occurrences of word $w$ in the corpus and $f(w, y_i)$ is the frequency of bi-gram $(w, y_i)$. Similarly, we can define a predecessor score function:

$$Sc_{pre}(w) = \sqrt{\frac{1}{\parallel \gamma \parallel -1} \sum_{y_i \in \gamma} (\frac{p(y_i, w) - p(\cdot, w)}{p(\cdot, w)})^2} \tag{2.15}$$

Finally, by taking the average of the predecessor and successor scores, we obtain an overall score that can be used to estimate the relevance of words:

$$Sc(w) = \frac{Sc_{pre}(w) + Sc_{suc}(w)}{2}. \tag{2.16}$$

According to this method, if the word $w$ is followed or following a lot of different words, then the score of this word is pretty low; if the word is frequent and has a small set of successors and predecessors, then the score will be high.

The Successor-Predecessor Quotient (SPQ) measure is another statistical metric that measures the importance of the word $w$ based on the quotient between its number of distinct

successors and its number of distinct predecessors. The SPQ value can be obtained by the equation:

$$SPQ(w) = \frac{N_{suc}(w)}{N_{ant}(w)}, \tag{2.17}$$

where $N_{suc}(w)$ and $N_{ant}(w)$ represent the number of distinct successors and predecessors of word $w$ in the corpus. Experimental results in [17] show that SPQ is better than Sc.

The Islands method extracts a word if it is more relevant than its neighbor words. In this approach we compute the average relevance of the predecessors and successors as follows:

$$Avg_{pre}(w) = \sum_{y_i \in \{predecsofw\}} p(y_i, w) * r(y_i), \tag{2.18}$$

$$Avg_{suc}(w) = \sum_{y_i \in \{succesofw\}} p(w, y_i) * r(y_i), \tag{2.19}$$

where $p(y_i, w)$ means the probability of occurrence of bigram $(y_i, w)$ and $r(y_i)$ is the relevance value given by some generic $r(\cdot)$ metric, which could be the Score function or SQP function. A word is considered relevant if it satisfies:

$$r(w) \geq 0.9 * max(Avg_{pre}(w), Avg_{suc}(w)). \tag{2.20}$$

The above techniques identify potentially relevant words based on different properties than frequency. While these properties are interesting, it is not clear that they extract good words for the labels. Ideally, the computer should understand the meaning of the words, sentences and documents to extract good labels. To that effect, the section 2.5 reviews topic models that take a step in this direction.

## 2.4 n-gram Models

An n-gram is a subsequence of $n$ words $(w_0, ..., w_{(n-1)})$ from a given sequence $\mathbf{w} = (w_0, ..., w_{(N-1)})$. Since it is generally difficult to model $P(\mathbf{w})$ directly, we can apply the chain rule as follows:

$$P(\mathbf{w}) = P(w_0)P(w_1|w_0)...P(w_{N-1}|w_0...w_{N-2}). \tag{2.21}$$

When we condition each word on at most $n-1$ previous words, we obtain an n-gram model. The uni-gram and bi-gram models are illustrated in Figure 2.1 using the plate notation.

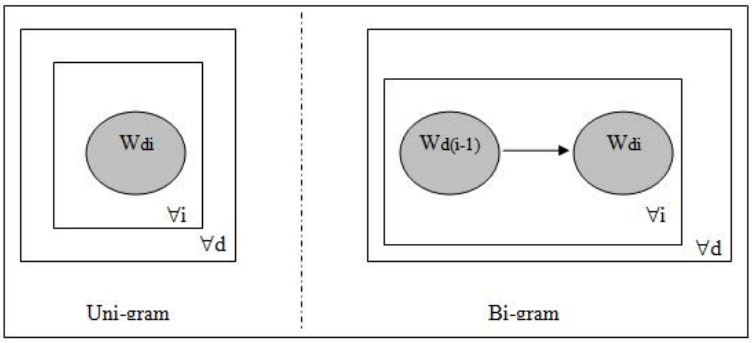


Figure 2.1: Uni-gram and Bi-gram

### 2.4.1 Uni-gram Model

If $n$ equals one, it is called a uni-gram model. When we get a word sequence $\mathbf{w}$ with $N$ tokens, we can denote it as $\mathbf{w}$ = "$w_0 w_1 ... w_{N-1}$". Then the likelihood of this sequence is assumed to be the product of the probabilities of each word separately:

$$P(\mathbf{w}) = \prod_{i=0}^{N-1} P(w_i), \tag{2.22}$$

It is common to approximate $P(w_i)$ by the relative frequency of $w_i$:

$$P(w_i) = \frac{N_{w_i}}{N}, \tag{2.23}$$

Here, $N_{w_i}$ is the frequency of word $w_i$ in corpus $\mathcal{D}$. This model makes a strong assumption that each word is sampled independently and identically. Hence, this model ignores the relationship between words. For example, when analyzing the capitalization of some words, such as "new", it is important to consider neighboring words, "new" is widely used as an adjective, but, it can also be used in a compound noun, for instance, "New York". So, higher level n-gram models are needed to consider short sequences of several words that will avoid this kind of concern.

### 2.4.2 Bi-gram Model

When $n$ equals two, we get a bi-gram model. Consider a sequence of words $\mathbf{w}$ of length $N$. The probability of this sequence is computed as follows for the bi-gram model:

$$P(\mathbf{w}) = P(w_0)P(w_1|w_0)...P(w_{N-1}|w_{N-2}) \tag{2.24}$$

$$= P(w_0)\prod_{i=1}^{N-1} P(w_i|w_{i-1}), \tag{2.25}$$

Let $N_{w_i}$ be the frequency of word $w_i$ in $\mathbf{w}$, and let $N_{w_i|w_{i-1}}$ be the number of occurrences of word $w_i$ following word $w_{i-1}$ in $\mathbf{w}$. When training by maximum likelihood $P(w_i|w_{i-1}) = N_{w_i|w_{i-1}}/N_{w_{i-1}}$. So, with a bi-gram model, "new" and "New" could be separated because of their neighbor words.

For higher order of n-grams, such as tri-gram where $n = 3$, we need to calculate the likelihood by taking a product of larger conditionals $P(w_i|w_{i-2}, w_{i-1})$.

### 2.4.3 Bayesian Smoothing

After training a uni-gram or a bi-gram model by maximum likelihood, it is possible that some words or some pairs of words in the test set do not occur in the training set, leading to zero probabilities in the test step. There are some Bayesian smoothing techniques that can be used to avoid this problem. We start with a prior over $\theta$, which we assume to be a Dirichlet distribution with hyperparameters $\alpha =< \alpha_0, ..., \alpha_{|W|-1} >$. For uni-gram, we get:

$$P(w = w_i|D) = \frac{N_{w_i} + \alpha_i}{N + \sum_{i=0}^{|\mathcal{W}|} \alpha_i}, \tag{2.26}$$

Similarly, for a bi-gram model, we can get:

$$P(w_i|w_{i-1}) = \frac{N_{w_i|w_{i-1}} + \alpha_i}{N_{w_{i-1}} + \sum_{i=0}^{|\mathcal{W}|} \alpha_i}, \tag{2.27}$$

Lidstone [9] and Jeffreys [7] proposed to use $\alpha_i = 1$ for each $i$, which corresponds to a uniform prior, so we get the following equation:

$$P(w_i|w_{i-1}) = \frac{P(w_{i-1}w_i)}{P(w_{i-1})} \tag{2.28}$$

$$= \frac{N_{w_i|w_{i-1}} + 1}{N_{w_{i-1}} + |\mathcal{W}|}. \tag{2.29}$$

When we consider two sequences $\mathbf{w}$ = "$w_0w_1$" and $\mathbf{w}'$ = "$w_0w_2$", if both of them do not occur in the training set, by the above smoothing method, $P(w_1|w_0)$ and $P(w_2|w_0)$ are the same. However, if word $w_1$ occurs more frequently in the training set than word $w_2$, it is natural to expect that $\mathbf{w}$ should be generated with higher probability than $\mathbf{w}'$. So, another simple smoothing technique consists of using a weight $\lambda$ to smooth n-grams via (n-1)-grams, such as smoothing a bi-gram via a uni-gram [8]. We can create a predictive distribution smoothed by weight $\lambda$ as follows:

$$P(w_i|w_{i-1}, \mathcal{W}) = \lambda N_{w_i}/N + (1-\lambda)N_{w_i|w_{i-1}}/N_{w_{i-1}} \tag{2.30}$$

$$= \lambda f_{w_i} + (1-\lambda) f_{w_i|w_{i-1}}. \tag{2.31}$$

Here, we can use cross validation to estimate the weight $\lambda$.

## 2.5 Topic models

When writing a document, it is common practice to first jot down (perhaps in point form) what are the main ideas/topics that we want to write. After that, we flesh out those ideas with full sentences and paragraphs. Inspired by this two step approach, topic models consist of formal probabilistic generative models that first sample some topics from which some words are sampled. Here topics, really correspond to abstract latent components, however in many situations they can be interpreted as topics. In this section, we introduce some popular topic models and their variants.

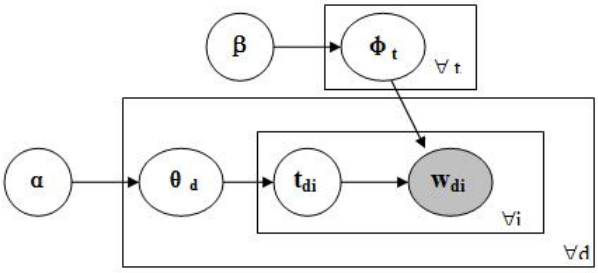


Figure 2.2: plate notation of LDA

### 2.5.1 Latent Dirichlet Allocation

When we get a corpus, and we know that certain topics characterize the documents, it is an interesting problem to classify the words into topics, which indicate their possible meanings. Latent Dirichlet Allocation is a popular model based on Bayesian networks to infer the underlying topics of text data.

Latent Dirichlet Allocation (LDA) was introduced by Blei et al. [1]. It is a three level generative probabilistic model for a collection of discrete data such as a text corpus $\mathcal{D}$ of documents $d$. In this model, there is a set $\mathcal{T}$ of topics $t$ where each topic is a multinomial distribution over a dictionary $\mathcal{W}$ of words $w$. Moreover, each document $d$ is a multinomial mixture over the topic set $\mathcal{T}$. As Figure 2.2 shows, the variable $w_{di}$ is the only observed variable in this model. Here, $w_{di}$ denotes the $i^{th}$ word in document $d$. The latent variable $t_{di}$ is the topic assignment for the word $w_{di}$. $\theta_d$ is the topic vector $< \theta_{d1}, \theta_{d2}, ..., \theta_{d|\mathcal{T}|} >$ for document $d$. Finally, $\phi$ is a $k \times |\mathcal{W}|$ matrix where each row is denoted by $\phi_t$ which corresponds to a distribution over words for each topic $t$. Here, the ranges of $\theta$ and $\phi$ is $[0, 1]$. The prior for the word distribution of each topic is typically set to a symmetric Dirichlet with hyperparameters $(\beta, ..., \beta) \in \mathbb{R}_+^{|\mathcal{W}|}$ and the prior for the topic distribution of each document is also set to a symmetric Dirichlet with hyperparameters $(\alpha, ..., \alpha) \in \mathbb{R}_+^{|\mathcal{T}|}$.

The Dirichlet distribution is the conjugate prior distribution for the parameters of the multinomial distribution [5]. Let $\theta$ denote a random vector with $K$ elements such that the sum of all elements in this vector is one. So, each element $\theta_k$ is the probability of event $k$.

Then, under the Dirichlet model with parameter vector $\alpha$, the probability density is:

$$f(\theta_1, ..., \theta_K; \alpha_1, ..., \alpha_K) = \frac{1}{Beta(\alpha)} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1}, \tag{2.32}$$

and, the Beta function can be expressed in terms of Gamma functions as:

$$Beta(\alpha) = \frac{\prod_{k=1}^{K} \Gamma(\alpha_k)}{\Gamma(\sum_{k=1}^{K} \alpha_k)}. \tag{2.33}$$

The generative process for each document $d$ in corpus $\mathcal{D}$ is:

- sample $\theta_d$ from the symmetric Dirichlet prior (with hyperparameter $\alpha$).
- for each of the words $w_{di}$ where $1 \leq i \leq N$, $N$ is the length of the sequence of words in document $d$
  1. sample topic $t_{di}$ from Multinomial($\theta$)
  2. sample a word $w_{di}$ from $P(w_{di}|t_{di}, \beta)$

Given a corpus of documents, we can use Gibbs sampling for inference and to estimate the parameters. In Gibbs sampling, the next state is reached by sequentially sampling all variables from their distribution when conditioned on the current values of all other variables and the data [10]. A collapsed Gibbs sampler for LDA is introduced by Griffiths and Steyvers in [6]. We can sample the topic assignment for each word in the corpus as follows:

$$P(T_{di} = t|\mathbf{t}_{-w_{di}}, \mathbf{w}) \propto P(T_{di} = t, \mathbf{t}_{-w_{di}}, \mathbf{w}) \tag{2.34}$$

$$\propto P(w_{di}|\mathbf{t}, \mathbf{w}_{-di}) P(T_{di} = t|\mathbf{t}_{-w_{di}}) \tag{2.35}$$

$$= \frac{n_{-w_{di},t}^{(w_{di})} + \beta}{n_{-w_{di},t}^{(\cdot)} + |\mathcal{W}| * \beta} \times \frac{n_{-w_{di},t}^{(d)} + \alpha}{n_{-w_{di},\cdot}^{(d)} + |\mathcal{T}| * \alpha}, \tag{2.36}$$

where $\mathbf{t}_{-w_{di}}$ is the topic assignments for word $w_{di}$ except the current word,$n_{-w_{di},t}^{(w_{di})}$ is the number of times that the word at location $i$ in document $d$ is assigned with topic $t$ except the current word. Similarly, $n_{-w_{di},t}^{(\cdot)}$ is the total number of words assigned to topic $t$ except

the current word. Also, $n^{(d)}_{-w_{di},t}$ is the number of occurrences of word $w_{di}$ that are assigned to topic $t$ in document $d$. Similarly, $n^{(d)}_{-w_{di},\cdot}$ is the number of words in document $d$ except the current word. At last, the term $\frac{n^{(w_{di})}_{-w_{di},t}+\beta}{n^{(\cdot)}_{-w_{di},t}+|\mathcal{W}|*\beta}$ can be thought as $\hat{\phi}^{w_{di}}_t$ which is the probability of the word $w_{di}$ under topic $t$. The term $\frac{n^{(d)}_{-w_{di},t}+\alpha}{n^{(d)}_{-w_{di},\cdot}+|\mathcal{T}|*\alpha}$ can also be thought as $\hat{\theta}^d_t$ which is the probability of the topic $t$ in document $d$. Moreover, hyperparameters $\beta$ and $\alpha$ control the prior of $\phi$ and $\theta$. The pseudo code for LDA is shown in Algorithm 1. Here, $N_d$ is the total number of words in the document $d$ and $ITERR$ is the number of iteration for Gibbs Sampling.

**Algorithm 1** LDA

**Function:** LDA

**Inputs:** $\mathcal{D}$, $\mathcal{T}$ and $ITER$

**Output:** $\theta$, $\phi$

Initialize $T_{di}$ for each word $w_{di}$ randomly from $\mathcal{T}$

iteration = 0;

**while** iteration $< ITER$ **do**

  **for** each $d \in \mathcal{D}$ and $i \in 1, ..., N_d$ **do**

    sample a new topic $T_{di}$ from

$$\Pr(T_{di} = t|\mathbf{t}_{-di}, \mathbf{w}) \propto \frac{n^{(w_{di})}_{-w_{di},t} + \beta}{n^{(\cdot)}_{-w_{di},t} + |\mathcal{W}| * \beta} \times \frac{n^{(d)}_{-w_{di},t} + \alpha}{n^{(d)}_{-w_{di},\cdot} + |\mathcal{T}| * \alpha}$$

  **end for**

  iteration++;

**end while**

Compute topic-word distribution $\phi = \frac{n^{(w_{di})}_{-w_{di},t}+\beta}{n^{(\cdot)}_{-w_{di},t}+|\mathcal{W}|*\beta}$

Compute document-topic distribution $\theta = \frac{n^{(d)}_{-w_{di},t}+\alpha}{n^{(d)}_{-w_{di},\cdot}+|\mathcal{T}|*\alpha}$

Because LDA assumes a finite number of hidden components, this algorithm needs to know the number of topics $\mathcal{K}$ before running.

### 2.5.2 Online LDA

In some situations it is desirable to incrementally update the model as new data (i.e. documents) become available. Ideally, we'd like this update to be fast and perhaps with time that does not depend on the amount of data that we already have.

While there are no known online algorithm that can do an update in constant time without impacting the quality of the update, we review three algorithms that provide different tradeoffs between runtime and accuracy. These algorithms are variants of Gibbs sampling and try to reduce the number of topic assignments in the previous data that need to be re-sampled at each update. More precisely, they perform regular Gibbs sampling (for LDA) on the current corpus, then sample the topic assignments of the new data given the corpus and re-sample a small set of topic assignments in the corpus.

The first algorithm is called "o-LDA" and is described in Song et al. [15]. This algorithm applies regular LDA to an initial topic assignment to the data set, then samples each new word by conditioning on the previous results. Suppose that the set of all documents received so far is called $\mathcal{A}$ and some new documents arrive that we refer as $\mathcal{B}$. First, we run LDA on $\mathcal{A}$ according to $Algorithm1$. Then we estimate a topic-word distribution $\phi_{\mathcal{A}}$ as follows:

$$\phi_{\mathcal{A}} = \frac{n_{-w_{di},t}^{(w_{di})} + \beta}{n_{-w_{di},t}^{(\cdot)} + |\mathcal{W}| * \beta}. \tag{2.37}$$

Second, sample a topic assignment for each word in $\mathcal{B}$ that takes into account the topic-word distribution $\phi_{\mathcal{A}}$. The equation is:

$$P(T_{di} = t|\mathbf{T}_{<di}^{A \bigcup B}, \mathbf{W}^{A \bigcup B}) \propto \frac{\hat{n}_{w_{di},t}^{(w_{di})} + \beta}{\hat{n}_{w_{di},t}^{(\cdot)} + |\mathcal{W}| * \beta} \times \frac{n(d)_{-w_{di},t} + \alpha}{n(d)_{-w_{di},\cdot} + |\mathcal{T}| * \alpha} \tag{2.38}$$

$$\propto \phi_{\mathcal{A}} \times \frac{n(d)_{-w_{di},t} + \alpha}{n(d)_{-w_{di},\cdot} + |\mathcal{T}| * \alpha} \tag{2.39}$$

where $\hat{n}$ comes from corpus $\mathcal{A}$. The algorithm "o-LDA" is summarized in Algorithm 2:

**Algorithm 2** o-LDA

**Inputs:** $\mathcal{A}$,$\mathcal{B}$, $\mathcal{T}$

**Output:** $\theta$, $\phi$

Perform regular LDA with input parameters: $\mathcal{A}$ and $\mathcal{T}$.

Estimate $\phi_{\mathcal{A}} = \frac{n_{-w_{di},t}^{(w_{di})}+\beta}{n_{-w_{di},t}^{(\cdot)}+|\mathcal{W}|*\beta}$ from data set $\mathcal{A}$.

**for** each $d \in \mathcal{B}$ and each $i \in \{1, ..., N_d\}$ **do**

sample a new topic $T_{di}$ from $\Pr(T_{di} = t|\mathbf{t}_{<di}^{\mathcal{A}\bigcup\mathcal{B}}, \mathbf{w}^{\mathcal{A}\bigcup\mathcal{B}}) \propto \phi_{\mathcal{A}} \times \frac{n_{-w_{di},t}^{(d)}+\alpha}{n_{-w_{di},\cdot}^{(d)}+|\mathcal{T}|*\alpha}$.

**end for**

Compute topic-word distribution $\phi = \frac{n_{-w_{di},t}^{(w_{di})}+\beta}{n_{-w_{di},t}^{(\cdot)}+|\mathcal{W}|*\beta}$

Compute document-topic distribution $\theta = \frac{n_{-w_{di},t}^{(d)}+\alpha}{n_{-w_{di},\cdot}^{(d)}+|\mathcal{T}|*\alpha}$

This algorithm's accuracy depends on the accuracy of regular LDA on dataset $\mathcal{A}$. If the latent structure of the documents in dataset $\mathcal{B}$ is different from the latent structure of the document in dataset $\mathcal{A}$, the result won't be good.

Another online algorithm is called "Incremental Gibbs Sampler" which is an instance of the delayed MCMC framework [12]. It is an extension of "o-LDA". There is an extra step after sampling the topic of each word $w_{di}$ in $\mathcal{B}$. The algorithm re-samples the topic assignments of some words that occur in $\mathcal{A}$ or in $\mathcal{B}$ before the word $w_{di}$. This re-sampling tries to adjust the topic assignments to take into account all of $\mathcal{A}\bigcup\mathcal{B}$ instead of only the words in $\mathcal{A}$. To do this we set a "Rejuvenation sequence", $\mathcal{R}(i)$, which is a sequence of words related to $w_i$. After we sample the topic for the word $w_i$, we need to sample each word $w_j$ where $j \in \mathcal{R}(i)$ according to $P(T_j = t|\mathbf{t}_{i\backslash j}, \mathbf{w}_i)$. The pseudo code of this incremental Gibbs sampler is provided in Algorithm 3:

**Algorithm 3** incremental-LDA

**Inputs:** $\mathcal{A}$,$\mathcal{B}$, and $\mathcal{T}$

**Output:** $\theta$, and $\phi$

Perform regular LDA with input parameters: $\mathcal{A}$, $\mathcal{T}$.

Estimate $\phi_{\mathcal{A}}$

**for** each $d \in \mathcal{B}$ and each $i \in \{1, ..., N_d\}$ **do**

sample a new topic $T_{di}$ from $\Pr(T_{di} = t|\mathbf{t}_{<di}^{\mathcal{A}\bigcup\mathcal{B}}, \mathbf{w}^{\mathcal{A}\bigcup\mathcal{B}}) \propto \phi_{\mathcal{A}} \times \frac{n_{-w_{di},t}^{(d)}+\alpha}{n_{-w_{di},\cdot}^{(d)}+|\mathcal{T}|*\alpha}$.

**for** $j \in \mathcal{R}(i)$ **do**

Sample $Pr(T_j = t|\mathbf{t}_{i\setminus j}, \mathbf{w}_i)$

**end for**

**end for**

Compute topic-word distribution $\phi = \frac{n_{-w_{di},t}^{(w_{di})}+\beta}{n_{-w_{di},t}^{(\cdot)}+|\mathcal{W}|*\beta}$

Compute document-topic distribution $\theta = \frac{n_{-w_{di},t}^{(d)}+\alpha}{n_{-w_{di},\cdot}^{(d)}+|\mathcal{T}|*\alpha}$

In this algorithm, if enough resampling steps are performed, we can approximate the posterior distribution $P(t_{di}|w_{di})$ for each word in the corpus availably closely. If we do not re-sample enough then the accuracy may suffer.

The author of [2] designed a more precise algorithm, which uses Particle filtering (a.k.a. sequential Monte Carlo). We will call this algorithm p-LDA. Most of the time, Particle filtering is used to approximate a distribution conditioned on a sequence of observations. In particular, it can be used to estimate the distribution $P(t_{0:i}|w_{0:i})$, where $t_{0:i}$ is the hidden state sequence corresponding to the topic assignment up to word $i$, and $w_{0:i}$ is the sequence of observations from word 0 to word $i$. Let $p$ denote a particle, the author represents $P(t_{0:i}|w_{0:i})$ by a collection of $N$ weighted samples (a.k.a. particles), $t_{0:i}^{(p)}, {c_i^{(p)}}_{p=1}^{N}$, where the $c_i^{(p)}$ is the weight of $t_{1:i}^{(p)}$. A particle representation of this density is:

$$P(t_{0:i}|w_{0:i}) \approx \sum_p c_{0:i-1}^{(p)} \delta(t_{0:i} - t_{0:i-1}^{(p)}). \tag{2.40}$$

The integral is:

$$P(t_{0:i}|w_{0:i}) = \alpha P(w_i|t_i) \int P(t_{0:i-1}|w_{0:i-1}) P(t_i|t_{0:i-1}, w_{0:i}) dt_{0:i-1} \tag{2.41}$$

which approximates to

$$P(t_{0:i}|w_{0:i}) \approx \alpha P(w_i|t_i) \sum_p c_{i-1}^{(p)} P(t_i|t_{0:i-1}^{(p)}, w_{0:i}). \tag{2.42}$$

In p-LDA model, we need to compute $P(t_{0:i}|w_{0:i})$ recursively from $P(t_{0:i-1}|w_{0:i-1})$ after word $w_i$ is observed. Suppose one samples $t_i$ from $P(t_i^{(p)}|t_{0:i-1}^{(p)}, w_{0:i})$, which we denoted by $Q(t_i^{(p)}|t_{0:i-1}^{(p)}, w_{0:i})$ to emphasize that this is the proposal distribution [3]. We obtain the important weight as follows:

$$c_i^{(p)} = \frac{P(t_{0:i}^{(p)}|w_{1:i})}{Q(t_i^{(p)}|t_{0:i-1}^{(p)}, w_{1:i}) \times P(t_{0:i-1}^{(p)}|w_{1:i-1})}. \tag{2.43}$$

We can compute the importance weight $c_i^{(p)}$ recursively from $c_{i-1}^{(p)}$:

$$\frac{c_i^{(p)}}{c_{i-1}^{(p)}} \propto \frac{P(w_i|\mathbf{t}_i^p, \mathbf{w}_{i-1}) P(t_i^{(p)}|\mathbf{t}_{i-1}^{(p)})}{Q(t_i^{(p)}|\mathbf{t}_{i-1}^{(p)}, \mathbf{w}_i)} \tag{2.44}$$

$$= P(w_i|\mathbf{t}_{i-1}^{(p)}, \mathbf{w}_{i-1}), \tag{2.45}$$

Once the weights are normalized to one. the particle filter approximates the posterior distribution over topic assignments according to the following equation:

$$P(\mathbf{t}_i|\mathbf{w}_i) = \sum_{p=1}^{P} c_i^{(p)} \delta_{t_i}(t_i^{(p)}), \tag{2.46}$$

where $p \in \mathcal{P}$, and $\delta_{t_i}(\cdot)$ is the indicator function for $t_i$. Then, we could say:

$$\delta_{\mathbf{t}_i}(\mathbf{t}_i^{(p)}) = \begin{cases} 1, & \text{if } t_i^{(p)} \in \mathbf{t}_i; \\ 0, & \text{otherwise.} \end{cases} \tag{2.47}$$

The pseudocode of the Particle Filtering-LDA algorithm is described in Algorithm 4:

**Algorithm 4** Particle Filtering-LDA

**Inputs:** $\mathcal{A}$,$\mathcal{B}$, $\mathcal{T}$

**Output:** $\theta$, $\phi$

Initialize weights $c_0^{(p)} = \frac{1}{|P|}$ for $p = 1, ..., P$

**for** each $d \in \mathcal{A} \bigcup \mathcal{B}$ and $i \in \{1, ..., N_d\}$ **do**

  **for** $p = 1, ..., |P|$ **do**

    set $c_i^{(p)} = c_{i-1}^{(p)} P(w_i | \mathbf{t}_{i-1}^{(p)}, \mathbf{w}_{i-1})$

    sample $t_i^{(p)}$ from $Q(t_i^{(p)} | \mathbf{t}_{i-1}^{(p)}, \mathbf{w}_i)$

  **end for**

  normalize weights $\mathbf{c}_i$ to sum to 1.

  **if** $\|c_i\|^{-2} \leq ESS$ threshold **then**

    resample particles

    **for** $j \in \mathcal{R}(i)$ **do**

      **for** $p = 1, ..., P$ **do**

        sample $t_j^{(p)}$ from $\Pr(t_j^{(p)} | \mathbf{t}_{i \setminus j}^{(p)}, \mathbf{w}_i)$

      **end for**

    **end for**

    set $c_i^{(p)} = |P|^{-1} (p = 1, ..., P)$.

  **end if**

**end for**

Compute topic-word distribution $\phi = \frac{n_{-w_{di},t}^{(w_{di})} + \beta}{n_{-w_{di},t}^{(\cdot)} + |\mathcal{W}| * \beta}$

Compute document-topic distribution $\theta = \frac{n_{-w_{di},t}^{(d)} + \alpha}{n_{-w_{di},\cdot}^{(d)} + |\mathcal{T}| * \alpha}$

Here, the $ESS$ is the effective sample size, which is a constant value, and the definition of $R(i)$, "Rejuvenation Sequence", is the same as for the previous algorithm. In this algorithm, how we set $|P|$ and $ESS$ will directly affect the accuracy for this model. If the number of topics is large, we need a large number of particles for sampling. Also, if $ESS$ is small then some particles may dominate and reduce the diversity in the estimate of the topic assignment.

### 2.5.3 Hierarchical Dirichlet Processes (HDP)

LDA is very efficient when we know the number of latent topics. However, most of the time, the number of topics for a corpus is unknown. In that case, a non-parametric model is a good choice since the number of parameters (such as the number of topics) is not set a priori, but learned from data. This model can adapt the number of topics based on the data.

Let's first review the Dirichlet process [4]. A Dirichlet process is a prior used in Bayesian modeling of data. It is a distribution over infinite multinomials. A Dirichlet process is also known as an infinite mixture model, which has Dirichlet distributed finite dimensional marginal distributions. Distributions drawn from a Dirichlet process are discrete, however, they cannot be described by a finite number of parameters. Thus, the Dirichlet Process is a non-parametric model. For example, if a random variable $G$ is distributed according to a DP, its marginal distributions are Dirichlet distributed. Let $\theta$ be a continuous space and $A_1, ..., A_r$ be a partition of $\theta$. Then for every finite measurable partition $A_1, ..., A_r$ of $\theta$ the vector $\langle G(A_1), ..., G(A_r)\rangle$ is random since $G$ is random. In particular, when $G$ is distributed according to a Dirichlet process with base distribution $H$ and concentration parameter $\alpha$, written as $G \sim DP(\alpha, H)$ then:

$$(G(A_1), ..., G(A_r)) \sim Dir(\alpha H(A_1), ..., \alpha H(A_r) \tag{2.48}$$

for every finite measurable partition $A_1, ..., A_r$ of $\theta$.

Back to HDP which was described by Teh et al. [16], it is a model derived from LDA, but the number of topics is not assumed to be given. Instead, it is learned from the data. Since the number of topics is unknown a priori and could be arbitrarily large, one could replace the finite topic mixture of each document by an infinite topic mixture $\theta_d = \langle \theta_{d1}, ..., \theta_{d\infty}\rangle$ with a Dirichlet process as a prior. While this is fine in theory, the documents would almost certainly not share any topic in practice. This is an artefact of the way Dirichlet processes are defined. Consider a symmetric Dirichlet process with concentration $\alpha$ and a uniform mean measure. Since there are infinitely many possible topics, the probability of any topic under a uniform measure is 0. Hence, when a new topic is sampled for a document, it will almost certainly be different from all the topics

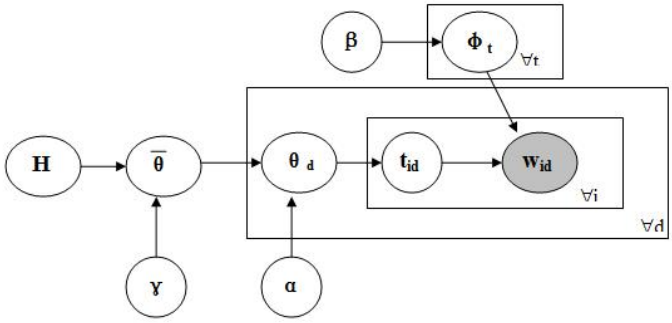


Figure 2.3: plate notation of HDP

sampled in other documents. An elegant solution to this problem consists of using a two-level hierarchical Dirichlet process (see Figure 2.3). More precisely, the mean measure of the Dirichlet process over each document's topic mixture is replaced by a corpus level topic mixture $\bar{\theta}$, which is itself distributed according to a Dirichlet process (with some mean $H$ often chosen to be uniform). This construction ensures that the topic mixtures of all documents share the same topics defined at the corpus level. Even though there are infinitely many topics in the corpus-level mixture, the probability of each topic is non-zero. Hence, when each document samples a new topic, the probability that it is the same as some other document's topic is non-zero.

While it is possible to define a generative process that follows the graphical model of the HDP topic model in Figure 2.3, in practice it is difficult to sample infinite topic mixtures. An alternative generative process samples the words of each document in sequence by following an urn model. To distinguish between the topic mixtures at the document and corpus levels, we will refer to group mixtures at the document level and topic mixture at the corpus level. Note however that each group will be associated with a topic and therefore a group can be thought as identifying a topic. We denote by $w_{di}$ and $g_{di}$ the word and the group of the $i$th term in document $d$, and by $t_g$ the topic of group $g$. First, we choose an existing group $g$ with probability $\frac{n_{dg}^{<i}}{i-1+\alpha}$ or create a new group with probability $\frac{\alpha}{i-1+\alpha}$. Here, $n_{dg}^{<i}$ is the number of terms in group $g$ of document $d$ that precede the $i$th

term. When a new group is created, an existing topic t is selected with probability $\frac{m_{\cdot t}}{m_{\cdot\cdot}+\gamma}$ or a new topic is created with probability $\frac{\gamma}{m_{\cdot\cdot}+\gamma}$. Here, $m_{\cdot t}$ is the number of groups assigned to topic $t$ in the corpus so far and $m_{\cdot\cdot}$ is the total number of groups in the corpus so far. When a new topic is created, a distribution over words $\phi_t$ is sampled from a Dirichlet prior with hyperparameter $\beta$. Finally the $i$th word of document $d$ is sampled from $\phi_{w_{di},t}$. This process is repeated for the next word of document $d$ and so on.

Similar to LDA, Gibbs sampling can be used to infer a likely topic assignment. A simple way of doing Gibbs sampling is by sampling the group and topic of each word. The algorithm is as follows:

**Algorithm 5** HDP

**Inputs:** $\mathcal{D}$, $\mathcal{T}$ and $ITER$

**Output:** $\theta$, and $\phi$

Initialize $T_{di}$ for each $w_{di}$

iteration =0;

**while** iteration $< ITER$ **do**

**for** each $d \in \mathcal{D}$ and $i \in \{1, ..., N_d\}$ **do**

Sample topic from $\Pr(T_{di} = t|\mathbf{t}_{-di}, \bar{\theta}) = \begin{cases} (n_{-w_{di},t}^{(w_{di})} + \alpha\bar{\theta}_t) f_t^{-w_{di}}(w_{di}), & \text{if } t \text{ already exists;} \\ \alpha\bar{\theta}_u f_t^{-w_{di}}(w_{di}), & \text{Otherwise.} \end{cases}$

Sample $m_{dt}$ from $\Pr(M_{dt} = m|\mathbf{t}, \mathbf{m}_{-dt}, \bar{\theta}) \propto \frac{\Gamma(\alpha\bar{\theta}_t)}{\Gamma(\alpha\bar{\theta}_t + n_{-w_{di},t}^{(w_{di})})} s(n_{-w_{di},t}^{(w_{di})}, m)(\alpha\theta_t)^m$.

Sample $\bar{\theta}$ from $(\bar{\theta}_1, ..., \bar{\theta}_k, \bar{\theta}_u) \sim Dir(m_{\cdot 1}, ..., m_{\cdot k}, \gamma)$

**end for**

iteration++;

**end while**

Compute topic-word distribution $\phi = \frac{n_{-w_{di},t}^{(w_{di})} + \beta}{n_{-w_{di},t}^{(\cdot)} + |\mathcal{W}| * \beta}$

Compute document-topic distribution $\theta = \frac{n_{-w_{di},t}^{(d)} + \alpha}{n_{-w_{di},\cdot}^{(d)} + |\mathcal{T}| * \alpha}$

Here, $\bar{\theta}_u$ is the probability of the new topic, $f_t^{-w_{di}}(w_{di}) = \frac{n_{-w_{di},t}^{(w_{di})} + \beta}{n_{-w_{di},t}^{(\cdot)} + |\mathcal{W}|\beta}$ where the $n_{-w_{di},t}^{(w_{di})}$ is the number of terms with word $w_{di}$ assigned to topic $t$ except the $i$th term of document $d$ and $n_{-w_{di},t}^{(\cdot)}$ is the total number of terms assigned to topic $t$ except the $i$th term in document

$d$, and the $s(n, m)$ functions are Stirling numbers. More precisely, the Stirling numbers are defined as follows::

$$\begin{cases} s(0,0) = s(1,1) = 1, & \\ s(n,0) = 0, & \text{if } n > 0; \\ s(n,m) = 0, & \text{if } m > n; \\ s(n+1,m) = s(n,m-1) + ns(n,m), & \text{otherwise.} \end{cases} \tag{2.49}$$

The main difference between HDP and LDA is that LDA assumes a fixed number of topics whereas HDP learns the number of topics from data. HDP's ability to adjust the number of topics will be particularly useful in the next chapter when we develop an approach for transfer learning between a side corpus and a small cluster. When the cluster is very different from the corpus, it will make sense to create additional topics to capture the content of the cluster.

## 2.6 Beyond Topic Model

There are language models that combine the latent topic and some previous words from an n-gram model to enhance the performance [18]. We review a model that gives the same importance to the latent topic assignment and the previous word to predict the next word. Given a text corpus $\mathcal{D}$ with document $d$, and a word sequence of $N$ tokens in each document, let $i$ be the location of word $w_i$ in document $d$, and let word $w_{i-1}$ be the word preceding $w_i$. The range of $i$ is from 1 to $N-1$. The first method introduced by Hanna M. Wallach [18] is to use the same hyperparameters $\alpha$ and $\beta$ are as for LDA, and both of them are vectors. The second method introduced in the same paper is to set $\beta$ differently for each topic given the current word $w_i$. The graphical model for the first method is shown in Figure 2.4. Here, the word $w_{di}$ is not only related to its topic assignment $t_{di}$, but also related to its previous word $w_{d(i-1)}$. Moreover, the topic-word distribution is changed from $\phi_t$ to $\phi_{wt}$ which is a matrix with $|\mathcal{W}||\mathcal{T}|(|\mathcal{W}|-1)$ free parameters. This matrix has $|\mathcal{W}||\mathcal{T}|$ rows for the previous word and topic, and $(|\mathcal{W}|-1)$ columns for the current word. The document-topic distribution $\theta$ is the same as for regular LDA, which has $|\mathcal{D}||\mathcal{T}|$ free parameters. The generative flow for a text corpus $\mathcal{D}$ is:

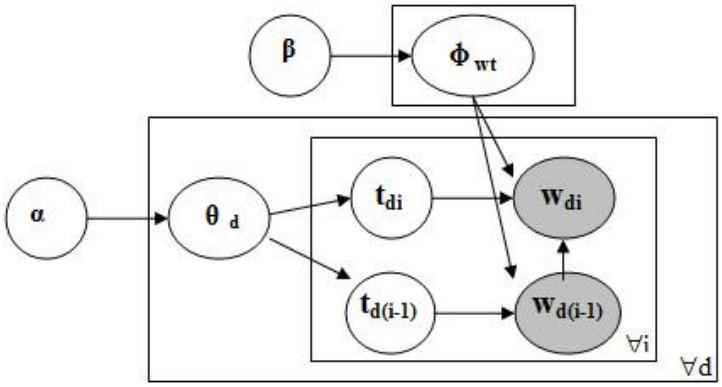


Figure 2.4: plate notation of Beyond Topic Model

- Sample $\phi_{wt}$ from the symmetric Dirichlet prior (with hyperparameter $\beta$)
- Sample $\theta_d$ from the symmetric Dirichlet prior (with hyperparameter $\alpha$)
- For each word $w_{di}$
    1. Draw a topic assignment $t_{di}$ from Multinomial $\theta_d$
    2. Draw a word $w_{di}$ from $P(w_{di}|t_{di}, \phi_{wt}, w_{d(i-1)})$

An Expectation Maximization ($EM$) algorithm is used in this paper. Nowadays, $EM$ is a popular algorithm in statistical estimation problems involving incomplete data. Each iteration of the $EM$ algorithm consists of two steps:

- E-step (Expectation): the missing data is estimated given observed data and the current estimate of the parameters.
- M-step (Maximization): the likelihood function is maximized under the assumption that the missing data is known.

In this model, by method one, given text corpus $\mathcal{D}$, the $EM$ algorithm is executed as follows:

**Algorithm 6** BI-GRAM LDA

**Inputs:** $\mathcal{D}$, $\mathcal{T}$, and *iteration*

**Output:** $\theta$, and $\phi$

Initialize topic assignment $\mathbf{z}$, $U = (\alpha, \beta)$ and $r = 0$. Here, variable $r$ is used for the iteration.

**for** $r = 1$ to *iteration* **do**

E-step: Draw $|\mathcal{S}|$ samples $z^{(s)}{}_{s=1}^{|\mathcal{S}|}$ from $P(\mathbf{z}|\mathbf{w}, U^{(r-1)})$.

M-step: $U^{(r)} = \arg\max_{\mathbf{U}} \frac{1}{|\mathcal{S}|} \sum_{s=1}^{|\mathcal{S}|} logP(\mathbf{w}, \mathbf{z}^s|U)$.

**end for**

Compute topic-word distribution $\phi = \frac{n^{(w_{di})}_{-w_{di},t} + \beta}{n^{(\cdot)}_{-w_{di},t} + |\mathcal{W}| * \beta}$

Compute document-topic distribution $\theta = \frac{n^{(d)}_{-w_{di},t} + \alpha}{n^{(d)}_{-w_{di},\cdot} + |\mathcal{T}| * \alpha}$

This model can obtain lower perplexity than a regular topic model because it uses bi-grams instead of uni-grams. More precisely, the frequency of the occurrence of a pair of words is lower than the occurrence of a single word. The results show that some common words, such as "the" and "a", do not exist in the key word list anymore since their generative probability is lower due to the fact that they have a lot of different previous words.

# Chapter 3

# Language Modeling for Transfer Learning

## 3.1 Transfer Learning

Recently, numerous researchers have focused on Transfer Learning, which studies how previous knowledge can be generalized and reused in different, but similar contexts. Transfer Learning is widely used in mental base activities, such as spelling and movement assistance. In this chapter, we use some current generative models, such as LDA, and HDP, to create some new Transfer Learning models for our cluster labeling task.

Suppose we have a large side corpus $\mathcal{A}$ and a small cluster $\mathcal{B}$ that we would like to label. The simplest approach is to ignore $\mathcal{A}$ and to build a language model for $\mathcal{B}$ (with LDA or HDP), from which we can extract keywords with the highest generation probability. However, since $\mathcal{B}$ is small, the latent structure of $\mathcal{B}$ may not be correct because the model can be easily affected by abnormal occurrences of some words. Another way is to build our language model based on $\mathcal{A} \bigcup \mathcal{B}$. But there are two possible pitfalls:

- Since we do not know the relationship between $\mathcal{A}$ and $\mathcal{B}$, if $\mathcal{A}$ is quite different from $\mathcal{B}$, then there may be some negative learning. In other words, the label may be good for $\mathcal{A}$ but not for $\mathcal{B}$ since $\mathcal{A}$ is much bigger than $\mathcal{B}$ .

- The efficiency is another issue. When training with $\mathcal{A}$ (which should be as large as possible), Gibbs sampling for either LDA or HDP may take hours or days to converge. Many web pages are updated on a daily or even hourly bases, so it is too expensive to label the large set of rapidly changing web pages.

Ideally, we would like to learn a generic language model from $\mathcal{A}$ only once, which is similar to a child learning his (or her) native language. Assuming $\mathcal{A}$ is large, this may take a long time. This procedure is similar to a child going through several years of education to grasp grammar and meaning of each word in his (or her) vocabulary. When children get a cluster of documents to label, they would naturally use their previous knowledge to understand the words they have already seen, and then synthesize the topic of the cluster. We use a similar idea to build our Transfer Learning models in this chapter. We import (or transfer) some information, as prior knowledge, from $\mathcal{A}'$s language model to quickly build a new tailored language model each time a new cluster must be labeled. The importance of the language model of $\mathcal{A}$ to improve the language model of $\mathcal{B}$ depends on how close $\mathcal{A}$ is to $\mathcal{B}$.

As explained in chapter two, there are three outputs generated after a topic model is built from a data set. They are the document-topic distribution $\theta$, the topic-word distribution $\phi$, and the topic assignment for each word at the last sampling step. Then, we need to decide which output is good to transfer knowledge to the new cluster. $\theta$ is the topic distribution for each document. Since we don't know the new cluster's document structure, we cannot pass $\theta$ for transfer learning. If we pass $\phi$ to the new cluster, then it seems like we assume that the topic-word distributions are the same in $\mathcal{A}$ and $\mathcal{B}$, and this will correspond to o-LDA in chapter two. So, we pass the words' topic assignments to the new cluster as a previous sample result that is used to set the hyperparameters of a Dirichlet prior over the topic-word distribution. This allows us to bias the language model of $\mathcal{B}$ without necessarily making it equal to $\mathcal{A}$'s language model. Let's use $n_{wt}$ to denote the frequency with which word $w$ is assigned to topic $t$. To compensate for the differences between $\mathcal{A}$ and $\mathcal{B}$, we introduce another parameter $c \in [0, 1]$ as the weight of $\mathcal{A}$ in the prior of $\mathcal{B}$ (i.e., the frequencies $n_{wt}$ are multiplied by $c$ in the hyperparameters of the Dirichlet priors). When $c = 0$, this means that we assume $\mathcal{A}$ and $\mathcal{B}$ are totally unrelated and so

ignore $\mathcal{A}$. The result is same as apply LDA on $\mathcal{B}$ only. When $c = 1$ then $\mathcal{A}$ and $\mathcal{B}$ are from the same corpus and contain documents with roughly the same statistics. The result of this case is same as apply LDA on $\mathcal{A} \bigcup \mathcal{B}$. So, as we could obtain the weight $c$ between these two values, we could let data to claim how many knowledge, which the model learned from $\mathcal{A}$, need to be transfer to the language model on $\mathcal{B}$. Since the perplexity curve with respect to $c$ is generally convex, it is possible to use a binary search to find the best weight faster.

In this chapter, we will show how to find the best weight $c$ using binary search algorithm, and how to use LDA, and HDP combined with the weight to transfer knowledge from one language model to another.

## 3.2 Corpus Perplexity

The metric used to evaluate the fitness of language models is the standard *perplexity* from the information retrieval literature. Perplexity is a measure of how "surprised" the model would be to see a sequence of words, which is related to the probability that the model would produce a sequence of words [14]. For example, suppose we have a test set $\mathcal{D}_{test}$ of documents, let us use $\mathbf{w}_d$ to denote the sequence of words $\mathbf{w}$ in each document $d$ in $\mathcal{D}_{test}$ and $\tilde{\mathbf{w}}$ denotes the entire word sequence in $\mathcal{D}_{test}$ ($\tilde{\mathbf{w}} = \mathbf{w}_1 + ... + \mathbf{w}_{|\mathcal{D}_{test}|}$). Also, let's denote our model by $\mathcal{M}$. The perplexity of $\tilde{\mathbf{w}}$ is:

$$perplexity(\tilde{\mathbf{w}}|\mathcal{M}) = \prod_{d \in \mathcal{D}_{test}} P(\mathbf{w}_d|\mathcal{M})^{-\frac{1}{n_d}} \tag{3.1}$$

$$= exp(-\frac{\sum_{d \in \mathcal{D}_{test}} logP(\mathbf{w}_d|\mathcal{M})}{\sum_{d \in \mathcal{D}_{test}} n_d}) \tag{3.2}$$

where $n_d$ is the number of words in document $d$. Since $P(\mathbf{w}_d|M)$ is the conditional probability of the word sequence given the language model $\mathcal{M}$ , we can compute it as follows:

$$P(\mathbf{w}_d|\mathcal{M}) = \prod_{n=1}^{n_d} \sum_{k=1}^{K} P(w_n = w|t_n = k) \cdot P(t_n = k|d = d) \tag{3.3}$$

$$= \prod_{i=1}^{V} [\sum_{k=1}^{K} \phi_{kw_i} \cdot \theta_{dk}]^{n_d^{w_i}} \tag{3.4}$$

Here, $V$ is the dictionary size of the test data set, and $w_i$ is the $i$th word in the dictionary, where $i$ is from 1 to $V$. In contrast, $w_n$ is the word in the $n$th location in a document, $t_n$ is the topic assignment for word $w_n$, and $K$ is the total number of topics. Taking the logarithm, we obtain:

$$logP(\mathbf{w}_d|\mathcal{M}) = \sum_{i=1}^{V} n_d^{w_i} log(\sum_{k=1}^{K} \phi_{kw_i} \cdot \theta_{dk}). \tag{3.5}$$

The above equation shows that the perplexity is the geometric average of the reciprocal probability over all words in the test data set. So, we can see that without the constant factor $(-\frac{1}{\sum_{d\in\mathcal{D}_{test}} n_d})$, the term $\sum_{d\in\mathcal{D}_{test}} logP(\mathbf{w}_d|\mathcal{M})$ is the average conditional log probability or log likelihood of the test corpus. So, a lower value of perplexity implies a high data likelihood for text analysis. We use perplexity as the main measure to compare transfer learning models in this thesis.

## 3.3 LDA Transfer Model

We introduce how to use LDA to build the LDA-Transfer model (LDA-Trans) in this section. For cross validation, we separate the new cluster in two parts. 90% of the documents are chosen at random and denoted by $\mathcal{B}$. The remaining 10% is denoted by $\mathcal{C}$. In this way, $\mathcal{B}$ is used for learning and $\mathcal{C}$ is used for testing. To generate the Transfer Learning model based on $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$, we propose the following approach. Suppose that we have already built a language model with $|\mathcal{T}|$ topics for $\mathcal{A}$ and stored the frequencies $n_{w,t}$ (number of terms with word $w$ assigned to topic $t$). However, we need to weigh the frequencies of $\mathcal{A}$ with the weight $c$, which indicates implicitly our belief of similarity between $\mathcal{A}$ and $\mathcal{B}$.

At the first step, we apply batch LDA on the side corpus $\mathcal{A}$, then, we store the topic assignment for each word in a matrix $\mathcal{F}_\mathcal{A}$ where each element is $n_{wt}$. Next, we obtain a weight $c$ based on the perplexity. Although we don't know how similar the side corpus $\mathcal{A}$ and the new cluster $\mathcal{B}\bigcup\mathcal{C}$ are, we know that if $c = 0$ then our transfer learning model will do the same thing as applying batch LDA on the cluster. In other words, there is no knowledge passed to the cluster to improve the accuracy. If $c = 1$, then the topic

assignments will be passed to the cluster under the assumption that $\mathcal{A}$ and $\mathcal{B}$ have the same latent structure. We do a search for the weight $c$ between 0 and 1. Since we observed that the perplexity curves generated by our transfer learning models for the new cluster are convex with respect to $c$, a binary search described in Algorithm 7 can quickly find the weight $c$ with the lowest perplexity:

**Algorithm 7** FindWeight

**Inputs:** $ACC$

**Output:** weight $c$

Set $w_L = 0$, $w_M = 0.5$, $w_R = 1$

Evaluate $p_L = TransModel(w_L)$; $p_M = TransModel(w_M)$, $p_R = TransModel(w_R)$;

**repeat**

Set $w_{LM} = (w_L + w_M)/2$; Evaluate $p_{LM} = TransModel(w_{LM})$;

Set $w_{MR} = (w_M + w_R)/2$; Evaluate $p_{MR} = TransModel(w_{MR})$;

Set $x^* = argmin_{x \in \{L,LM,M,MR,R\}} p_x$;

Set $c = w_{x^*}$;

**if** $c == w_L$ **then**

$w_M = w_{LM}; w_R = w_M$;

$p_M = p_{LM}; p_R = p_M$;

**else if** $c == w_R$ **then**

$w_M = w_{MR}; w_L = w_M$;

$p_M = p_{MR}; p_L = p_M$;

**else**

$w_M = c$; $w_L = LeftNeighbor(w_M)$; $w_R = RightNeighbor(w_M)$;

$p_M = p_{x^*}$; $p_L = p_{w_L}$; $p_R = p_{w_R}$;

**end if**

**until** $|w_L - w_M| < ACC$ or $|w_R - w_M| < ACC$

**return** $c$

Here, the function $TransModel(w)$ is used to call LDA-Trans model or HDP-Trans model with the weight $w$ and return the corresponding perplexity value. The function $LeftNeighbor(w)$ is used to return the weight on the left hand side of $w$ and $RightNeighbor(w)$ returns the right hand side of $w$.

For the LDA-Trans model, we initialize the prior distribution over topic assignments in $\mathcal{B}$ with the posterior distribution (based on $\mathcal{A}$) weighted by $c$. In other words, we set the hyperparameters of the prior to $c$ times $n^A_{w_{di},t}$. We use Gibbs sampling to repeatedly re-sample each $T_{di}$ as follows:

$$P(T_{di} = t|\mathbf{t}_{-di}, \mathbf{w}) \propto P(w_{di}|T_{di} = t, \mathbf{t}_{-di}, \mathbf{w}_{-di})P(T_{di} = t|\mathbf{t}_{-di}) \tag{3.6}$$

$$\propto \frac{c \cdot n^A_{w_{di}t} + n^{w_{di}}_{-w_{di},t} + \beta}{c \cdot n^A_{\cdot,t} + n^{\cdot}_{-w_{di},t} + |\mathcal{W}|\beta} \frac{n^d_{-w_{di},t} + \alpha}{n^d_{-w_{di},\cdot} + |\mathcal{T}|\alpha} \tag{3.7}$$

Here $n^A_{w_{di},t}$ is the number of terms with word $w_{di}$ assigned to topic $t$ in corpus $\mathcal{A}$ and $n^{w_{di}}_{-w_{di},t}$ is the number of terms with word $w_{di}$ assigned to topic $t$ in $\mathcal{B}$ excluding the $ith$ term of document $d$. LDA-Trans is described in Algorithm 8.

**Algorithm 8** LDA-Transfer Model

**Inputs:** $\mathcal{F}_{\mathcal{A}}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{T}$, and weight $c$

**Output:** $\theta$, $\phi$ and $perplexity_c$

**for** each $d \in \mathcal{B}, i \in \{1, ..., N_d\}$ **do**

Apply weight $c$ to sample the topic of each word:

$$P(T_{di} = t|\mathbf{t}_{-di}, \mathbf{w}) \propto P(w_{di}|T_{di} = t, \mathbf{t}_{-di}, \mathbf{w}_{-di})P(T_{di} = t|\mathbf{t}_{-di})$$

$$\propto \frac{c \cdot n^A_{w_{di}t} + n^{w_{di}}_{-w_{di},t} + \beta}{c \cdot n^A_{\cdot,t} + n^{(\cdot)}_{-w_{di},t} + |\mathcal{W}|\beta} \frac{n^{(d)}_{-w_{di},t} + \alpha}{n^{(d)}_{-w_{di},\cdot} + |\mathcal{T}|\alpha}$$

**end for**

Compute topic-word distribution $\phi = \frac{n^{(w_{di})}_{-w_{di},t} + \beta}{n^{(\cdot)}_{-w_{di},t} + |\mathcal{W}| * \beta}$

Compute document-topic distribution $\theta = \frac{n^{(d)}_{-w_{di},t} + \alpha}{n^{(d)}_{-w_{di},\cdot} + |\mathcal{T}| * \alpha}$

Apply LDA on Test data set $\mathcal{C}$ with $\phi$

Calculate Perplexity of Test data set $\mathcal{C}$ via

$$perplexity_c = perplexity(\tilde{\mathbf{w}}|\mathcal{M}) = exp(-\frac{\sum_{d \in \mathcal{C}} logP(\mathbf{w}_d|\mathcal{M})}{\sum_{d \in \mathcal{C}} n_d})$$

**return** $perplexity_c$

Note that the running time of the algorithm is dominated by the time to search for the

best weight $c$ which is $O(2 \log n)$ that of LDA. Here, $n$ is the number of possible values that are considered for $c$. For example, we set $ACC$ (accuracy) to 0.001 in this thesis. Hence, there are 1000 values between 0 and 1, however, we only need to evaluate the perplexity of about 20 weights need to ensure an accuracy of 0.001.

We will show in the experiment section of this chapter that by using $\mathcal{A}$ in this fashion, we obtain a perplexity for $c$ that is much better than training based on $\mathcal{B}$ only and almost the same as training with $\mathcal{A} \bigcup \mathcal{B}$, but in less time.

## 3.4 HDP Transfer Model

If we already know the number of topics in the language, then LDA-Trans is an efficient model for transfer learning. However, if we don't know the number of topics in the language, then LDA-Trans is quite slow because we need to try several numbers of topics to find the best one. Moreover, after we generate a language model for $\mathcal{A}$ and we get the topic set $\mathbf{t}_{\mathcal{A}}$, we still don't know whether the topic set in $\mathcal{B}$ will be covered by $\mathbf{t}_{\mathcal{A}}$. In other words, if there is a new topic $t_{new}$ in $\mathcal{B}$, it cannot be found by LDA-Trans. Hence, we consider HDP to build an alternative transfer model. In addition to weighting the importance of $\mathcal{A}$, we allow the number of topics to vary. This is quite useful when $\mathcal{B}$ is significantly different from $\mathcal{A}$. In order to apply HDP, we need more information from $\mathcal{A}$.

Besides the term frequencies, we also need to know how many groups $m_{dt}$ in each document $d$ are assigned to each topic $t$. Because we use LDA to generate a language model for $\mathcal{A}$, there is no variable to store the information about the groups. However, we can sample $M_{dt}$, the number of groups to be assigned with topic $t$ in document $d$, from the average topic distribution over documents, denoted as $\bar{\theta}$. When the language model is obtained by LDA, we do not have a cluster-level topic mixture. However, the document level topic mixtures $\theta_d$ are sampled i.i.d. from some Dirichlet for each document, so we construct a cluster-level topic mixture by taking the average: $\bar{\theta} = \sum_d \theta_d$. So, $M_{dt}$ can be sampled as follows:

$$P(M_{dt}^{\mathcal{A}} = m | \mathbf{m}_{-dt}^{\mathcal{A}}, \bar{\theta}) \propto \frac{\Gamma(\alpha \bar{\theta}_t)}{\Gamma(\alpha \bar{\theta}_t + n_{d \cdot t})} s(n_{d \cdot t}^{A}, m) (\alpha \bar{\theta}_t)^m. \tag{3.8}$$

Here, $n_{d\cdot t}^{\mathcal{A}}$ is the number of words assigned to topic $t$ in document $d$ in $\mathcal{A}$. Also, the weight $c$ in HDP-Trans can be generated by the same method as LDA-Trans. Then, we can apply HDP to learn a language model for the new cluster as Algorithm 9.

**Algorithm 9** HDP-Transfer Learning

**Inputs:** $\mathcal{F}_{\mathcal{A}}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{T}$ and weight $c$

**Output:** $\phi$, $\theta$, and $perplexity_c$

**for** each $d \in \mathcal{B}$, and $i \in \{1, ..., N_d\}$ **do**

Compute $f_t^{-w_{di}}(w_{di}) = \frac{cn_{w_{di},t}^{\mathcal{A}} + n_{w_{di},t}^{-di} + \beta}{cn_{\cdot,t}^{\mathcal{A}} + n_{\cdot,t}^{-di} + |\mathcal{W}|\beta}$

Sample the topic $t_{di}$ of the $ith$ word in the document $d$ in $\mathcal{B}$ by :

$$P(T_{di} = t|\mathbf{t}^{-di}, \mathbf{m}, \bar{\mathbf{t}}) \propto \begin{cases} \alpha\bar{\theta}_u f_t^{-w_{di}}(w_{di}), & \text{if } t \text{ is new;} \\ (n_{d\cdot t}^{-di} + \alpha\bar{\theta}_t) f_t^{-w_{di}}(w_{di}), & \text{otherwise.} \end{cases}$$

Sample $m_{dt}$ for document $d$ in $\mathcal{B}$ by

$$P(M_{dt} = m|\mathbf{t}, \mathbf{m}^{-dt}, \bar{\theta}) \propto \frac{\Gamma(\alpha\bar{\theta}_t)}{\Gamma(\alpha\bar{\theta}_t + n_{d\cdot t})} s(n_{d\cdot t}, m)(\alpha\bar{\theta}_t)^m;$$

Sample $\bar{\theta}$ according to $(\bar{\theta}_1, ..., \bar{\theta}_k, \bar{\theta}_u) \sim Dir(m_{\cdot 1}, ..., m_{\cdot k}, \gamma)$

calculate $m_{\cdot t} = \sum_d m_{dt} + c \cdot m_{\cdot t}^{\mathcal{A}}$.

**end for**

Compute topic-word distribution $\phi = \frac{n_{-w_{di},t}^{(w_{di})} + \beta}{n_{-w_{di},t}^{(\cdot)} + |\mathcal{W}| * \beta}$

Compute document-topic distribution $\theta = \frac{n_{-w_{di},t}^{(d)} + \alpha}{n_{-w_{di},\cdot}^{(d)} + |\mathcal{T}| * \alpha}$

Apply LDA on Test data set $\mathcal{C}$ with $\phi$

Calculate Perplexity of Test data set $\mathcal{C}$ via

$$perplexity_c = perplexity(\tilde{\mathbf{w}}|\mathcal{M}) = exp(-\frac{\sum_{d \in \mathcal{C}} logP(\mathbf{w}_d|\mathcal{M})}{\sum_{d \in \mathcal{C}} n_d})$$

**return** $perplexity_c$

In the step that samples topic $t_{di}$, a new topic may be generated with a probability, that depends on the hyperparameter $\gamma$. Hence, HDP-Trans can generate new topics for

the new cluster. Moreover, if there exist more topics than necessary, HDP will also delete extra topics that have not been assigned any word and that do not exist in $\mathcal{A}$. In this way, HDP-Trans can generate a suitable number of topics for $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$.

## 3.5 Experiments

### 3.5.1 Experiments Dataset

We use three different data sets to evaluate our models: two from Reuters and one provided by our industry partner, Google Inc. The Reuters data set ("Reuters 21578, Distribution 1.0") can be obtained via Lewis' professional home page [1]. This data set is widely used in information retrieval, machine learning, and other corpus-based research areas. The data was originally collected and labeled by Carnegie Group, Inc. and Reuters, Ltd. in the course of developing the $CONSTRUE$ text categorization system. The documents in the Reuters-21578 collection appeared on the Reuters newswire in 1987. The documents were assembled and indexed with categories (or topics) by personnel from Reuters Ltd. and Carnegie Group, Inc. There are 135 topics in this data set. Considering the hardware available and the model complexity, we just choose part of the data set to build our test cases.

Each test case consists of a large side corpus $\mathcal{A}$ and a new cluster $\mathcal{B} \bigcup \mathcal{C}$, which needs to be labeled. The new cluster is divided in a training subset $\mathcal{B}$ and a testing subset $\mathcal{C}$. Moreover, the subset $\mathcal{B}$ includes 90% of the documents from the new cluster, and the subset $\mathcal{C}$ includes 10% of the documents.

For the first case, we use a large data set $\mathcal{A}$ as our prior knowledge, which consists of random documents from the ACQ (acquisition), AUSTDLR (Australian Dollar), BARLEY, CARCASS, COCOA and WHEAT categories. We randomly choose some documents from the ALUM (aluminium) cluster to be our new cluster $\mathcal{B} \bigcup \mathcal{C}$. This is an example of a situation where the new cluster is quite different from the large side corpus.

[1] http://www.research.att.com/~lewis

| Properties | ALUM | ACQ | Industry |
|---|---|---|---|
| number of docs in A | 454 | 195 | 1047 |
| number of docs in B | 53 | 31 | 45 |
| number of docs in C | 5 | 4 | 4 |
| size of dictionary | 5473 | 2587 | 43154 |
| number of terms in A | 38124 | 11851 | 456298 |
| number of terms in B $\bigcup$ C | 4555 | 2172 | 17948 |
| Labeling word | alum | acq | N/A |

Table 3.1: Size and properties of each dataset

The cluster $\mathcal{B} \bigcup \mathcal{C}$ of the second Reuters data set consists of 35 documents from the ACQ (aluminium) cluster, while the side corpus $\mathcal{A}$ owns 195 documents from ACQ (different from those in $\mathcal{B} \bigcup \mathcal{C}$). This is an example of a situation where the cluster and corpus share similar content.

The cluster $\mathcal{B} \bigcup \mathcal{C}$ of the industry data set consists of web pages from a set of jewelry shopping sites, while the side corpus $\mathcal{A}$ consists of web pages from other clusters assembled by our industry partner. From now on, we will refer to these data sets by ALUM, ACQ and Industry. Table 3.1 indicates the size of the documents, the dictionary, the number of terms, and labeling word for each data set.

### 3.5.2 Comparisons

In the first step of our labeling task, we build a language model for each large side corpus $\mathcal{A}$ in each data set using LDA. We separate the corpus $\mathcal{A}$ in two parts: one part that contains 90% of the documents for training, and the rest for testing. Since the number of topics of each data set is unknown, we denote $|\mathcal{T}_{LDA}|$ to be the number of topics set a priori for LDA. The range of $|\mathcal{T}_{LDA}|$ is from 1 to 100. Also, we set $\alpha = 5/(3 * |\mathcal{T}_{LDA}|)$ and $\beta = 0.01$ for each data set. After we generated a perplexity curve in function of the number of topics, we found that the perplexity goes down as the number of topics increases without any sign of overfitting. The reason for this is that each document is allowed to

have its own mixture of topics, which means that the more topics the better the fit will be. In other words, we can think of the topics as basis functions and the mixture of topics as coefficients of the basis functions. So, to keep the number of topics in a reasonable range, we set a penalty function for our LDA models. The purpose of this penalty function is to increase perplexity as the number of topics increases. For example, we use $perplexity + 2 * thenumberoftopic$ as a penalty function in this thesis. Also, we use "penalized perplexity" to denote the perplexity value after apply a penalty function. The perplexity curves for the three data sets are shown in Figures 3.1, 3.2, and 3.3.

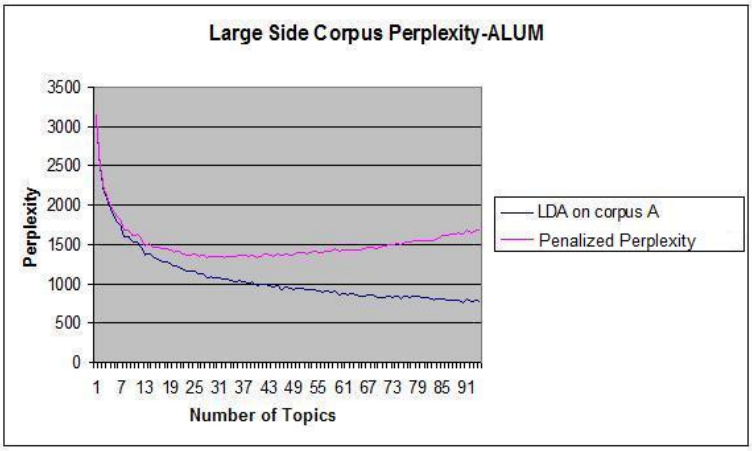


Figure 3.1: LDA on Large Side Corpus A - ALUM

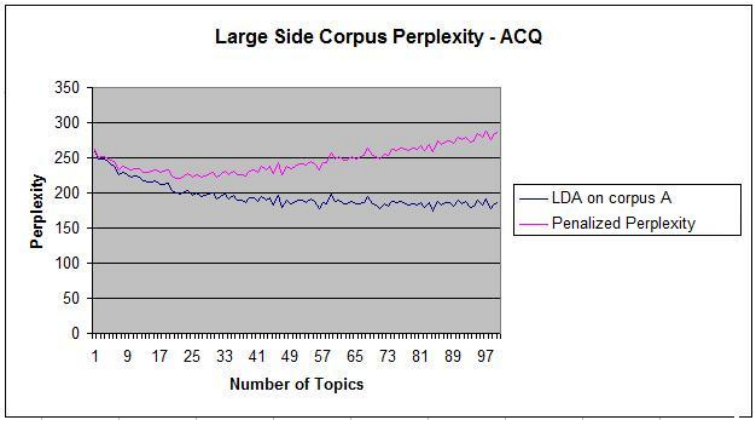


Figure 3.2: LDA on Large Side Corpus A - ACQ

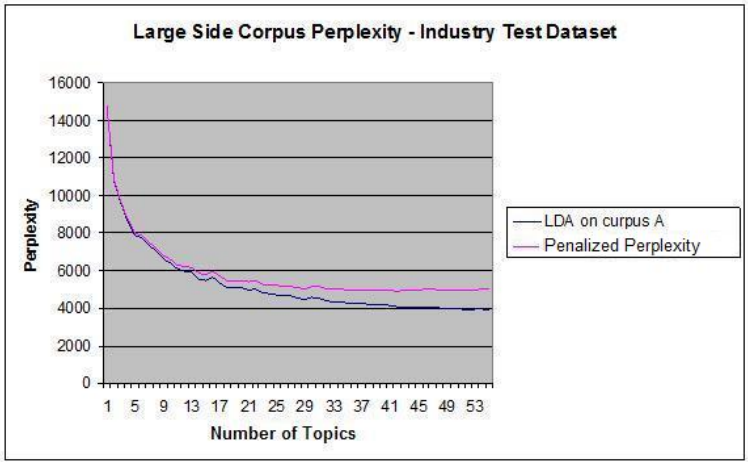


Figure 3.3: LDA on Large Side Corpus A - Industry Test Case

The results suggest that the best number of topics for ALUM is 28, for ACQ is 24, and for the Industry Test Case is 42. At the same time, we also obtain the frequencies $n_{wt}$ of the topic-word assignment for $\mathcal{A}$.

The second step is to transfer the language model learned from the large side corpus to the new cluster. More specifically, we use the frequencies $n_{wt}$ of $\mathcal{A}$'s topic assignment weighted by the best weight $c$ found for each cluster to set the prior for each cluster. This

| Parameters | ALUM | ACQ | Industry |
|---|---|---|---|
| $\|\mathcal{T}_{LDA}\|$ | 28 | 24 | 42 |
| $\alpha_{LDA}$ | $5/(3 * \|\mathcal{T}_{LDA}\|)$ | $5/(3 * \|\mathcal{T}_{LDA}\|)$ | $5/(3 * \|\mathcal{T}_{LDA}\|)$ |
| $\beta$ | 0.01 | 0.01 | 0.01 |
| $\alpha_{HDP}$ | 5/3 | 5/3 | 5/3 |
| $\gamma$ | 0.1 | 0.01 | 0.1 |

Table 3.2: Parameters for each dataset

process summarizes the LDA-Trans and HDP-Trans models for the new text cluster. The parameters for these models are shown in Table 3.2.

The perplexities of LDA-Trans and HDP-Trans for each test data set are shown in Figures 3.4, 3.5, and 3.6. In those figures, we show the perplexity of 4 language modeling techniques for the cluster of each dataset. In all cases, testing is done on the subset $\mathcal{C}$ of each cluster. The first technique uses LDA and trains only on $\mathcal{B}$, which is fast, but not robust when the cluster is small. The second technique uses LDA to train on $\mathcal{A} \bigcup \mathcal{B}$, which may improve or worsen the quality of the language model depending on the content similarity of $\mathcal{A}$ and $\mathcal{B}$. Furthermore, training takes much longer due to the inclusion of the side corpus (the run time is reported in the next section). The third and fourth approaches train with LDA and HDP respectively on $\mathcal{B}$ with the weighted frequencies of $\mathcal{A}$ included in the prior.

For the ALUM dataset, since the content of the cluster and corpus are quite different, there is no advantage to include $\mathcal{A}$ in the training. Figure 3.4 confirms that training only on $\mathcal{B}$ yields lower perplexity than training on $\mathcal{A} \bigcup \mathcal{B}$ for a small number of topics. As the number of topics increases, all techniques eventually perform similarly. The best weight found to scale the frequencies of $\mathcal{A}$ in the prior of LDA and HDP was lower than 0.1, confirming again that $\mathcal{A}$ provides no relevant information to the language model of $\mathcal{B}$. In addition to ignoring $\mathcal{A}$, the HDP-Trans technique increased the number of clusters to 29 to better fit cluster $\mathcal{B}$ and to obtain a slightly lower perplexity.

For the ACQ dataset, since all of the documents of the side corpus are from the same

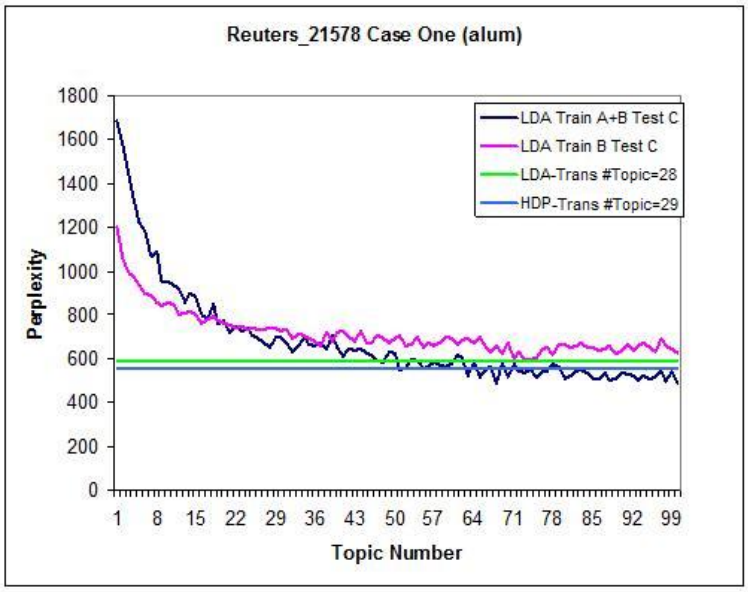


Figure 3.4: Perplexity - ALUM

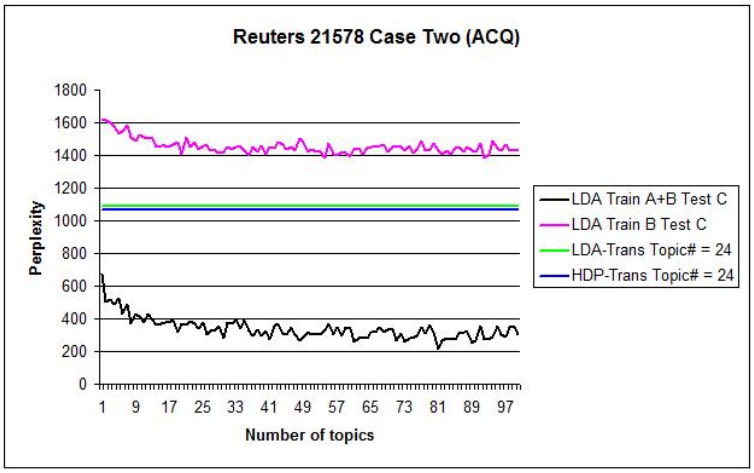


Figure 3.5: Perplexity - ACQ

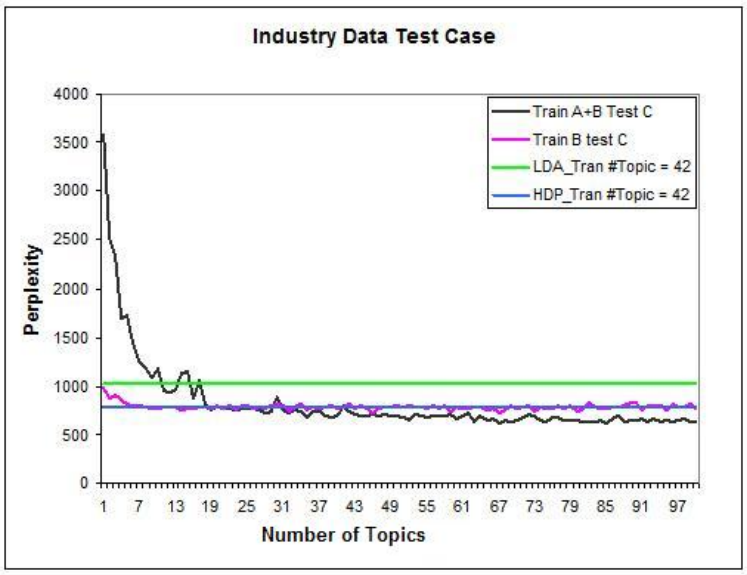


Figure 3.6: Perplexity - Industry

homogeneous set as the cluster, including $\mathcal{A}$ in the training improves the quality of the language model. This is confirmed in Figure 3.5 where the perplexity of training on $\mathcal{A} \bigcup \mathcal{B}$ is clearly lower than training on $\mathcal{B}$ only. However, the training time was much longer as shown in Table 3.3. The approaches that include the weighted frequencies of $\mathcal{A}$ in the prior for LDA and HDP achieved a lower perplexity than only training on $\mathcal{B}$. While the best weight is small (i.e., 0.195), note that it is more than 30 times larger than the best weight for the ALUM cluster and the resulting weighted frequencies had a clear effect since the perplexity was decreased to much lower than the perplexity obtained by training on $\mathcal{B}$ only. Since the LDA-Trans and HDP-Trans techniques perform the training on $\mathcal{A}$ in a separate initial off-line phase, they only need to train on $\mathcal{B}$ at the time of labeling the cluster. The fact that the weighted frequencies of $\mathcal{A}$ are included in the prior does not affect the running time since Gibbs sampling only needs to re-sample the topic assignments of the words in $\mathcal{B}$. As a result, HDP-Trans and LDA-Trans obtain a language model of close quality to training on $\mathcal{A} \bigcup \mathcal{B}$, but at a much lower cost.

For the industry dataset, the cluster and side corpus include web pages of various sites, but it wasn't clear to us initially how related their content would be. This is also

confirmed in Figure 3.6 where training only on $\mathcal{B}$ has lower perplexity than training on $\mathcal{A} \bigcup \mathcal{B}$. While LDA-Trans does a bit worse than training only on $\mathcal{B}$, HDP-Trans obtains a similar perplexity.

## 3.6 Running Time

In order to deploy our transfer model in an industrial environment, we not only focus on the accuracy, but also consider the run time. Since the size of a cluster to be labeled may not be large enough to produce a robust language model, our transfer learning models use a large side corpus as prior knowledge to complement the cluster. However, since the side corpus is processed offline once, our transfer learning approaches have the benefit that the online time to learn the language model of the cluster depends only on the size of the cluster. This is particularly useful when we consider the fact that new clusters of documents are generated every day on the web and there is a need to process them as they arise.

| | LDA | | LDA | | LDA-Trans | | | HDP-Trans | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Training set contains: | B | | A+B | | A, B, ACC=0.001 | | | A, B, ACC=0.001 | | |
| | topic# | time | topic# | time | topic# | time | weight | topic# | time | weight |
| ALUM | 1~100 | 21.5 min | 1~100 | 230.4 min | 28 | 5.42 min | 0.07 | 28->29 | 192.5 min | 0.041 |
| ACQ | 1~100 | 11.5 min | 1~100 | 83.8 min | 24 | 2.46 min | 0.195 | 24 | 82.6 min | 0.274 |
| Industry Test Data | 1~100 | 118.6 min | 1~100 | 3268.9 mir | 42 | 35.1 min | 0.008 | 42 | 1303 min | 0.011 |

Table 3.3: Running time

Table 3.3 reports the online running time of building a language model for a new cluster $\mathcal{B}$ with 4 different approaches. The simplest approach consists of running LDA on $\mathcal{B}$ (first column of Table 3.3). Since the number of topics is unknown for LDA, LDA the time reported is for 100 runs of LDA for 1 to 100 topics where the best number of topics is selected by minimizing the perplexity with a penalty term that acts as a regularizer. This is our first baseline, which has a fast running time, but may not be robust when $\mathcal{B}$ is small as demonstrated in the previous section. Our second baseline consists of running LDA on

the union of $\mathcal{A}$ and $\mathcal{B}$ to improve the robustness of the language model. Again, LDA is run 100 times to find the best number of topics. Since the running time scales linearly with the size of the dataset and $\mathcal{A}$ is a large side corpus, it takes much longer to run. When the side corpus is really large (Terabytes or even Petabytes), this approach does not scale. We compared those two baselines to our proposed transfer learning models: LDA-Trans and HDP-Trans. Here only the online time is reported. Although the side corpus is used in both models, Gibbs sampling is performed only with respect to $\mathcal{B}$. The frequencies of $\mathcal{A}$ are included in the prior of the models, but they do not affect the running time of Gibbs sampling. As a result, LDA-trans is much faster than LDA on $\mathcal{A} \bigcup \mathcal{B}$. It is also faster than LDA on $\mathcal{B}$ only because the number of topics is optimized offline based on $\mathcal{A}$. HDP-Trans takes about the same amount of time as LDA on $\mathcal{A} \bigcup \mathcal{B}$ and is slower than LDA on $\mathcal{B}$ because it adjusts the number of topics online. Note here that HDP-Trans would be faster than LDA on $\mathcal{A} \bigcup \mathcal{B}$ had we considered larger side corpus since the running time of HDP-Trans is not affected by the size of $\mathcal{A}$. Recall also from the previous section that the transfer models produce robust language models that weigh the side corpus according to the degree of similarity of $\mathcal{A}$ and $\mathcal{B}$, and HDP-Trans also adjusts the number of topics. In the next chapter we will show how to generate labels based on the language models obtained by LDA-Trans and HDP-Trans.

# Chapter 4

# Cluster Labeling

For our labeling task, we would like to return the words with the highest probability of being generated by the language model for $\mathcal{B}$. When the language model is obtained by HDP, we use the corpus level topic mixture $\bar{\theta}$ and the word distribution $\phi_t$ for each topic to compute the probability of generating word $w$ as follows:

$$P(w) = \sum_{t \in \mathcal{T}} \bar{\theta}_t \phi_{tw} \tag{4.1}$$

Note that the words with the highest probability of being generated are similar to those with the highest frequency in $\mathcal{B}$. However, the language model smooths out the frequencies in a way that words that do not appear frequently, but are semantically related to other frequent words, will see their generation probability boosted. For instance, a cluster of web pages about jewelry shopping may repeat the words of specific items such as ring, gold and diamond frequently, but may not include the word jewelry as frequently, although it is the unifying concept. Since jewelry is closely related to these items, we expect its generation probability to be boosted and therefore to rank higher among the top keywords.

In general, selecting words with highest generation probability is not perfect. Ideally, we would like the keywords to be representative of the cluster, but also to distinguish the cluster from other clusters. Common words will often have a high generation probability, but won't be specific enough to the cluster to distinguish it. Such common words tend to be part of every topic. So to bias the labels towards words that are more specific, we consider

only the dominating topics. More precisely, we modify the above equation to include the most likely topics up to a cumulative probability which is denoted as $CPV$. Without loss of generality, assuming that the topics are ranked in decreasing order of mixture probability (i.e., $\bar{\theta}_1 \geq \bar{\theta}_2 \geq ... \geq \bar{\theta}_{|\mathcal{T}|}$) then we can score words as follows:

$$Score(w) = \sum_{t=1}^{k} \bar{\theta}_t \phi_{tw}, \tag{4.2}$$

and

$$\sum_{t=1}^{k-1} \bar{\theta}_t < CPV \ and \ \sum_{t=1}^{k} \bar{\theta}_t \geq CPV \tag{4.3}$$

## 4.1 Experiments

The label results are shown in Table 4.1. It compares the labels found by selecting the 19 words that are the most likely to be generated by the language models obtained by LDA-Trans and HDP-Trans to the 19 most frequent words of each cluster (after removal of the stop words) and the 19 words found by LDA when trained on $\mathcal{B}$ only (training on the union of $\mathcal{A}$ and $\mathcal{B}$ is not practical for large side corpora as discussed in the previous chapter). We manually bolded the keywords that are the most representative for each cluster.

For the ALUM data set, the cluster consists of financial news articles about the aluminium industry. Here HDP-Trans ranks the two spellings "aluminium" and "aluminum" at the top, which is optimal. HDP-Trans and LDA-Trans both decreased the rank of the generic words "reuters" and "tonne" despite having the second and third highest frequency. In comparison to LDA trained on $\mathcal{B}$ only, the results for LDA-Trans and HDP-Trans are similar. This is exactly what we want since as explained in the previous chapter that $\mathcal{A}$ and $\mathcal{B}$ have no relationship for the ALUM case. So the danger is that by using $\mathcal{A}$, LDA-Trans and HDP-Trans may end up producing worse results, but they effectively ignored $\mathcal{A}$ and produced good results.

For the ACQ data set, the documents consist of financial news articles about acquisitions and mergers. In this case, $\mathcal{A}$ and $\mathcal{B}$ are from the same set of documents and therefore

| Reuters-21578 Case One (alum) | | | | Reuters-21578 Case Two (acq) | | | |
|---|---|---|---|---|---|---|---|
| Frequency | LDA | LDA-Trans | HDP-Trans | Frequency | LDA | LDA-Trans | HDP-Trans |
| **aluminium** | **aluminium** | contract | **aluminum** | **share** | **share** | **share** | **offer** |
| tonne | **price** | **aluminium** | **aluminium** | **company** | corp | **offer** | **share** |
| reuter | cost | credit | **smelter** | reuter | reuter | sale | **company** |
| **aluminum** | **alcan** | company | **alcan** | **offer** | **company** | **merger** | **acquire** |
| company | credit | **aluminum** | credit | viacom | agreed | amusement | viacom |
| year | **smelter** | **price** | company | corp | viacom | unit | corp |
| **smelter** | company | tonne | plan | group | **offer** | benequity | group |
| **price** | plant | market | azpurua | **stock** | plan | viacom | **tender** |
| plant | capacity | year | reuter | holding | **stock** | **commerice** | exercisable |
| contract | azpurua | source | **price** | stake | **merger** | **company** | transport |
| **metal** | reuter | billiton | famer | security | shareholder | value | financing |
| state | tonne | azpurua | wheat | **management** | sale | wholly | traffic |
| billion | billiton | january | state | redstone | stake | asset | **stock** |
| credit | world | official | cent | common | circuit | revenue | **merger** |
| alcan | spokesman | indonesia | **industry** | bank | common | speculated | holding |
| cost | year | finance | cost | agreed | holding | profit | stake |
| **industry** | refinery | expansion | billion | **merger** | subsidiary | delayed | offered |
| capacity | electricity | project | farm | cash | sell | march | preferred |
| azpurua | higher | financing | production | owned | outstanding | variou | **management** |

| Industry Data Test Case | | | | |
|---|---|---|---|---|
| Frequency | LDA | LDA-Trans | HDP-Trans | Industry Ref-Labels |
| **ring** | **gold** | **gold** | **gold** | **ring** |
| **gold** | **diamond** | **ring** | **jewelry** | **rings** |
| **diamond** | **ring** | **jewelry** | **sapphire** | **gold** |
| **sapphire** | **jewelry** | samuel | white | **diamond** |
| size | white | available | earring | white-gold |
| earring | **sapphire** | **diamond** | **bracelet** | |
| **jewelry** | **bracelet** | **gemstone** | yellow | |
| available | yellow | white | product | |
| samuel | product | size | price | |
| **bracelet** | price | designer | **pearl** | |
| **silver** | **gemstone** | earring | **gemstone** | |
| designer | pearl | small | **necklace** | |
| **necklace** | **necklace** | product | pink | |
| black | stone | aquamarine | information | |
| aquamarine | **silver** | yellow | stone | |
| pink | **pendant** | **necklace** | **pendant** | |
| yellow | search | wedding | gift | |
| product | home | pink | **silver** | |
| blue | collection | **bracelet** | search | |

Table 4.1: Labels for each data set

we expect LDA-Trans and HDP-Trans to do better than basic frequencies and LDA trained on $\mathcal{B}$ only. In fact, HDP-Trans obtained the keyword, "acquire", which was generated from the side corpus since the cluster doesn't contain that word. Also, HDP-Trans ranked "tender" fairly high even though this word appears only twice in the cluster. LDA-Trans improved the ranking of "merger" in comparison to the ranking based on Frequency and LDA.

Finally, the industry cluster consists of web pages from jewelry shopping sites. The last table includes the keywords currently used by our industry partner in the "Industry Ref-Labels" column. While many keywords denoting specific pieces of jewelry have higher frequency than "jewelry" itself, both LDA-Trans and HDP-Trans improved the ranking of "jewelry" in comparison to the ranking based on frequency.

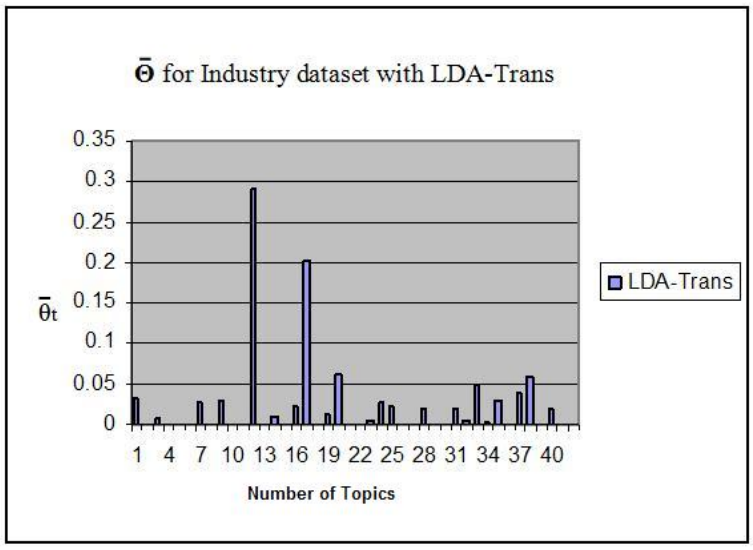


Figure 4.1: LDAs topic mixture $\bar{\theta}$ for the industry cluster.

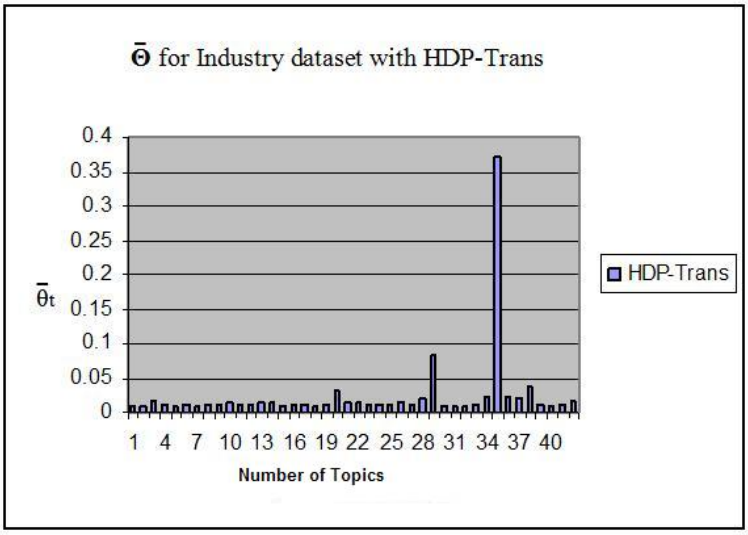


Figure 4.2: HDPs topic mixture $\bar{\theta}$ for the industry cluster.

We can gain some insights into the performance of HDP-Trans and LDA-Trans by comparing the topic mixtures $\bar{\theta}$ of their language models. According to the pseudocode for HDP, HDP-Trans does not only sample a topic for each word in the document, but it also samples $m_{dt}$ for each document. So, HDP-Trans tends to produce topic mixtures that are concentrated in fewer topics as illustrated in Figures 4.1 and Figures 4.2. This may explain the better language models and better labels obtained by HDP-Trans.

# Chapter 5

# Conclusion

## 5.1 Summary

Our objective is to design a new approach for efficiently labeling a bunch of text documents in a cluster. In this thesis, we studied and reviewed n-gram language models, frequency models, topic models and some related models for text analysis. Among those approaches, the n-gram model considers a previous subsequence of $n-1$ words to predict the next word. To improve the robustness of the predictions, it is common to smooth n-gram models by combining them with lower order gram models or setting the prior with a weighted combination of lower-order gram models. We borrowed this idea to design our transfer learning models for topic modeling. N-gram methods obtain low perplexity because they consider the word order, however, they are not suitable for cluster labeling because they do not capture any notion of semantics. The frequency model uses term frequency and document frequency to calculate and create a score metric for each word in the text data set. This kind of method can find some important words, but, it ignores any latent relationship between words. The topic model creates a topic distribution based on observed variables (words) for each document. This kind of model also generates document-topic distributions for each cluster in a data set. Since topic models focus on the semantic relatedness of words while n-gram models focus on the ordering of words induced by the syntactic structure of sentences, some researchers combined topic models with n-gram models to obtain lower

perplexity. While this hybrid approach does yield lower perplexity, it is mostly due to the n-gram part of the model since syntactic structure is usually much more informative than semantic information to predict future words. In fact the latent variables that used to be interpreted as topics do not correspond anymore to topics. They simply capture additional statistics beyond syntax to refine the prediction of future words. Unfortunately, because the latent variables do not seem to capture any notion of semantic relatedness, the resulting language model is not useful for cluster labeling. In the end, topic models seem to be the most suitable for cluster labeling. However, to ensure that the topic models are robust, it is often necessary to use a large amount of data such as a side corpus. Two issues may arise when using a side corpus: it may take too long to process the side corpus each time we want to label a cluster and the side corpus may not be related to the cluster. Hence, this thesis explores transfer modeling techniques that can process the side corpus once offline and weigh the importance of the side corpus based on the degree of similarity of the cluster and the corpus.

In chapter three, we proposed two unsupervised techniques to achieve this goal. The approaches, LDA-Trans and HDP-Trans, are based on topic modeling and transfer information from a side corpus that is processed offline in order to reduce the online computation to the cluster itself. The first step is to build a language model that implicitly quantifies the semantic relatedness of words from a large side corpus, which is employed in addition to the cluster itself. Let us use "knowledge" to denote the relationships between words. Then we transfer this "knowledge" to the prior of the language model of the cluster and automatically weight it based on the degree of similarity between the side corpus and the cluster. Finally, we use the resulting language model to label the cluster. More precisely, LDA-trans labels the cluster by reusing the topics learned from the side corpus. In contrast, HDP-trans may delete and generate new topics if the latent structure of the cluster is different from the side corpus.

As the results show in chapter three, the complexity of these new transfer learning models can be low. Moreover, as shown in chapter four, the transfer models rank better keywords higher than by using raw term frequencies. Furthermore, our transfer models permit extensive off line processing of the side corpus, but rapid labeling of new clusters.

## 5.2 Future Directions

This work could be extended in several directions. For instance, similar to online LDA, we could grow the side corpus by adding new documents as we label clusters. So, instead of keeping the topic assignments of the side corpus fixed, we could re-sample them according to a "rejuvenation" sequence [2] each time we label a new cluster. This would have the benefit of allowing the side corpus to grow, but more computation would be required each time a cluster needs to be labeled. However, the incorporation of the cluster into the side corpus could be done at a later time when some computing resources are available.

Another extension could consist of an alternative technique to determine the degree of similarity between the side corpus and the cluster. Instead of optimizing a weight by cross validation, which is time consuming, one could compare the topic distributions of the side corpus $\mathcal{A}$ and the cluster $\mathcal{B}$. More precisely, after generating the topic distribution for each word, we could get a topic-document distribution $\theta_d : d \in \mathcal{A}$ for each document $d$ inside corpus $\mathcal{A}$. Let $\theta_{\mathcal{A}}$ be the sum of the $\theta_d$'s in $\mathcal{A}$ and similarly, let $\theta_{\mathcal{B}}$ be the sum of the $\theta_d$'s in $\mathcal{B}$. Then, the similarity between $\mathcal{A}$ and $\mathcal{B}$ could be calculated as:

$$Similarity = \cos(\theta) = \frac{\theta_{\mathcal{A}} * \theta_{\mathcal{B}}}{||\theta_{\mathcal{A}}|| * ||\theta_{\mathcal{B}}||} \tag{5.1}$$

Our approaches' performances are good for web content text. As cloud computing becomes widely used, there is a great amount of data stored in various forms in the cloud. I believe extending our transfer models to label clusters of files with various data types, such as binary files or jpg files, should be a very interesting topic in the future.

# Bibliography


[1] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003. 14

[2] Kevin R. Canini, Lei Shi, and Thomas L. Griffiths. Online inference of topics with latent dirichlet allocation. *Journal of Machine Learning Research - Proceedings Track*, 5:65–72, 2009. 19, 52

[3] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. *Statistics and computing*, 10(3):197–208, 2000. 20

[4] T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1(2):209–230, 1973. 22

[5] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2nd edition, 2004. 5, 14

[6] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235, April 2004. 15

[7] H. Jeffreys. *Theory of Probability*. Clarendon Press, Oxford, 1948. 2nd edn Section 3.23. 13

[8] Fred Jelinek and Robert L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In Edzard S. Gelsema and Laveen N. Kanal, editors, *Proceedings, Workshop on Pattern Recognition in Practice*, pages 381–397. North Holland, Amsterdam, 1980. 13

[9] G.J. Lidstone. Note on the general case of the Bayes-Laplace formula for inductive or a posteriori probabilities. *Transactions of the Faculty of Actuaries*, 8:182–192, 1920. 13

[10] D. J. C. MacKay. Introduction to Monte Carlo methods. In M. I. Jordan, editor, *Learning in Graphical Models*, NATO Science Series, pages 175–204. Kluwer Academic Press, 1998. 15

[11] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, 2001. 4

[12] Bhaskara Marthi, Hanna Pasula, Stuart J. Russell, and Yuval Peres. Decayed MCMC Filtering. In *UAI*, pages 319–326, 2002. 18

[13] G. Salton. *Automatic information organization and retrieval*. New York: McGraw-Hill, 1968. 6

[14] Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz-Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. Statistical Machine Translation. *Final Report, JHU Summer Workshop*, 1999. 30

[15] Xiaodan Song, Ching-Yung Lin, Belle L. Tseng, and Ming-Ting Sun. Modeling and predicting personal information dissemination behavior. In *KDD*, pages 479–488, 2005. 17

[16] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Sharing clusters among related groups: Hierarchical Dirichlet Processes. In *NIPS*, 2004. 22

[17] João Ventura and Joaquim Ferreira da Silva. *Ranking and Extraction of Relevant Single Words in Text*. InTech Education and Publishing, Austria, 2008. 8, 10

[18] Hanna M. Wallach. Topic modeling: beyond bag-of-words. In *ICML*, pages 977–984, 2006. 25

[19] H. Zhou and G. Slater. A metric to search for relevant words. *Physica A: Statistical Mechanics and its Applications.*, 329(1):309–327, 2003. 7