Chapter 8

PROBABILISTIC MODELS FOR TEXT MINING

Yizhou Sun
Department of Computer Science
University of Illinois at Urbana-Champaign
sun22@illinois.edu

Hongbo Deng
Department of Computer Science
University of Illinois at Urbana-Champaign
hbdeng@illinois.edu

Jiawei Han
Department of Computer Science
University of Illinois at Urbana-Champaign
hanj@illinois.edu

Abstract A number of probabilistic methods such as LDA, hidden Markov models, Markov random fields have arisen in recent years for probabilistic analysis of text data. This chapter provides an overview of a variety of probabilistic models for text mining. The chapter focuses more on the fundamental probabilistic techniques, and also covers their various applications to different text mining problems. Some examples of such applications include topic modeling, language modeling, document classification, document clustering, and information extraction.

Keywords: Probabilistic models, mixture model, stochastic process, graphical model

1. Introduction

Probabilistic models are widely used in text mining nowadays, and applications range from topic modeling, language modeling, document classification and clustering to information extraction. For example, the well known topic modeling methods PLSA and LDA are special applications of mixture models.

A probabilistic model is a model that uses probability theory to model the uncertainty in the data. For example, terms in topics are modeled by multinomial distribution; and the observations for a random field are modeled by Gibbs distribution. A probabilistic model describes a set of possible probability distributions for a set of observed data, and the goal is to use the observed data to learn the distribution (usually associated with parameters) in the probabilistic model that can best describe the current data. In this chapter, we introduce several frequently used fundamental probabilistic models and their applications in text mining. For each probabilistic model, we will introduce their general framework of modeling, the probabilistic explanation, the standard algorithms to learn the model, and their applications in text mining.

The major probabilistic models covered in this chapter include:

- Mixture Models. Mixture models are used for clustering data points, where each component is a distribution for that cluster, and each data point belongs to one cluster with a certain probability. Finite mixture models require user to specify the number of clusters. The typical applications of mixture model in text mining include topic models, like PLSA and LDA.

- Bayesian Nonparametric Models. Beyesian nonparametric models refer to probabilistic models with infinite-dimensional parameters, which usually have a stochastic process that is infinite-dimensional as the prior distribution. Infinite mixture model is one type of nonparametric models, which can deal with the problem of selecting the number of clusters for clustering. Dirichlet process mixture model belongs to infinite mixture model, and can help to detect the number of topics in topic modeling.

- Bayesian Networks. A Bayesian network is a graphical model with directed acyclic links indicating the dependency relationship between random variables, which are represented as nodes in the network. A Bayesian network can be used to inference the unobserved node in the network, by learning parameters via training datasets.

- Hidden Markov Model. A hidden Markov model (HMM) is a simple case of dynamic Bayesian network, where the hidden states are forming a chain and only some possible value for each state can be observed. One goal of HMM is to infer the hidden states according to the observed values and their dependency relationships. A very important application of HMM is part-of-speech tagging in NLP.

- Markov Random Fields. A Markov random field (MRF) belongs to undirected graphical model, where the joint density of all the random variables in the network is modeled as a production of potential functions defined on cliques. An application of MRF is to model the dependency relationship between queries and documents, and thus to improve the performance of information retrieval.

- Conditional Random Fields. A conditional random field (CRF) is a special case of Markov random field, but each state of node is conditional on some observed values. CRFs can be considered as a type of discriminative classifiers, as they do not model the distribution over observations. Name entity recognition in information extraction is one of CRF's applications.

This chapter is organized as follows. In Section 2, mixture models that are frequently used in topic modeling and clustering is introduced, as well as its standard learning algorithms. In Section 3, we present several Bayesian nonparametric models, where stochastic processes are used as priors and can be used in modeling the uncertainty of the number of clusters in mixture models. In Section 4, several well-known graphical models that use nodes to represent random variables and use links in the graph to model the dependency relations between variables are introduced. Section 5 introduces several situations that constraints with domain knowledge can be integrated into probabilistic models. Section 6 is a brief introduction of parallel computing of probabilistic models for large scale datasets. The concluding remarks are given in Section 7.

2. Mixture Models

Mixture model [39] is a probabilistic model originally proposed to address the multi-modal problem in the data, and now is frequently used for the task of clustering in data mining, machine learning and statistics. Generally, a mixture model defines the distribution of a random variable, which contains multiple components and each component represents a different distribution following the same distribution family but with different parameters. The number of components are specified by

users in this section, and these mixture models are called finite mixture models. Infinite mixture models that deal with how to learn the number of components in mixture models will be covered in Section 3. To learn the model, not only the probability membership for each observed data point but also the parameter set for each component need to be learned. In this section, we introduce the basic framework of mixture models, their variations and applications in text mining area, and the standard learning algorithms for them.

2.1 General Mixture Model Framework

In a mixture model, given a set of data points, e.g., the height of people in a region, they are treated as an instantiation of a set of random variables, which are following the mixture model. Then, according to the observed data points, the parameters in the mixture model can be learned. For example, we can learn the mean and standard deviation for female and male height distributions, if we model height of people as a mixture model of two Gaussian distributions. Formally, assume we have n i.i.d. random variables $X_1, X_2, \ldots, X_n$ with observations $x_1, x_2, \ldots, x_n$, following the mixture model with K components. Let each of the kth component be a distribution following a distribution family with parameters (θ_k) and have the form of $F(x|\theta_k)$, and let π_k ($\pi_k \geq 0$ and $\sum_k \pi_k = 1$) be the weight for kth component denoting the probability that an observation is generated from the component, then the probability of x_i can be written as:

$$p(x_i) = \sum_{k=1}^{K} \pi_k f(x_i|\theta_k)$$

where $f(x_i|\theta_k)$ is the density or mass function for $F(x|\theta_k)$. The joint probability of all the observations is then:

$$p(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k f(x_i|\theta_k)$$

Let $Z_i \in \{1, 2, \ldots, K\}$ be the hidden cluster label for X_i, the probability function can be viewed as the summation over a complete joint distribution of both X_i and Z_i:

$$p(x_i) = \sum_{z_i} p(x_i, z_i) = \sum_{z_i} p(x_i|Z_i = z_i)p(z_i)$$

where $X_i|Z_i = z_i \sim F(x_i|\theta_{z_i})$ and $Z_i \sim \mathcal{M}_K(1; \pi_1, \ldots, \pi_K)$, the multinomial distribution of K dimensions with 1 observation. Z_i is also referred

to missing variable or auxiliary variable, which identifies the cluster label of the observation x_i. From generative process point of view, each observed data x_i is generated by:

1 sample its hidden cluster label by $z_i|\pi \sim \mathcal{M}_K(1; \pi_1, \ldots, \pi_K)$

2 sample the data point in component z_i: $x_i|z_i, \{\theta_k\} \sim F(x_i|\theta_{z_i})$

The most well-known mixture model is the Gaussian mixture model, where each component is a Gaussian distribution. In this case, the parameter set for kth component is $\theta_k = (\mu_k, \sigma_k^2)$, where μ_k and σ_k^2 are the mean and variance of the Gaussian distribution.

Example: Mixture of Unigrams. The most common choice of the component distribution for terms in text mining is multinomial distribution, which can be considered as a unigram language model and determines the probability of a bag of terms. In Nigam et al. [50], a document d_i composed of a bag of words $\mathbf{w}_i = (c_{i,1}, c_{i,2}, \ldots, c_{i,m})$, where m is the size of the vocabulary and $c_{i,j}$ is the number of term w_j in document d_i, is considered as a mixture of unigram language models. That is, each component is a multinomial distribution over terms, with parameters $\beta_{k,j}$, denoting the probability of term w_j in cluster k, i.e., $p(w_j|\beta_k)$, for $k = 1, \ldots, K$ and $j = 1, \ldots, m$. The joint probability of observing the whole document collection is then:

$$p(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k \prod_{j=1}^{m} (\beta_{k,j})^{c_{i,j}}$$

where π_k is the proportion weight for cluster k. Note that, in mixture of unigrams, one document is modeled as being sampled from exactly one cluster, which is not typically true, since one document usually covers several topics.

2.2 Variations and Applications

Besides the mixture of unigrams, there are many other applications for mixture models in text mining, with some variations to the general framework. The most frequent variation to the framework of general mixture models is to adding all sorts of priors to the parameters, which are sometimes called Bayesian (finite) mixture models [33]. The topic models PLSA [29, 30] and LDA [11, 28] are among the most famous applications, which have been introduced in Chapter 5 in a dimension reduction view. In this section, we briefly describe them in the view of mixture models. Some other applications in text mining, such as

comparative text mining, contextual text mining, and topic sentiment analysis, are introduced too.

2.2.1 Topic Models.

- PLSA. Probabilistic latent semantic analysis (PLSA) [29] is also known as probabilistic latent semantic indexing (PLSI) [30]. Different from the mixture unigram, where each document d_i connects to one latent variable Z_i, in PLSA, each observed term w_j in d_i corresponds to a different latent variable $Z_{i,j}$. The probability of observation term w_j in d_i is then defined by the mixture in the following:

$$p(w_j|d_i) = \sum_{k=1}^{K} p(k|d_i)p(w_j|\beta_k)$$

  where $p(k|d_i) = p(z_{i,j} = k)$ is the mixing proportion of different topics for d_i, β_k is the parameter set for multinomial distribution over terms for topic k, and $p(w_j|\beta_k) = \beta_{k,j}$. $p(k|d_i)$ is usually denoted by the parameter $\theta_{i,k}$, and $Z_{i,j}$ is then following the discrete distribution with K-d vector parameter $\theta_i = (\theta_{i,1}, \ldots, \theta_{i,K})$. The joint probability of observing all the terms in document d_i is:

$$p(d_i, \mathbf{w}_i) = p(d_i) \prod_{j=1}^{m} p(w_j|d_i)^{c_{i,j}}$$

  where $\mathbf{w}_i$ is the same defined as in the mixture of unigrams and $p(d_i)$ is the probability of generating d_i. And the joint probability of observing all the document corpus is $\prod_{i=1}^{n} p(d_i, \mathbf{w}_i)$.

- LDA. Latent Dirichlet allocation (LDA) [11] extends PLSA by further adding priors to the parameters. That is, $Z_{i,j} \sim \mathcal{M}_K(1; \theta_i)$ and $\theta_i \sim Dir(\alpha)$, where $\mathcal{M}_K$ is the K-dimensional multinomial distribution, θ_i is the K-d parameter vector denoting the mixing portion of different topics for document d_i, $Dir(\alpha)$ denotes a Dirichlet distribution with K-d parameter vector α, which is the conjugate prior of multinomial distribution. Usually, another Dirichlet prior $\beta \sim Dir(\eta)$ [11, 28] is added further to the multinomial distribution β over terms, which serves as a smoothing functionality over terms, where η is a m-d parameter vector and m is the size of the vocabulary. The probability of observing all the terms in document d_i is then:

$$p(\mathbf{w}_i|\alpha, \beta) = \int p(\mathbf{w}_i, \theta_i|\alpha, \beta) d\theta_i$$

where

$$p(\mathbf{w}_i, \theta_i|\alpha, \beta) = p(\mathbf{w}_i|\theta_i, \beta)p(\theta_i|\alpha)$$

and

$$p(\mathbf{w}_i|\theta_i, \beta) = \prod_{j=1}^{m}(\sum_{k=1}^{K} p(z_{i,j} = k|\theta_i)p(w_j|\beta_k))^{c_{i,j}}$$

The probability of observing all the document corpus is:

$$p(\mathbf{w}_1, \ldots, \mathbf{w}_n|\alpha, \eta) = \prod_{i=1}^{n} \int p(\mathbf{w}_i|\alpha, \beta)p(\beta|\eta)d\beta$$

Notice that, compared with PLSA, LDA has stronger generative power, as it describes how to generate the topic distribution θ_i for an unseen document d_i.

2.2.2 Other Applications. Now, we briefly introduce some other applications of mixture models in text mining.

- Comparative Text Mining. Comparative text mining (CTM) is proposed in [71]. Given a set of comparable text collections (e.g., the reviews for different brands of laptops), the task of comparative text mining is to discover any latent common themes across all collections as well as special themes within one collection. The idea is to model each document as a mixture model of the background theme, common themes cross different collection, and specific themes within its collection, where a theme is a topic distribution over terms, the same as in topic models.

- Contextual Text Mining. Contextual text mining (CtxTM) is proposed in [43], which extracts topic models from a collection of text with context information (e.g., time and location) and models the variations of topics over different context. The idea is to model a document as a mixture model of themes, where the theme coverage in a document would be a mixture of the document-specific theme coverage and the context-specific theme coverage.

- Topic Sentiment Analysis. Topic Sentiment Mixture (TSM) is proposed in [42], which aims at modeling facets and opinions in weblogs. The idea is to model a blog article as a mixture model of a background language model, a set of topic language models, and two (positive and negative) sentiment language models. Therefore, not only the topics but their sentiments can be detected simultaneously for a collection of weblogs.

2.3 The Learning Algorithms

In this section, several frequently used algorithms for learning parameters in mixture models are introduced.

2.3.1 Overview. The general idea of learning parameters in mixture models (and other probabilistic models) is to find a set of "good" parameters θ that maximizes the probability of generating the observed data. Two estimation criterions are frequently used, one is maximum-likelihood estimation (MLE) and the other is maximum-a-posteriori-probability (MAP).

The likelihood (or likelihood function) of a set of parameters given the observed data is defined as the probability of all the observations under those parameter values. Formally, let $x_1, \ldots, x_n$ (assumed iid) be the observations, let the parameter set be θ, the likelihood of θ given the data set is defined as:

$$\mathcal{L}(\theta \mid x_1, \ldots, x_n) = p(x_1, x_2, \ldots, x_n|\theta) = \prod_{i=1}^{n} p(x_i|\theta)$$

In the general form of mixture models, the parameter set includes both the component distribution parameter θ_k for each component k, and the mixing proportion of each component π_k. MLE estimation is then to find the parameter values that maximizes the likelihood function. Most of the time, log-likelihood is optimized instead, as it converts products into summations and makes the computation easier:

$$\log \mathcal{L}(\theta \mid x_1, \ldots, x_n) = \sum_{i=1}^{n} \log p(x_i|\theta)$$

When priors are incorporated to the mixture models (such as in LDA), the MAP estimation is used instead, which is to find a set of parameters θ that maximizes the posterior density function of θ given the observed data:

$$p(\theta|x_1, \ldots, x_n) \propto p(x_1, \ldots, x_n|\theta)p(\theta)$$

where $p(\theta)$ is the prior distribution for θ and may involve some further hyper-parameters.

Several frequently used algorithms of finding MLE or MAP estimations for parameters in mixture models are introduced briefly in the following.

2.3.2 EM Algorithm. Expectation-Maximum (EM) [7, 22, 21, 12] algorithm is a method for learning MLE estimations for probabilistic

models with latent variables, which is a standard learning algorithm for mixture models. For mixture models, the likelihood function can be further viewed as the marginal over the complete likelihood involving hidden variables:

$$\mathcal{L}(\theta \,|\, x_1, \ldots, x_n) = \sum_{\mathbf{Z}} p(x_1, \ldots, x_n, z_1, \ldots, z_n|\theta) = \prod_{i=1}^{n} \sum_{z_i} p(x_i, z_i|\theta)$$

The log-likelihood function is then:

$$\log \mathcal{L}(\theta \,|\, x_1, \ldots, x_n) = \sum_{i=1}^{n} \log \sum_{z_i} p(x_i|\theta, z_i)p(z_i)$$

which is difficult to maximize directly, as there is summation inside the logarithm operation. EM algorithm is an iterative algorithm involving two steps that maximizes the above log-likelihood, which can solve this problem. The two steps in each iteration are E-step and M-step respectively.

In **E-step (Expectation step)**, a tight lower bound for the log-likelihood called Q-function is calculated, which is the expectation of the complete log-likelihood function with respect to the conditional distribution of hidden variable $\mathbf{Z}$ given the observations of the data $\mathbf{X}$ and current estimation of parameters $\theta^{(t)}$:

$$Q(\theta|\theta^{(t)}) = \mathrm{E}_{\mathbf{Z}|\mathbf{X},\theta^{(t)}} \left[\log L(\theta; \mathbf{X}, \mathbf{Z})\right]$$

Note $L(\theta; \mathbf{X}, \mathbf{Z})$ is a complete likelihood function as it uses both the observed data $\mathbf{X}$ and the hidden cluster labels $\mathbf{Z}$.

In **M-step (Maximization-step)**, a new $\theta = \theta^{(t+1)}$ is computed which maximizes the Q-function that is derived in E-step:

$$\theta^{(t+1)} = \arg\max_{\theta} \, Q(\theta|\theta^{(t)})$$

It is guaranteed that EM algorithm converges to a local maximum of the log-likelihood function, since Q-function is a tight lower bound and the M-step can always find a θ that increases the log-likelihood. The learning algorithm in PLSA is a typical application of EM algorithm. Notice that, in M-step there could exist no closed form solution for $\theta^{(t+1)}$ and requires iterative solutions via methods such as gradient descent or Newton's method (also called Newton-Raphson method) [34].

There are several variants for EM algorithm when the original EM algorithm is difficult to compute, and some of which are listed in the following:

- Generalized EM. For generalized EM (GEM) [12], it relaxes the requirement of finding the θ that maximizes Q-function in M-step to finding a θ that increases Q-function somehow. The convergence can still be guaranteed using GEM, and it is often used when maximization in M-step is difficult to compute.

- Variational EM. Variational EM is one of the approximate algorithms used in LDA [11]. The idea is to find a set of variational parameters with respective to the hidden variables that attempts to obtain the tightest possible lower bound in E-step, and to maximize the lower bound in M-step. The variational parameters are chosen in a way that simplifies the original probabilistic model and are thus easier to calculate.

2.3.3 Gibbs Sampling. Gibbs sampling is the simplest form of Markov chain Monte Carlo (MCMC) algorithm, which is a sampling-based approximation algorithm for model inference. The basic idea of Gibbs sampling is to generate samples that converge to the target distribution, which itself is difficult to obtain, and to estimate the parameters using the statistics of the distribution according to the samples.

In [28], a Gibbs sampling-based inference algorithm is proposed for LDA. The goal is to maximize the posterior distribution of hidden variables (MAP estimation) given the observations of the documents $p(\mathbf{Z}|\mathbf{w})$, which is a very complex density function with hyper-parameters α and η that are specified by users. As it is difficult to directly maximize the posterior, Gibbs sampling is then used to construct a Markov chain of $\mathbf{Z}$, which converges to the posterior distribution in the long run. The hidden cluster $z_{i,j}$ for term $w_{i,j}$, i.e., the term w_j in document d_i, is sampled according to the conditional distribution of $z_{i,j}$, given the observations of all the terms as well as the their hidden cluster labels except for $w_{i,j}$ in the corpus:

$$p(z_{i,j}|\mathbf{z}_{-i,j}, \mathbf{w}) \propto p(z_{i,j}, w_{i,j}|\mathbf{z}_{-i,j}, \mathbf{w}_{-i,j}) = p(w_{i,j}|\mathbf{z}, \mathbf{w}_{-i,j})p(z_{i,j}|\mathbf{z}_{-i,j})$$

which turns out to be easy to calculate, where $\mathbf{z}_{-i,j}$ denotes the hidden variables of all the terms except for $w_{i,j}$ and $\mathbf{w}_{-i,j}$ denotes the all the terms except $w_{i,j}$ in the corpus. Note that the conditional probability is also involving the hyper-parameters α and η, which are not shown explicitly. After thousands of iterations (called burning period), the Markov chain is considered to be stable and converges to the target posterior distribution. Then the parameters of θ and β can be estimated according to the sampled hidden cluster labels from the chain as well as the given observations and the hyper-parameters. Please refer to [28] and

[53] for more details of Gibbs sampling in LDA, and more fundamental introductions for Gibbs sampling and other MCMC algorithms in [3].

3. Stochastic Processes in Bayesian Nonparametric Models

Priors are frequently used in probabilistic models. For example, in LDA, Dirichlet priors are added for topic distributions and term distributions, which are both multinomial distributions. A special type of priors that are stochastic processes, which emerges recently in text related probabilistic models, is introduced in this section. Different from previous methods, with the introduction of priors of stochastic processes, the parameters in such models become infinite-dimensional. These models belong to the category of *Bayesian nonparametric models* [51].

Different from the traditional priors as static distributions, stochastic process priors can model more complex structures for the probabilistic models, such as the number of the components in the mixture model, the hierarchical structures and evolution structure for topic models, and the power law distribution for terms in language models. For example, it is always a difficult task for users to determine the number of topics when applying topic models for a collection of documents, and a Dirichlet process prior can model infinite number of topics and finally determine the best number of topics.

3.1 Chinese Restaurant Process

The Chinese Restaurant Process (CRP) [33, 9, 67] is a discrete-time stochastic process, which defines a distribution on the partitions of the first n integers, for each discrete time index n. As for each n, CRP defines the distribution of the partitions over the n integers, it can be used as the prior for the sizes of clusters in the mixture model-based clustering, and thus provides a way to guide the selection of K, which is the number of clusters, in the clustering process.

Chinese restaurant process can be described using a random process as a metaphor of costumers choosing tables in a Chinese restaurant. Suppose there are countably infinite tables in a restaurant, and the nth costumer walks in the restaurant and sits down at some table with the following probabilities:

1 The first customer sits at the first table (with probability 1).

2 The nth customer either sits at an occupied table k with probability $\frac{m_k}{n-1+\alpha}$, or sits at the first unoccupied table with probability

$\frac{\alpha}{n-1+\alpha}$, where m_k is the number of existing customers sitting at table k and α is a parameter of the process.

It is easy to see that the customers can be viewed as data points in the clustering process, and the tables can be viewed as the clusters. Let $z_1, z_2, \ldots, z_n$ be the table label associated with each customer, let K_n be the number of tables in total, and let m_k be the number of customers sitting in the kth table, the probability of such an arrangement (a partition of n integers into K_n groups) is as follows:

$$p(z_1, z_2, \ldots, z_n) = p(z_1)p(z_2|z_1)\ldots p(z_n|z_{n-1}, \ldots, z_1) = \frac{\alpha^{K_n}\prod_{k=1}^{K_n}(m_k-1)!}{\alpha(\alpha+1)\ldots(\alpha+n-1)}$$

The expected number of tables K_n given n customers is:

$$E(K_n|\alpha) = \sum_{i=1}^{n}\frac{\alpha}{i-1+\alpha} \approx \alpha\log(1+\frac{n}{\alpha}) = O(\alpha\log n)$$

In summary, CRP defines a distribution over partitions of the data points, that is, a distribution over all possible clustering structures with different number of clusters. Moreover, prior distributions can also be provided over cluster parameters, such as a Dirichlet prior over terms in LDA for each topic. A stochastic process called Dirichlet process combines the two types of priors, and thus is frequently used as the prior for mixture models, which is introduced in the following section.

3.2 Dirichlet Process

We now introduce Dirichlet process and Dirichlet process-based mixture model, the inference algorithms and applications are also briefly mentioned.

3.2.1 Overview of Dirichlet Process. Dirichlet process (DP) [33, 67, 68] is a stochastic process, which is a distribution defined on distributions. That is, if we draw a sample from a DP, it would be a distribution over values instead of a single value. In addition to CRP, which only considers the distribution over partitions of data points, DP also defines the data distribution for each cluster, with an analogy of the dishes served for each table in the Chinese restaurant metaphor. Formally, we say a stochastic process G is a Dirichlet process with base distribution H and concentration parameter α, written as $G \sim DP(\alpha, H)$, if for an arbitrary finite measurable partition $A_1, A_2, \ldots, A_r$ of the probability space of H, denoted as Θ, the following holds:

$$(G(A_1), G(A_2), \ldots, G(A_r)) \sim Dir(\alpha H(A_1), \alpha H(A_2), \ldots, \alpha H(A_r))$$

where $G(A_i)$ and $H(A_i)$ are the marginal probability of G and H over partition A_i. In other words, the marginal distribution of G must be Dirichlet distributed, and this is why it is called Dirichlet process. Intuitively, the base distribution H is the mean distribution of the DP, and the concentration parameter α can be understood as an inverse variance of the DP, namely, an larger α means a smaller variance and thus a more concentrated DP around the mean H. Notice that, although the base distribution H could be a continuous distribution, G will always be a discrete distribution, with point masses at most countably infinite. This can be understood by studying the random process of generating distribution samples ϕ_i's from G:

$$\phi_n|\phi_{n-1},\ldots,\phi_1 = \begin{cases} \phi_k^*, & \text{with probability } \frac{m_k}{n-1+\alpha} \\ \text{new draw from } H, & \text{with probability } \frac{\alpha}{n-1+\alpha} \end{cases}$$

where ϕ_k^* represents the kth *unique* distribution sampled from H, indicating the distribution for kth cluster, and ϕ_i denotes the distribution for the ith sample, which could be a distribution from existing clusters or a new distribution.

In addition to the above definition, a DP can also be defined through a stick-breaking construction [62]. On one hand, the proportion of each cluster k among all the clusters, π_k, is determined by a stick-breaking process:

$$\beta_k \sim Beta(1,\alpha) \; and \; \pi_k = \beta_k \prod_{l=1}^{k-1}(1-\beta_l)$$

Metaphorically, assuming we have a stick with length 1, we first break it at β_1 that follows a Beta distribution with parameter α, and assign it to π_1; for the remaining stick with length $1-\beta_1$, we repeat the process, break it at $\beta_2 \sim Beta(1,\alpha)$, and assign it $(\beta_2(1-\beta_1))$ to π_2; we recursively break the remaining stick and get π_3, π_4 and so on. The stick-breaking distribution over π is sometimes written as $\pi \sim GEM(\alpha)$, where the letters GEM stand for the initials of the inventors. On the other hand, for each cluster k, its distribution ϕ_k^* is sampled from H. G is then a mixture model over these distributions, $G \sim \sum_k \pi_k \delta_{\phi_k^*}$, where $\delta_{\phi_k^*}$ denotes a point mass at ϕ_k^*.

Further, hierarchical Dirichlet process (HDP) [68] can be defined, where the base distribution H follows another DP. HDP can model topics across different collections of documents, which share some common topics across different corpora but may have some special topics within each corpus.

3.2.2 Dirichlet Process Mixture Model. By using DP as priors for mixture models, we can get Dirichlet process mixture model (DPM) [48, 67, 68], which can model the number of components in a mixture model, and sometimes is also called infinite mixture model. For example, we can model infinite topics for topic modeling, infinite components in infinite Gaussian mixture model [57], and so on. In such mixture models, to sample a data value, it will first sample a distribution ϕ_i and then sample a value x_i according to the distribution ϕ_i. Formally, let $x_1, x_2, \ldots, x_n$ be n observed data points, and let $\theta_1, \theta_2, \ldots, \theta_n$ be the parameters for the distributions of latent clusters associated with each data point, where the distribution ϕ_i with parameter θ_i is drawn i.i.d from G, the generative model for x_i is then:

$$\begin{aligned} x_i|\theta_i &\sim F(\theta_i) \\ \theta_i|G &\sim G \\ G|\alpha, H &\sim DP(\alpha, H) \end{aligned}$$

where $F(\theta_i)$ is the distribution for x_i with the parameter θ_i. Notice that, since G is a discrete distribution, multiple θ_i's can share the same value.

From the generative process point of view, the observed data x_i's are generated by:

1 Sample π according to $\pi|\alpha \sim GEM(\alpha)$, namely, the stick-breaking distribution;

2 Sample the parameter θ_k^* for each distinctive cluster k according to $\theta_k|H \sim H$;

3 For each x_i,

   (a) first sample its hidden cluster label z_i by $z_i|\pi \sim \mathcal{M}(1;\pi)$,

   (b) then sample the value according to $x_i|z_i, \{\theta_k^*\} \sim F(\theta_{z_i}^*)$.

where $F(\theta_k^*)$ is the distribution of data in component k with parameter θ_k^*. That is, each x_i is generated from a mixture model with component parameters θ_k^*'s and the mixing proportion π.

3.2.3 The Learning Algorithms. As DPM is a nonparametric model with infinite number of parameters in the model, EM algorithm cannot be directly used in the inference for DPM. Instead, MCMC approaches [48] and variational inference [10] are standard inference methods for DPM.

The general goal for learning DPM is to learn the hidden cluster labels z_i's and the parameters θ_i's for its associated cluster component for all

the observed data points. It turns out that Gibbs sampling is very convenient to implement for such models especially when G is the conjugate prior for the data distribution F, as the conditional distribution of both θ_i and z_i can be easily computed and thus the posterior distribution of these parameters or hidden cluster labels can be easily simulated by the obtained Markov chain. For more details, please refer to [48], where several MCMC-based algorithms are provided and discussed.

The major disadvantages of MCMC-based algorithms are that the sampling process can be very slow and the convergence is difficult to diagnose. Therefore Blei et al. [10] proposed an alternative approach called variational inference for DPMs, which is a class of *deterministic* algorithms that convert inference problems into optimization problems. The basic idea of variational inference methods is to relax the original likelihood function or posterior probability function P into a simpler variational distribution function Q_μ, which is indexed by new free variables μ that are called variational parameters. The goal is to compute the variational parameters μ that minimizes the KL divergence between the variation distribution and the original distribution:

$$\mu^* = \arg\min D(Q_\mu||P)$$

where D refers to some distance or divergence function. And then Q_{μ^*} can be used to approximate the desired P. Please refer to [10] for more details of variational inference for DPM.

3.2.4 Applications in Text Mining. There are many successful applications in text mining by using DPMs, and we select some of the most representative ones in the following.

In [9], a hierarchical LDA model (LDA) that based on nested Chinese restaurant process is proposed, which can detect hierarchical topic models instead of topic models in a flat structure from a collection of documents. In addition, hLDA can detect the number of topics automatically, which is the number of nodes in the hierarchical tree of topics. Compared with original LDA, hLDA can detect topics with higher interpretability and has higher predictive held-out likelihood in the testing set.

In [73], a time-sensitive Dirichlet process mixture model is proposed to detect clusters from a collection of documents with time information, for example, detecting subject threads for emails. Instead of considering each document equally important, the weights of history documents are discounted in the cluster. A time-sensitive DPM (tDPM) is then built based on the idea, which can not only output the number of clusters,

but also introduce the temporal dependencies between documents, with less influence from older documents.

Evolution structure can also be detected using DPMs. A temporal Dirichlet process mixture model (TDPM) [2] is proposed as a framework to model the evolution of topics, such as retain, die out or emerge over time. In [1], an infinite dynamic topic model (iDTM) is further proposed to allow each document to be generated from multiple topics, by modeling documents in each time epoch using HDP instead of DP. An evolutional hierarchical Dirichlet process approach (EvoHDP) is proposed in [72] to detect evolutionary topics from multiple but correlated corpora, which can discover different evolving patterns of topics, including emergence, disappearance, evolution within a corpus and across different corpora.

3.3 Pitman-Yor Process

Pitman-Yor process [52, 66], also known as two-parameter Poisson-Dirichlet process, is a generalization over DP, which can successfully model data with power law [18] distributions. For example, if we want to model the distribution of all the words in a corpus, Pitman-Yor process is a better option than DP, where each word can be viewed as a table and the number of occurrences of the word can be viewed as the number of customers sitting in the table, in a restaurant metaphor.

Compared with CP, Pitman-Yor process has one more discount parameter $0 \leq d < 1$, in addition to the strength parameter $\alpha > -d$, which is written as $G \sim PY(d, \alpha, H)$, where H is the base distribution. This can be understood by studying the random process of generating distribution samples ϕ_i's from G:

$$\phi_n|\phi_{n-1}, \ldots, \phi_1 = \begin{cases} \phi_k^*, & \text{with probability } \frac{m_k - d}{n-1+\alpha} \\ \text{new draw from } H, & \text{with probability } \frac{\alpha + dK_n}{n-1+\alpha} \end{cases}$$

where ϕ_k^* is the distribution of table k, m_k is the number of customers sitting at table k, and K_n is the number of tables so far. Notice that when $d = 0$, Pitman-Yor process reduces to DP.

Two salient features of Pitman-Yor process compared with CP are: (1) given more occupied tables, the chance to have even more tables is higher; (2) tables with small occupancy number have a lower chance to get more customers. This implies that Pitman-Yor process has a power law (e.g., Zipf's law) behavior. The expected number of tables is $O(\alpha n^d)$, which has the power law form. Compared with the expected number of tables $O(\alpha \log n)$ for DP, Pitman-Yor process indicates a faster growing in the expected number of tables.

In [66], a hierarchial Pitman-Yor n-gram language model is proposed. It turns out that the proposed model has the best performance compared with the state-of-the-art methods, and has demonstrated that Bayesian approach can be competitive with the best smoothing techniques in languages modeling.

3.4 Others

There are many other stochastic processes that can be used in Bayesian nonparametric models, such as Indian buffet process [27], Beta process [69], Gaussian process [58] for infinite Gaussian mixture model, Gaussian process regression, and so on. We now briefly introduce them in the following, and the readers can refer to the references for more details.

- Indian buffet process. In mixture models, one data point can only belong to one cluster, with the probability determined by the mixing proportions. However, sometimes one data point can have multiple features. For example, a person can participate in a number of communities, all with a large strength. Indian buffet process is a stochastic process that can define the infinite-dimensional features for data points. It has a metaphor of people choosing (infinite) dishes arranged in a line in Indian buffet restaurant, which is where the name "Indian buffet process" is from.

- Beta process. As mentioned in [69], a beta process (BP) plays the role for the Indian buffet process that the Dirichlet process plays for the Chinese restaurant process. Also, a hierarchical beta process (hBP)-based method is proposed in [69] for the document classification task.

- Gaussian process. Intuitively, a Gaussian process (GP) extends a multivariate Gaussian distribution to the one with infinite dimensionality, similar to DP's role to Dirichlet distribution. Any finite subset of the random variables in a GP follows a multivariate Gaussian distribution. The applications for GP include Gaussian process regression, Gaussian process classification, and so on, which are discussed in [58].

4. Graphical Models

A Graphical model [32, 36] is a probabilistic model for which a graph denotes the conditional independence structure between random variables. Graphical model provides a simple way to visualize the structure of a probabilistic model and can be used to design and motivate new

models. In a probabilistic graphical model, each node represents a random variable, and the links express probabilistic relationships between these variables. The graph then captures the way in which the joint distribution over all of the random variables can be decomposed into a product of factors each depending only on a subset of the variables. There are two branches of graphical representations of distributions that are commonly used: *directed* and *undirected*. In this chapter, we discuss the key aspects of graphical models and their applications in text mining.

4.1 Bayesian Networks

Bayesian networks (BNs), also known as *belief networks* (or Bayes nets for short), belong to the *directed graphical models*, in which the links of the graphs have a particular directionality indicated by arrows.

4.1.1 Overview. Formally, BNs are directed acyclic graphs (DAG) whose nodes represent random variables, and edges represent conditional dependencies. For example, a link from x to y can be informally interpreted as indicating that x "causes" y.

Conditional Independence. The simplest conditional independence relationship encoded in a BN can be stated as follows: a node is conditionally independent of its non-descendants given its parents, where the parent relationship is with respect to some fixed topological ordering of the nodes. This is also called *local Markov property*, denoted by $X_v \perp\!\!\!\perp X_{V \setminus de(v)} | X_{pa(v)}$ for all $v \in V$, where $de(v)$ is the set of descendants of v. For example, as shown in Figure 8.1(a), we obtain $x_1 \perp\!\!\!\perp x_3 | x_2$.

Factorization Definition. In a BN, the joint probability of all random variables can be factored into a product of density functions for all of the nodes in the graph, conditional on their parent variables. More precisely, for a graph with n nodes (denoted as $x_1, ..., x_n$), the joint distribution is given by:

$$p(x_1, ..., x_n) = \Pi_{i=1}^{n} p(x_i | pa_i), \tag{8.1}$$

where pa_i is the set of parents of node x_i. By using the chain rule of probability, the above joint distribution can be written as a product of conditional distributions, given the topological order of these random variables:

$$p(x_1, ..., x_n) = p(x_1)p(x_2|x_1)...p(x_n|x_{n-1}, ..., x_1). \tag{8.2}$$

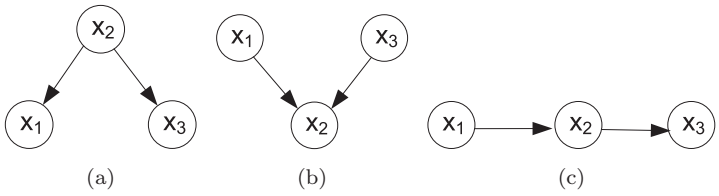


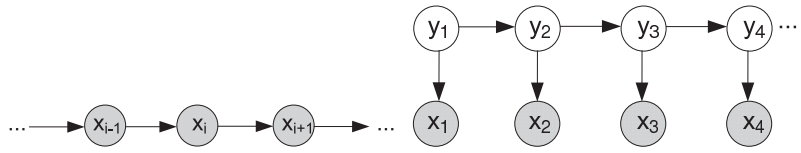
Figure 8.1. Examples of directed acyclic graphs describing the joint distributions.

The difference between the two expressions is the *conditional independence* of the variables encoded in a BN, that variables are conditionally independent of their non-descendants given the values of their parent variables.

Consider the graph shown in Figure 8.1, we can go from this graph to the corresponding representation of the joint distribution written in terms of the product of a set of conditional probability distributions, one for each node in the graph. The joint distributions for Figure 8.1(a)-(c) are therefore $p(x_1, x_2, x_3) = p(x_1|x_2)p(x_2)p(x_3|x_2)$, $p(x_1, x_2, x_3) = p(x_1)p(x_2|x_1, x_3)p(x_3)$, and $p(x_1, x_2, x_3) = p(x_1)p(x_2|x_1)p(x_3|x_2)$, respectively.

4.1.2 The Learning Algorithms. Because a BN is a complete model for the variables and their relationships, a complete joint probability distribution (JPD) over all the variables is specified for a model. Given the JPD, we can answer all possible inference queries by summing out (marginalizing) over irrelevant variables. However, the JPD has size $O(2^n)$, where n is the number of nodes, and we have assumed each node can have 2 states. Hence summing over the JPD takes exponential time. The most common *exact inference* method is **Variable Elimination** [19]. The general idea is to perform the summation to eliminate the non-observed non-query variables one by one by distributing the sum over the product. The reader can refer to [19] for more details. Instead of exact inference, a useful *approximate algorithm* called *Belief propagation* [46] is commonly used on general graphs including Bayesian network, which will be introduced in Section 4.3.

4.1.3 Applications in Text Mining. Bayesian networks have been widely used in many applications in text mining, such as spam filtering [61] and information retrieval [20]. In [61], a Bayesian approach is proposed to identify spam email by making use of a naive Bayes classifier. The intuition is that particular words have particular probabilities of occurring in spam emails and in legitimate emails. For



(a) Regular Markov model (b) Hidden Markov model

Figure 8.2. Graphical structures for the regular and hidden Markov model.

instance, the words "free" and "credit" will frequently appear in spam emails, but will seldom occur in other emails. To train the filter, the user must manually indicate whether an email is spam or not for a training set. With such a training dataset, Bayesian spam filters will learn a spam probability for each word, e.g., a high spam probability for the words "free" and "credit", and a relatively low spam probability for words such as the names of friends. Then, the email's spam probability is computed over all words in the email, and if the total exceeds a certain threshold, the filter will mark the email as a spam.

4.2 Hidden Markov Models

In a regular Markov model as Figure 8.2(a), the state x_i is directly visible to the observer, and therefore the state transition probabilities $p(x_i|x_{i-1})$ are the only parameters. Based on the Markov property, the joint distribution for a sequence of n observations under this model is given by

$$p(x_1, ..., x_n) = p(x_1) \prod_{i=2}^{n} p(x_i|x_{i-1}). \tag{8.3}$$

Thus if we use such a model to predict the next observation in a sequence, the distribution of predictions will depend on the value of the immediately preceding observation and will be independent of all earlier observations, conditional on the preceding observation.

4.2.1 Overview. A hidden Markov model (HMM) can be considered as the simplest dynamic Bayesian network. In a hidden Markov model, the state y_i is not directly visible, and only the output x_i is visible, which is dependent on the state. The hidden state space is discrete, and is assumed to consist of one of N possible values, which is also called latent variable. The observations can be either discrete or continuous, which are typically generated from a categorical distribution or a Gaussian distribution. Generally, a HMM can be considered as a

generalization of a *mixture model* where the hidden variables are related through a Markov process rather than independent of each other.

Suppose the latent variables form a first-order Markov chain as shown in Figure 8.2(b). The random variable y_t is the hidden state at time t, and the random variable x_t is the observation at time t. The arrows in the figure denote conditional dependencies. From the diagram, it is clear that y_{t-1} and y_{t+1} are independent given y_t, so that $y_{t+1} \perp\!\!\!\perp y_{t-1}|y_t$. This is the key conditional independence property, which is called the *Markov property*. Similarly, the value of the observed variable x_t only depends on the value of the hidden variable y_t. Then, the joint distribution for this model is given by

$$p(x_1, ..., x_n, y_1, ..., y_n) = p(y_1) \prod_{t=2}^{n} p(y_t|y_{t-1}) \prod_{t=1}^{n} p(x_t|y_t), \tag{8.4}$$

where $p(y_t|y_{t-1})$ is the state transition probability, and $p(x_t|y_t)$ is the observation probability.

4.2.2 The Learning Algorithms. Given a set of possible states $\Omega_Y = \{q_1, ..., q_N\}$ and a set of possible observations $\Omega_X = \{o_1, ..., o_M\}$. The parameter learning task of HMM is to find the best set of state transition probabilities $A = \{a_{ij}\}$, $a_{ij} = p(y_{t+1} = q_j|y_t = q_i)$ and observation probabilities $B = \{b_i(k)\}$, $b_i(k) = p(x_t = o_k|y_t = q_i)$ as well as the initial state distribution $\Pi = \{\pi_i\}, \pi_i = p(y_0 = q_i)$ for a set of output sequences. Let $\Lambda = \{A, B, \Pi\}$ denote the parameters for a given HMM with fixed Ω_Y and Ω_X. The task is usually to derive the *maximum likelihood estimation* of the parameters of the HMM given the set of output sequences. Usually a local maximum likelihood can be derived efficiently using the *Baum-Welch algorithm* [5], which makes use of *forward-backward algorithm* [55], and is a special case of the generalized EM algorithm [22].

Given the parameters of the model Λ, there are several typical inference problems associated with HMMs, as outlined below. One common task is to compute the probability of a particular output sequence, which requires summation over all possible state sequences: The probability of observing a sequence $X_1^T = o_1, ..., o_T$ of length T is given by $P(X_1^T|\Lambda) = \sum_{Y_1^T} P(X_1^T|Y_1^T, \Lambda)P(Y_1^T|\Lambda)$, where the sum runs over all possible hidden-node sequences $Y_1^T = y_1, ..., y_T$.

This problem can be handled efficiently using the **forward-backward algorithm**. Before we describe the algorithm, let us define the forward (alpha) values and backward (beta) values as follows: $\alpha_t(i) = P(x_1 = o_1, ..., x_t = o_t, y_t = q_i|\Lambda)$ and $\beta_t(i) = P(x_{t+1} = o_{t+1}, ..., x_T = o_T|y_t =$

$q_i, \Lambda)$. Note the forward values enable us to solve the problem through marginalizing, then we obtain

$$P(X_1^T|\Lambda) = \sum_{i=1}^{N} P(o_1, ..., o_T, y_T = q_i|\Lambda) = \sum_{i=1}^{N} \alpha_T(i).$$

The forward values can be computed efficiently with the principle of *dynamic programming*:

$$\begin{aligned} \alpha_1(i) &= \pi_i b_i(o_1), \\ \alpha_{t+1}(j) &= \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}). \end{aligned}$$

Similarly, the backward values can be computed as

$$\begin{aligned} \beta_T(i) &= 1, \\ \beta_t(i) &= \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j). \end{aligned}$$

The backward values will be used in the Baum-Welch algorithm.

Given the parameters of HMM and a particular sequence of observations, another interesting task is to compute the most likely sequence of states that could have produced the observed sequence. We can find the most likely sequence by evaluating the joint probability of both the state sequence and the observations for each case. For example, in part-of-speech (POS) tagging [37], we observe a token (word) sequence $X_1^T = o_1, ..., o_T$, and the goal of POS tagging is to find a stochastic optimal tag sequence $Y_1^T = y_1 y_2 ... y_T$ that maximizes $P(Y_1^n, X_1^n)$. In general, finding the most likely explanation for an observation sequence can be solved efficiently using the **Viterbi algorithm** [24] by the recurrence relations:

$$\begin{aligned} V_1(i) &= b_i(o_1)\pi_i, \\ V_t(j) &= b_i(o_t) \max_i \left(V_{t-1}(i) a_{ij}\right). \end{aligned}$$

Here $V_t(j)$ is the probability of the most probable state sequence responsible for the first t observations that has q_j as its final state. The Viterbi path can be retrieved by saving back pointers that remember which state $y_t = q_j$ was used in the second equation. Let $Ptr(y_t, q_i)$ be the function that returns the value of y_{t-1} used to compute $V_t(i)$ as follows:

$$\begin{aligned} y_T &= \arg\max_{q_i \in \Omega_Y} V_T(i), \\ y_{t-1} &= Ptr(y_t, q_i). \end{aligned}$$

The complexity of this algorithm is $O(T \times N^2)$, where T is the length of observed sequence and N is the number of possible states.

Now we need a method of adjusting the parameters Λ to maximize the likelihood for a given training set. The **Baum-Welch algorithm** [5] is used to find the unknown parameters of HMMs, which is a particular case of a generalized EM algorithms [22]. We start by choosing arbitrary values for the parameters, then compute the expected frequencies given the model and the observations. The expected frequencies are obtained by weighting the observed transitions by the probabilities specified in the current model. The expected frequencies obtained are then substituted for the old parameters and we iterate until there is no improvement. On each iteration we improve the probability of being observed from the model until some limiting probability is reached. This iterative procedure is guaranteed to converge to a local maximum [56].

4.2.3 Applications in Text Mining. HMM models have been applied to a wide variety of problems in information extraction and natural language processing, which have been introduced in Chapter 2, including POS tagging [37] and named entity recognition [6]. Taking POS tagging [37] as an example, each word is labeled with a tag indicating its appropriate part of speech, resulting in annotated text, such as: "[VB heat] [NN water] [IN in] [DT a] [JJ large] [NN vessel]". Given a sequence of words X_1^n, e.g., "heat water in a large vessel", the task is to assign a sequence of labels Y_1^n, e.g., "VB NN IN DT JJ NN", for the words. Based on HMM models, we can determine the sequence of labels by maximizing a joint probability distribution $p(X_1^n, Y_1^n)$.

With the success of HMMs in POS tagging, it is natural to develop a variant of an HMM for the name entity recognition task [6]. Intuitively, the locality of phenomena may indicate names in the text, such as titles like "Mr." preceding a person's name. The HMM classifier models such kinds of dependencies, and performs sequence classification by assigning each word to one of the named entity types. The states in the HMM are organized into regions, one region for each type of named entity. Within each of the regions, a statistical bi-gram language model is used to compute the likelihood of words occurring within that region (named entity type). The transition probabilities are computed by deleted interpolation, and the decoding is done through the Viterbi algorithm.

4.3 Markov Random Fields

Now we turn to another major class of graphical models that are described by undirected graphs and that again specify both a factorization and a set of conditional independence relations.

4.3.1 Overview. A Markov random field (MRF), also known as an undirected graphical model [35], has a set of nodes each of which corresponds to a variable or group of variables, as well as a set of links each of which connects a pair of nodes. The links are undirected, that is they do not carry arrows.

Conditional Independence. Given three sets of nodes, denoted A, B, and C, in an undirected graph G, if A and B are separated in G after removing a set of nodes C from G, then A and B are conditionally independent given the random variables C, denoted as $A \perp\!\!\!\perp B|C$. The conditional independence is determined by simple graph separation. In other words, a variable is conditionally independent of all other variables given its neighbors, denoted as $X_v \perp\!\!\!\perp X_{V\setminus\{v\cup ne(v)\}}|X_{ne(v)}$, where $ne(v)$ is the set of neighbors of v. In general, an MRF is similar to a Bayesian network in its representation of dependencies, and there are some differences. On one hand, an MRF can represent certain dependencies that a Bayesian network cannot (such as cyclic dependencies); on the other hand, MRF cannot represent certain dependencies that a Bayesian network can (such as induced dependencies).

Clique Factorization. As the Markov properties of an arbitrary probability distribution can be difficult to establish, a commonly used class of MRFs are those that can be factorized according to the cliques of the graph. A *clique* is defined as a subset of the nodes in a graph such that there exists a link between all pairs of nodes in the subset. In other words, the set of nodes in a clique is fully connected.

We can therefore define the factors in the decomposition of the joint distribution to be functions of the variables in the cliques. Let us denote a clique by C and the set of variables in that cliques by x_C. Then the joint distribution is written as a product of *potential functions* $\psi_C(x_C)$ over the maximal cliques of the graph

$$p(x_1, x_2, ..., x_n) = \frac{1}{Z}\Pi_C \psi_C(x_C),$$

where the *partition function* Z is a normalization constant and is given by $Z = \sum_x \Pi_C \psi_C(x_C)$. In contrast to the factors in the joint distribution for a directed graph, the potentials in an undirected graph do

not have a specific probabilistic interpretation. Therefore, how to motivate a choice of potential function for a particular application seems to be very important. One popular potential function is defined as $\psi_C(x_C) = \exp(-\epsilon(x_C))$, where $\epsilon(x_C) = -\ln \psi_C(x_C)$ is an *energy function* [45] derived from statistical physics. The underlying idea is that the probability of a physical state depends inversely on its energy. In the logarithmic representation, we have

$$p(x_1, x_2, ..., x_n) = \frac{1}{Z} \exp\left(-\sum_C \epsilon(x_C)\right).$$

The joint distribution above is defined as the product of potentials, and so the total energy is obtained by adding the energies of each of the maximal cliques.

A *log-linear model* is a Markov random field with feature functions f_k such that the joint distribution can be written as

$$p(x_1, x_2, ..., x_n) = \frac{1}{Z} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(x_{C_k})\right),$$

where $f_k(x_{C_k})$ is the function of features for the clique C_k, and λ_k is the weight vector of features. The log-linear model provides a much more compact representation for many distributions, especially when variables have large domains such as text.

4.3.2 The Learning Algorithms. In MRF, we may compute the conditional distribution of a set of nodes given values A to another set of nodes B by summing over all possible assignments to $v \notin A,\ B$, which is called *exact inference*. However, the exact inference is computationally intractable in the general case. Instead, approximation techniques such as MCMC approach [3] and loopy *belief propagation* [46, 8] are often more feasible in practice. In addition, there are some particular subclasses of MRFs that permit efficient maximum-a-posterior (MAP) estimation, or more likely assignment, inference, such as associate networks. Here we will briefly describe belief propagation algorithm.

Belief propagation is a message passing algorithm for performing inference on graphical models, including Bayesian networks and MRFs. It calculates the marginal distribution for each unobserved node, conditional on any observed nodes. Generally, belief propagation operates on a factor graph, which is a bipartite graph containing nodes corresponding to variables V and factors U, with edges between variables and the factors in which they appear. Any Bayesian network and MRF can be represented as a factor graph. The algorithm works by passing

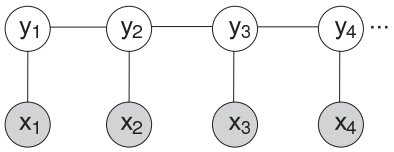


Figure 8.3. Graphical structure for the conditional random field model.

real valued function called *messages* along the edges between the nodes. Taking pairwise MRF as an example, let $m_{ij}(x_j)$ denote the message from node i to node j, and a high value of $m_{ij}(x_j)$ means that node i "believes" the marginal value $P(x_j)$ to be high. Usually the algorithm first initializes all messages to uniform or random positive values, and then updates message from i to j by considering all messages flowing into i (except for message from j) as follows:

$$m_{ij}(x_j) = \sum_{x_i} f_{ij}(x_i, x_j) \prod_{k \in ne(i) \setminus j} m_{ki}(x_i),$$

where $f_{ij}(x_i, x_j)$ is the potential function of the pairwise clique. After enough iterations, this process is likely to converge to a consensus. Once messages have converged, the marginal probabilities of all the variables can be determines by

$$p(x_i) \propto \prod_{k \in ne(i)} m_{ki}(x_i).$$

The reader can refer to [46] for more details. The main cost is the message update equation, which is $O(N^2)$ for each pair of variables (N is the number of possible states).

4.3.3 Applications in Text Mining. Recently, MRF has been widely used in many text mining tasks, such as text categorization [16] and information retrieval [44]. In [44], MRF is used to model the term dependencies using the joint distribution over queries and documents. The model allows for arbitrary text features to be incorporated as evidence. In this model, an MRF is constructed from a graph G, which consists of query nodes q_i and a document node D. The authors explore full independence, sequential dependence, and full dependence variants of the model. Then, a novel approach is developed to train the model that directly maximizes the mean average precision. The results show significant improvements are possible by modeling dependencies, especially on the larger web collections.

4.4 Conditional Random Fields

So far, we have described the Markov network representation as a joint distribution. In this subsection, we introduce one notable variant of an MRF, i.e., conditional random field (CRF) [38, 65], which is yet another popular model for sequence labeling and has been widely used in information extraction as described in Chapter 2.

4.4.1 Overview. A CRF is an undirected graph whose nodes can be divided into exactly two disjoint sets, the observed variables X and the output variables Y, which can be parameterized as a set of factors in the same way as an ordinary Markov network. The underlying idea is that of defining a conditional probability distribution $p(Y|X)$ over label sequences Y given a particular observation sequence X, rather than a joint distribution over both label and observation sequences $p(Y, X)$. The primary advantage of CRFs over HMMs is their conditional nature, resulting in the relaxation of the independence assumptions required by HMMs in order to ensure tractable inference.

Considering a linear-chain CRF with $Y = \{y_1, y_2, ..., y_n\}$ and $X = \{x_1, x_2, ..., x_n\}$ as shown in Figure 8.3, an input sequence of observed variable X represents a sequence of observations and Y represents a sequence of hidden state variables that needs to be inferred given the observations. The y_i's are structured to form a chain, with an edge between each y_i and y_{i+1}. The distribution represented by this network has the form:

$$p(y_1, y_2, ..., y_n|x_1, x_2, ..., x_n) \quad = \quad \frac{1}{Z(X)} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_i, y_{i-1}, x_i)\right),$$

where $Z(X) = \sum_{y_i} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_i, y_{i-1}, x_i)\right)$.

4.4.2 The Learning Algorithms. For general graphs, the problem of exact inference in CRFs is intractable. Basically, the inference problem for a CRF is the same as for an MRF. If the graph is a chain or a tree, as shown in Figure 8.3, message passing algorithms yield exact solutions, which are similar to the forward-backward [5, 55] and Viterbi algorithms [24] for the case of HMMs. If exact inference is not possible, generally the inference problem for a CRF can be derived using approximation techniques such as MCMC [48, 3], loopy *belief propagation* [46, 8], and so on. Similar to HMMs, the parameters are typically learned by maximizing the likelihood of training data. It can be solved using an iterative technique such as iterative scaling [38] and gradient-descent methods [63].

4.4.3 Applications in Text Mining. CRF has been applied to a wide variety of problems in natural language processing, including POS tagging [38], shallow parsing [63], and named entity recognition [40], being an alternative to the related HMMs. Based on HMM models, we can determine the sequence of labels by maximizing a joint probability distribution $p(\mathcal{X}, \mathcal{Y})$. In contrast, CRMs define a single log-linear distribution, i.e., $p(\mathcal{Y}|\mathcal{X})$, over label sequences given a particular observation sequence. The primary advantage of CRFs over HMMs is their conditional nature, resulting in the relaxation of the independence assumptions required by HMMs in order to ensure tractable inference. As expected, CRFs outperform HMMs on POS tagging and a number of real-word sequence labeling tasks [38, 40].

4.5 Other Models

Recently, there are many extensions of basic graphical models as mentioned above. Here we just briefly introduce the following two models, probabilistic relational model (PRM) [25] and Markov logic network (MLN) [59]. A Probabilistic relational model is the counterpart of a Bayesian network in statistical relational learning, which consists of relational schema, dependency structure, and local probability models. Compared with BN, PRM has some advantages and disadvantages. PRMs allow the properties of an object to depend probabilistically both on other properties of that object and on properties of related objects, while BN can only model relationships between at most one class of instances at a time. In PRM, all instances of the same class must use the same dependency mode, and it cannot distinguish two instances of the same class. In contrast, each instance in BN has its own dependency model, but cannot generalize over instances. Generally, PRMs are significantly more expressive than standard models, such as BNs. The well-known methods for learning BNs can be easily extended to learn these models.

A Markov logic network [59] is a probabilistic logic which combines first-order logic and probabilistic graphical models in a single representation. It is a first-order knowledge base with a weight attached to each formula, and can be viewed as a template for constructing Markov networks. Basically, probabilistic graphical models enable us to efficiently handle uncertainty. First-order logic enables us to compactly represent a wide variety of knowledge. From the point of view of probability, MLNs provide a compact language to specify very large Markov networks, and the ability to flexibly and modularly incorporate a wide range of domain knowledge into them. From the point of view of first-order logic, MLNs

add the ability to handle uncertainty, tolerate imperfect and contradictory knowledge, and reduce brittleness. The inference in MLNs can be performed using standard Markov network inference techniques over the minimal subset of the relevant Markov network required for answering the query. These techniques include belief propagation [46] and Gibbs sampling [23, 3].

5. Probabilistic Models with Constraints

In probabilistic models, domain knowledge is encoded into the model implicitly for most of the time. In this section, we introduce several situations that domain knowledge can be modeled as explicit constraints to the original probabilistic models.

By merely using PLSA or LDA, we may derive different topic models when the algorithms converge to different local maximums. It will be very useful if users can explicitly state which topic model they favor. A simple way to handle this issue is to list the terms that are desired by the users in each topic. For example, if "sport", "football" must be contained in Topic 1, users can indicate a related term distribution as prior distribution for Topic 1. This prior can be integrated into PLSA. Another sort of guidance is to specify which terms should have similar probabilities in one topic (must-link) and which terms should not have similar probabilities in any topic (cannot-link). This kind of prior can be modeled as Dirichlet forest prior, which is discussed in [4].

In traditional topic models, documents are considered indepedent with each other. However, in reality there could be correlations among documents. For example, linked webpages tends to be similar with each other, a paper cites another paper indicates the two papers are somehow similar. NetPLSA [41] and iTopicModel [64] are two algorithms that improve the original PLSA by consider the network constraints among the documents. NetPLSA takes the network constraints as an additional graph regularization term that forces two linked documents much similar, while iTopicModel models the network constraints using a Markov random field and also considers the direction of links in the network. These algorithms can still be solved by EM algorithm. By looking at the E-step, we can see that the constraints can be integrated into E-step, with a nice interpretation.

In [26], it proposes a framework of posterior regularization for probabilistic models. Different from traditional priors that are directly applied onto parameters, posterior regularization framework allows users to specify the constraints which are dependent on data space. For example, in an unsupervised part-of-speech tagging task, users may require

each sentence have at least one verb according to domain knowledge, which is difficult to encode as priors for model parameters only. In order to take the data-dependent constraints into consideration, a posterior regularization likelihood is proposed, which integrates both the model likelihood and the constraints. By studying several tasks with different constraint types, the new method has shown its flexibility and effectiveness.

Another line of systematical study of integrating probabilistic models and constraints are Constrained Conditional Models (CCMs) [54, 60, 13, 14]. CCM is a learning and inference framework that augments the learning of conditional models with declarative constraints. The objective function of a CCM includes two parts, one part involves the features of a task, and the other involves the penalties when constraints are violated. To keep the simplicity of the probabilistic model, complex global constraints are encoded as constraints instead of features. Usually, the inference problem given a trained model can be solved using integer linear programming. There are two strategies for the training stage, a local model that decouples learning and inference and a global model (joint learning) that optimizes the whole objective function. In practice, the local model for training is especially beneficial when joint learning for global model is computationally intractable or when training data is not available for joint learning. That is, it is more practical to train simple models using limited training data but inference with both the trained model and the global constraints at the decision stage.

6. Parallel Learning Algorithms

The efficiency of the learning algorithms is always an issue, especially for large scale of datasets, which is quite common for text data. In order to deal with such large datasets, algorithms with linear or even sub-linear time complexity are required, for which parallel learning algorithms provide a way to speed up original algorithms significantly. We now introduce several such algorithms among them.

The time complexity for original EM learning algorithm for PLSA is about linear to the total document-word occurrences in the corpus and the number of topics. By partitioning the document-word occurrence table into blocks, the calculation of the conditional probability for each term in each document can be parallelized for blocks with no conflicts. The tricky part is to partition the blocks such that the workload for each processing unit is balanced. Under this idea, [31] proposes a parallelized PLSA with 6 times' speedup on an eight-processor machine compared with the baseline. In [15], a Graphic Processing Unit

(GPU) instead of a multi-core machine is used to parallelize PLSA. GPU has a hundreds-of-core structure and high memory bandwith. It was designed to handle high-granularity graphics-related applications where many workloads can be simultaneously dispatched to processor elements, and now gradually becomes a general platform for parallel computing. In [15], both co-occurrence table-based partition and document-based partition are studied for the parallelization, which turn out to gain a significant speedup.

There are also some parallel learning algorithms for fast computing LDA. [47] proposes parallel version of algorithms based on variational EM algorithm for LDA. Two settings of implementations are considered, one is in a multiprocessor architecture and the other is in a distributed environment. In both settings, multiple threads or machines calculate E-step simultaneously for different partitions of the dataset, and a main program or a master machine will aggregate all the information and calculate M-step. In [49], parallel algorithms for LDA are based on Gibbs sampling algorithm. Two versions of algorithms, AD-LDA and HD-LDA, are proposed. AD-LDA is an approximate algorithm that applies local Gibbs sampling on each processor with periodic updates. HD-LDA is an algorithm with a theoretical guarantee to converge to Gibbs sampling using only one processor, which relies on a hierarchical Bayesian extension of the standard LDA model. Both algorithms have similar effectiveness performance as the single-processor learning, but with a significant time speedup. In PLDA [70], a further improvement is made by implementing AD-LDA on MPI (Message Passing Interface) and MapReduce, where MPI is a standardized and portable message-passing system for communicating between parallel computers, and MapReduce is a software framework introduced by Google to support distributed computing on clusters of computers.

In [17], instead of parallelizing one algorithm at a time, it proposes a broadly applicable paralleling programming method, which is easy to apply to many different learning algorithms. In the paper, it demonstrates the effectiveness of their methodology on a variety of learning algorithms by using MapReduce paradigm, which include locally weighted linear regression (LWLR), k-means, logistic regression (LR), naive Bayes (NB), SVM, ICA, PCA, gaussian discriminant analysis (GDA), EM, and backpropagation (NN).

7. Conclusions

In this chapter, we have introduced the most frequently used probabilistic models in text mining, which include mixture models with the

applications of PLSA and LDA, the nonparametric models that use stochastic processes as priors and thus can model infinite-dimensional data, the well-known graphical models including Bayesian networks, HMM, Markov random field and conditional random field. In some scenarios, it will be helpful to model user guidance as constraints to the existing probabilistic models.

The goal of learning algorithms for these probabilistic models are to find MLE, MAP estimators for parameters in these models. Most of the time, no closed form solutions can be provided. Iterative algorithms such as EM algorithm is a powerful tool to learn mixture models. In other cases, exact solutions are difficult to obtain, and sampling methods based on MCMC, belief propagation or variational inference methods are the options. When dealing with large scale of text data, parallel algorithms could be the right way to go.

References

[1] A. Ahmed and E. Xing. Timeline: A dynamic hierarchical dirichlet process model for recovering birth/death and evolution of topics in text stream. *Uncertainty in Artificial Intelligence*, 2010.

[2] A. Ahmed and E. P. Xing. Dynamic non-parametric mixture models and the recurrent chinese restaurant process: with applications to evolutionary clustering. In *SDM*, pages 219–230, 2008.

[3] C. Andrieu, N. De Freitas, A. Doucet, and M. Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1):5–43, 2003.

[4] D. Andrzejewski, X. Zhu, and M. Craven. Incorporating domain knowledge into topic modeling via dirichlet forest priors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 25–32, New York, NY, USA, 2009. ACM.

[5] L. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171, 1970.

[6] D. Bikel, R. Schwartz, and R. Weischedel. An algorithm that learns what's in a name. *Machine learning*, 34(1):211–231, 1999.

[7] J. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report TR-97-021, ICSI, 1997.

[8] C. Bishop. Pattern recognition and machine learning. Springer, New York, 2006.

[9] D. M. Blei, T. L. Griffiths, and M. I. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *J. ACM*, Aug 2009.

[10] D. M. Blei and M. I. Jordan. Variational inference for dirichlet process mixtures. *Bayesian Analysis*, 1:121–144, 2005.

[11] D. M. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[12] S. Borman. The expectation maximization algorithm: A short tutorial. *Unpublished Technical report*, 2004.
Available online at `http://www.seanborman.com/publications`.

[13] M. Chang, D. Goldwasser, D. Roth, and V. Srikumar. Discriminative learning over constrained latent representations. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, 6, 2010.

[14] M.-W. Chang, N. Rizzolo, and D. Roth. Integer linear programming in nlp – constrained conditional models. Tutorial, *NAACL*, 2010.

[15] H. Chen. Parallel implementations of probabilistic latent semantic analysis on graphic processing units. Computer science, University of Illinois at Urbana–Champaign, 2011.

[16] S. Chhabra, W. Yerazunis, and C. Siefkes. Spam filtering using a markov random field model with variable weighting schemas. In *ICDM Conference*, pages 347–350, 2004.

[17] C. T. Chu, S. K. Kim, Y. A. Lin, Y. Yu, G. R. Bradski, A. Y. Ng, and K. Olukotun. Map-Reduce for machine learning on multicore. In *NIPS*, pages 281–288, 2006.

[18] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Rev.*, 51:661–703, November 2009.

[19] F. Cozman. Generalizing variable elimination in bayesian networks. In *Workshop on Probabilistic Reasoning in Artificial Intelligence*, pages 27–32, 2000.

[20] L. de Campos, J. Fernández-Luna, and J. Huete. Bayesian networks and information retrieval: an introduction to the special issue. *Information processing & management*, 40(5):727–733, 2004.

[21] F. Dellaert. The expectation maximization algorithm. Technical report, 2002.

[22] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[23] J. R. Finkel, T. Grenager, and C. D. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, 2005.

[24] G. Forney Jr. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[25] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 1300–1309, 1999.

[26] K. Ganchev, J. A. Graça, J. Gillenwater, and B. Taskar. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049, Aug. 2010.

[27] T. Griffiths and Z. Ghahramani. Infinite latent feature models and the indian buffet process. In *NIPS*, pages 475–482, 2005.

[28] T. L. Griffiths and M. Steyvers. Finding scientific topics. *PNAS*, 101(suppl. 1):5228–5235, 2004.

[29] T. Hofmann. Probabilistic latent semantic analysis. In *Proceedings of Uncertainty in Artificial Intelligence, UAI*, 1999.

[30] T. Hofmann. Probabilistic latent semantic indexing. In *ACM SIGIR Conference*, pages 50–57, 1999.

[31] C. Hong, W. Chen, W. Zheng, J. Shan, Y. Chen, and Y. Zhang. Parallelization and characterization of probabilistic latent semantic analysis. *International Conference on Parallel Processing*, 0:628–635, 2008.

[32] M. I. Jordan. Graphical models. *Statistical Science*, 19(1):140–155, 2004.

[33] M. I. Jordan. Dirichlet processes, chinese restaurant processes and all that. *Tutorial presentation at the NIPS Conference*, 2005.

[34] C. T. Kelley. Iterative methods for optimization. *Frontiers in Applied Mathematics*, SIAM, 1999.

[35] R. Kindermann, J. Snell, and A. M. Society. *Markov random fields and their applications*. American Mathematical Society Providence, RI, 1980.

[36] D. Koller and N. Friedman. *Probabilistic graphical models*. MIT press, 2009.

[37] J. Kupiec. Robust part-of-speech tagging using a hidden markov model. *Computer Speech & Language*, 6(3):225–242, 1992.

[38] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[39] J.-M. Marin, K. L. Mengersen, and C. Robert. Bayesian modelling and inference on mixtures of distributions. In D. Dey and C. Rao, editors, *Handbook of Statistics: Volume 25*. Elsevier, 2005.

[40] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics, 2003.

[41] Q. Mei, D. Cai, D. Zhang, and C. Zhai. Topic modeling with network regularization. In *WWW Conference*, 2008.

[42] Q. Mei, X. Ling, M. Wondra, H. Su, and C. Zhai. Topic sentiment mixture: modeling facets and opinions in weblogs. In *WWW Conference*, pages 171–180, 2007.

[43] Q. Mei and C. Zhai. A mixture model for contextual text mining. In *ACM KDD Conference*, pages 649–655, 2006.

[44] D. Metzler and W. Croft. A markov random field model for term dependencies. In *ACM SIGIR Conference*, pages 472–479, 2005.

[45] T. Minka. Expectation propagation for approximate bayesian inference. In *Uncertainty in Artificial Intelligence*, volume 17, pages 362–369, 2001.

[46] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of Uncertainty in AI*, volume 9, pages 467–475, 1999.

[47] R. Nallapati, W. Cohen, and J. Lafferty. Parallelized variational em for latent dirichlet allocation: An experimental evaluation of speed and scalability. In *Proceedings of the Seventh IEEE International Conference on Data Mining Workshops*, pages 349–354, 2007.

[48] R. M. Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

[49] D. Newman, A. Asuncion, P. Smyth, and M. Welling. Distributed inference for latent dirichlet allocation. In *NIPS Conference*, 2007.

[50] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39:103–134, May 2000.

[51] P. Orbanz and Y. W. Teh. Bayesian nonparametric models. In *Encyclopedia of Machine Learning*, pages 81–89. 2010.

[52] J. Pitman and M. Yor. The Two-Parameter Poisson-Dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, 25(2):855–900, 1997.

[53] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *ACM KDD Conference*, pages 569–577, 2008.

[54] V. Punyakanok, D. Roth, W. Yih, and D. Zimak. Learning and inference over constrained output. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1124–1129, 2005.

[55] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[56] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, pages 4–15, January 1986.

[57] C. E. Rasmussen. The infinite gaussian mixture model. In *In Advances in Neural Information Processing Systems 12*, volume 12, pages 554–560, 2000.

[58] C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[59] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1):107–136, 2006.

[60] D. Roth and W. Yih. Integer linear programming inference for conditional random fields. In *International Conference on Machine Learning (ICML)*, pages 737–744, 2005.

[61] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk e-mail. In *AAAI Workshop on Learning for Text Categorization*, 1998.

[62] J. Sethuraman. A constructive definition of dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

[63] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141, 2003.

[64] Y. Sun, J. Han, J. Gao, and Y. Yu. itopicmodel: Information network-integrated topic modeling. In *ICDM*, pages 493–502, 2009.

[65] C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, pages 95–130, 2006.

[66] Y. W. Teh. A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting*

of the Association for Computational Linguistics, ACL-44, pages 985–992, 2006.

[67] Y. W. Teh. Dirichlet processes. In *Encyclopedia of Machine Learning*. Springer, 2010.

[68] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[69] R. Thibaux and M. I. Jordan. Hierarchical beta processes and the indian buffet process. *Journal of Machine Learning Research – Proceedings Track*, 2:564–571, 2007.

[70] Y. Wang, H. Bai, M. Stanton, W.-Y. Chen, and E. Y. Chang. Plda: Parallel latent dirichlet allocation for large-scale applications. In *Proceedings of the 5th International Conference on Algorithmic Aspects in Information and Management*, pages 301–314, 2009.

[71] C. Zhai, A. Velivelli, and B. Yu. A cross-collection mixture model for comparative text mining. In *ACM KDD Conference*, pages 743–748, 2004.

[72] J. Zhang, Y. Song, C. Zhang, and S. Liu. Evolutionary hierarchical dirichlet processes for multiple correlated time-varying corpora. In *ACM KDD Conference*, pages 1079–1088, New York, NY, USA, 2010. ACM.

[73] X. Zhu, Z. Ghahramani, and J. Lafferty. Time-sensitive dirichlet process mixture models. Technical report, Carnegie Mellon University, 2005.