Fast Inference for Interactive Models of Text

Jeffrey A. Lund
Brigham Young University - Provo

Fast Inference for Interactive Models of Text

Jeffrey A. Lund

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

November 2015

ABSTRACT

Fast Inference for Interactive Models of Text

Jeffrey A. Lund
Department of Computer Science, BYU
Master of Science

Probabilistic models of text are a useful tool for enabling the analysis of large collections of digital text. For example, Latent Dirichlet Allocation can quickly produce topical summaries of large collections of text documents. Many important uses cases of such models include human interaction during the inference process for these models of text. For example, the Interactive Topic Model extends Latent Dirichlet Allocation to incorporate human expertise during inference in order to produce topics which are better suited to individual user needs. However, interactive use cases of probabilistic models of text introduce new constraints on inference - the inference procedure must not only be accurate, but also fast enough to facilitate human interaction. If the inference is too slow, then the human interaction will be harmed, and the interactive aspect of the probabilistic model will be less useful. Unfortunately, the most popular inference algorithms in use today either require strong approximations which can degrade the quality of some models, or require time-consuming sampling. We explore the use of Iterated Conditional Modes, an algorithm which is able to obtain locally optimal *maximum a posteriori* estimates, as an alternative to popular inference algorithms such as Gibbs sampling or mean field variational inference. Iterated Conditional Modes algorithm is not only fast enough to facilitate human interaction, but can produce better *maximum a posteriori* estimates than sampling.

We demonstrate the superior performance of Iterated Conditional Modes on a wide variety of models. First we use a DP Mixture of Multinomials model applied to the problem of web search result cluster, and show that not only can we outperform previous methods in clustering quality, but we can achieve interactive runtimes when performing inference with Iterated Conditional Modes. We then apply Iterated Conditional Modes to the Interactive Topic Model. Not only is Iterated Conditional Modes much faster than the previous published Gibbs sampler, but we are better able to incorporate human feedback during inference, as measured by accuracy on a classification task using the resultant topic model. Finally, we utilize Iterated Conditional Modes with MomResp, a model used to aggregate multiple noisy crowdsourced data. Compared with Gibbs sampling, Iterated Conditional Modes is better able to recover ground truth labels from simulated noisy annotations, and runs orders of magnitude faster.

Keywords: Iterated Conditional Modes, Posterior Inference, Interactive Topic Modeling, Fast Inference

ACKNOWLEDGMENTS

First and foremost, I would like to think my advisor Dr. Kevin Seppi for all of his help and guidance with this work. He has been patient with me when I followed many related but fruitless tangents, and invaluable in helping me find which results I should more fully pursue. This work would not have been possible without this guidance. Dr. Eric Ringger has also been invaluable as a second committee member. His careful attention to detail while reviewing this work has sharpened the correctness of this work immensely.

Of course, the person who deserves the most gratitude is my wife. Kelli has given up the comforts that could have come with a regular salary to live the life of a graduate student's spouse. She cared for our children, proofread my attrocious writing, endured rather poor research related pillow talk, and unwaveringly supported me in my academic goals. I will forever be in her debt.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rate at which new digital text appears is rapidly increasing. The internet has enabled the layperson to easily distribute text in the form of email, blog posts, social media posts, and websites. Traditional media such as books and newspapers are also being published in digital form. Given this incredible volume of digital text, it has become impossible for humans to analyze it all without the aid of computers.

Many popular and useful approaches for analysis of large bodies of text documents use probabilistic models of text. For example, topic models such as Latent Dirichlet Allocation (hereafter LDA) [9] can automatically learn topics from a set of documents, giving users a glimpse into the common themes of the data, without the need to read each individual document. Model-based clustering techniques such as Mixture of Multinomials [40] are another kind of probablistic model which can automatically organize a collection of documents into groups based on similarity.

In this work, we will examine text analysis use cases which include user interaction. For example, the Interactive Topic Model (ITM) is a topic model which allows the user to give feedback by injecting word groupings into the model to influence model inference [23]. By including a human in the loop during inference, the topic model can be specialized to suit the particular needs of a specific task. The risk of including human interaction is that if the inference algorithm is too slow, then the delay between receiving user feedback and presenting the results from the updated model can harm the user experience and weaken the benefits of interactive modeling. Without sacrificing the accuracy of our analyses, the speed of inference will be our primary concern throughout this work.

Another example of an interactive use case is a clustering model applied to the problem of web search result clustering (WSRC). In this task, search engine results are clustered by topic before being presenting to the user [14]. Grouping the results helps the user quickly understand the search results, even when the search query is ambiguous and yields results from multiple topics. While model-based clustering is a way to produce high quality clusterings of documents [42], in order for the model to be useful in the context of web search result clustering inference must be sufficiently fast (in this case sub-second) in order to be useful for human interaction.

The purpose of this thesis is to explore the use of an inference algorithm known as Iterated Conditional Modes for training probablistic models of text in an interactive context. The remainder of this chapter introduces the problem of inference in general, including a brief description of some popular inference algorithms, including Iterated Conditional Modes. We will also briefly describe some interactive applications of probablistic models of text which are used to validate our thesis statement. Chapter 2 describes our experiments with Iterated Conditional Modes in detail. Note that chapter 2 is self-contained, as this chapter has been submitted for separate publication elsewhere. Chapter 3 expands upon chapter 2 to show the tradeoffs involved in using a hybrid approach with both Iterated Conditional Modes and more traditional sampling techniques. Finally, we conclude with some parting observations and direction for future work.

1.1 Probabilistic Inference

Many probabilistic models of text, including the ones we have already mentioned, are generative, meaning they specify a full joint probability distribution over the data and any latent variables. Consequently, they are structured so as to explain how the observed data could be generated. Typically such generative models involve latent topic or cluster variables and the words of the documents belonging to those topics. Models also include prior distributions, which allow the modeler to specify distributions over the latent variables. In practice, we infer the parameters of the model from observed data. The difficulty in learning arises from the fact that the topics or clusters are not known *a priori* and must be learned in tandem with the topic or cluster assignments themselves.

We typically conduct inference (or train) probabilistic models of text by finding the mode of the posterior probability distribution over the model parameters given the observed data. In other words, we take our *a priori* knowledge in the form of the prior distributions and attempt to find a setting of model parameters and latent topic variables which best explain the observed data. Such a mode is known as a *maximum a posteriori* estimate. In general, computing an exact *maximum a posteriori* estimate is an NP-hard problem, so we resort to various approximations in order to optimize the posterior distribution [17, 36].

There are a number of algorithms such as variational inference and Gibbs sampling which can be used to compute *maximum a posteriori* estimates for a particular model. In section 1.2, we will discuss the most popular inference algorithms for probabilistic models of text in greater detail. Each of these existing inference algorithms has some drawbacks. For example, while variational inference is often very fast, it makes simplifying assumptions about the posterior distribution which can seriously degrade the quality of solutions for certain models such as Mixture of Multinomials [39]. However, for other models such as LDA we can achieve reasonable estimates very quickly [4].

Gibbs sampling provably generates samples from the posterior distribution we are interested in. Unlike variational inference, Gibbs sampling is theoretically able to explore the entire support of the posterior manifold without approximation in the limit. However, any reasonable restriction in runtime means that we will only be able to explore a localized area of the support. Unlike most Bayesian analysis, we are not interested in characterizing the entire posterior distribution, but we instead focus on finding the mode of the posterior. Consequently, for most uses of probabilistic models of text, we will run a sampler for a period of time in the hope of finding an area of high probability, and then use the final sample as an approximation for the true mode. While for some models such as Mixture of Multinomials this technique gives very good results [34, 40, 42], the lack of convergence criteria can make the technique prohibitively slow for applications which require user interaction.

In this work, we will experiment with a less well known technique, namely Iterated Conditional Modes (ICM) [6], which is able to converge to local maxima in the posterior distribution, but requires no approximation of the posterior through sampling or through any simplifying assumptions. While we do not solve the NP-hard problem of finding the global mode of the posterior, we aim to demonstrate that ICM is a viable alternative whenever variational inference performs poorly and the lack of convergence in a Gibbs sampler is too slow.

1.2 Related Work

There are many techniques for performing inference on probabilistic models of text. In this section we will briefly introduce some of the most commonly used methods. A number of exact methods exist such as belief propagation [31] and the junction tree algorithm [24]. However, as previously mentioned, the complexity of exact inference in many models is NP-hard [36], which means that these exact algorithms do not scale well with models of large text collections. Instead, we will discuss a number of approximate algorithms for inference. The algorithms we will discuss are Expectation Maximization, variational inference, and Gibbs sampling. As previously mentioned, we can think of this problem as an optimization problem in which we seek model parameters which maximize the posterior distribution probability of the parameters given the observed text data (i.e. a *maximum a posteriori* estimate). Some of those parameters consist of latent topic variables which are used for further analysis of the text. We will discuss the strengths of each algorithm as well as the shortcomings of each algorithm which we wish to address with Iterated Conditional Modes.

1.2.1 Expectation Maximization

Expectation maximization (EM) is a popular iterative algorithm for maximizing likelihood in probabilistic models with missing or latent variables [18]. EM works in two phases. The first (expectation) step computes the expected value of the likelihood function given the value of the (hidden) parameters. The second (maximization) step uses those expectations to maximize the values of the unobserved parameters. EM alternates between these two phases until convergence is achieved.

As a hill-climbing algorithm EM is guaranteed to increase the value of the likelihood and find a local maximum. However, it is not guaranteed to find a global maximum. Furthermore, by optimizing the likelihood instead of the posterior, it only utilizes modeler provided parameters for prior distributions for smoothing. Without this extra information in the prior, for some models such as Mixture of Multinomials, it does not perform as well as other techniques which do utilize priors [34, 39, 40].

Other variants of EM also exist. Deterministic annealing EM can help overcome the problem of local maxima by performing EM on an annealed version of the likelihood distribution [37]. The annealed likelihood takes a temperature parameter which controls the amount of smoothing of the likelihood manifold. We start at a high temperature, and gradually lower the temperature according to a preset schedule. Another variant of EM is Expectation Conditional Maximization [26]. This variant interactively performs EM on conditional distributions in order to maximize the joint likelihood. For our purposes, we will focus on the original formulation of EM, as it is the most popular version for probabilistic models of text.

1.2.2 Variational Expectation Maximization

Given a posterior of the form $p(\theta|w)$, variational inference seeks to minimize the KL divergence between the true distribution p and an approximate but tractable distribution q. One popular way of choosing approximating distribution q is with the mean field assumption. Under this assumption, we fully factorize p as a product of marginals so that q has the following form: $q(\theta) = \prod_i q_i(\theta_i)$.

Using the mean field assumption, we can perform updates which function much the same as Expectation Maximization. Due to the similarity of this technique to EM, when infering latent variables it is sometimes referred to as Variational Expectation Maximization or VEM. Similarly, when this technique is used to infer parameters instead of latent variables, it is sometimes refered to as variational Bayes or VB [28]. We note that the derivation for these updates can be difficult to produce, and many models do not have published variational inference update equations. We apply these updates iteratively until convergence and are thereby able to approximately maximize the value of the posterior distribution $p(\theta|w)$.

This type of inference algorithm is very popular for models like Latent Dirichlet Allocation and is currently considered to be the state of the art when used in conjunction with hyper-parameter optimization [4]. Like EM, variational inference tends to converge very quickly. Unlike EM, it is able to take advantage of the information contained in the priors of the model. However, variational inference makes use of the mean field approximation and thereby introduces inappropriately strong assumptions of independence. Consequently, it can only maximize an approximation of the posterior and can yield very poor model quality for certain models [39].

Other variants of the general technique of variational inference also exist which do not rely on approximating the posterior by a product of marginal distributions. For example, Expectation Propagation (EP) is a form of belief propagation that approximates belief states in a Bayesian network and performs Loopy Belief Propagation with those approximated messages [27]. This type of inference can be framed as a variational problem in which we are trying to minimize KL-divergence between the true posterior and an approximate distribution, even though that approximation is no longer a globally valid joint distribution [38]. Initial results with the Latent Dirichlet Allocation model indicate that EP does not perform as well as other methods such as Gibbs sampling [22]. Other forms of variational inference include formulations which use linear programming relaxation [38] or graphcuts [10] to minimize KL-divergence. However both of these algorithms scale poorly due to their reliance on linear optimization and max-flow min-cut algorithms respectively. For the purposes of our discussion, we will limit ourselves to mean field inference as this is easily the most popular form of variational inference for probabilistic models of text.

1.2.3 Gibbs Sampling

Gibbs sampling is a Markov Chain Monte Carlo algorithm. The essential idea of Gibbs sampling is that while sampling from a posterior distribution with the form $p(\theta|w)$ may be intractable, sampling the value of a single variable θ_i while holding all other variables $\theta_{\neg i}$ fixed is often easier. Note that the complete (or full) conditional $p(\theta_i|\theta_{\neg i}, w)$ used for sampling will only exist in closed form if the prior and the posterior are conjugate distributions. However, modelers will often carefully choose distributions which satisfy this requirement. For example, many probabilistic models of text satisfy this requirement due to their reliance on the Dirichlet-multinomial conjugate pair.

The basic procedure for a Gibbs sampler is to iterate through each random variable of interest θ_i and update that value by sampling from the complete conditional $p(\theta_i|\theta_{\neg i}, w)$. During each iteration the procedure re-samples the value of every variable in the model and records the state of the variables as a sample from the entire posterior distribution. It has been shown that this procedure generates samples from the posterior in the correct proportions [21].

It is often the case in generative models of text, that there are parameters which are not as important as other variables in the model for downstream analysis. For example, we may care about the values of topic variables in a model but not the values of the parameters used to generated the topic values (in fact, we can usually recover such parameters using the topic variables themselves). Consequently, it is common to employ a collapsed Gibbs sampler (i.e., Rao-Blackwellized version of the Gibbs sampler). This sampler involves integrating out the unneeded variables, leaving only the topic variables to be sampled. It has been shown that the collapsed Gibbs sampler produces samples in the correct proportions to the marginalized posterior distribution [15].

Unfortunately, Gibbs sampling does not directly accomplish our goal of finding a *maximum a posteriori* estimate. The samples taken can serve to characterize an entire distribution, but the probability of any particular sample being a maxima in the posterior is extremely small. However, samples from the Markov chain generated by a Gibbs sampler will randomly tend towards regions of high probability. Consequently, it is often the case that selecting an arbitrary sample is a good approximation for a local maximum. Furthermore, because the random walk of the sampler is able to explore better than a maximizing technique, it can often find better solutions than EM or variational inference [34, 40], even if runtime constraints prevent the sampler from exploring more than a handful of modes in the posterior distribution.

For the purposes of interactive applications involving probabilistic models of text, Gibbs sampling has one major drawback in that it lacks clear convergence criteria. Diagnostics do exist for continuous models which can detect when the Gibbs sampler has explored a sufficient region of the distribution, but these methods require that the sampler be run much longer than is needed in order to ensure that the sampler has converged in distribution [32]. For our purposes, we desire an algorithm which can be convergent in order to reduce the runtime of inference.

1.3 Thesis Statement

An algorithm which iteratively maximizes complete conditionals, namely Iterated Conditional Modes, is able to obtain locally optimal *maximum a posteriori* estimates for various probabilistic models of text quickly enough to allow for human interaction during inference without the need for approximating the entire posterior distribution.

```
Input: observed data $w$
Output: values for unobserved variables $\theta$
    randomly initialize $\theta$
    while $\theta$ not converged:
        for $\theta_k$ in $\theta$:
            $\theta_k = \operatorname{argmax}_j p(\theta_k = j|\theta_{\neg k}, w)$
```

Figure 1.1: Iterated Conditional Modes Algorithm

1.4 Iterated Conditional Modes

In this section we will outline the Iterated Conditional Modes (ICM) algorithm. We will be brief here, as further details are given in chapter 2. Suppose we are given a probabilistic model of text with observed data w and unobserved variables θ. For the purpose of this discussion, θ may be any number of unobserved parameters and/or latent topic variables. These parameters and topic variables can be either continuous or discrete. Like Gibbs sampling, ICM relies on the fact that while computing a posterior distribution of the form $p(\theta|w)$ may be intractable, computing the complete conditional for a single variable θ_i while holding both w and the rest the parameters $\theta_{\neg i}$ may be feasible. By using the tractable complete conditional $p(\theta_i|\theta_{\neg i}, w)$ we should be able to maximize the posterior without the need to approximate.

The general procedure for ICM is similar to Gibbs sampling. As will be seen in listing 1.1, we cycle through each unobserved variable θ_i in the model and update each variable using the complete conditional for that variable. More precisely, we update the value of θ_i so that

$$\theta_i = \operatorname*{argmax}_k p(\theta_i = k|\theta_{\neg i}, w) \tag{1.1}$$

As seen in Section 2.2, we can show that this update equation monotonically increases the value of the posterior. Since the posterior is bounded above by 1, we can appeal to the monotone convergence theorem to assert that ICM must converge to a local maximum in the posterior.

1.5 Applications

In this section we will introduce some of the specific models which we will use in our experiments to validate the use of ICM in later chapters. These models represent a variety of probabilistic models applied to different interactive use cases.

1.5.1 Mixture of Multinomials

Figure 1.2 gives the conditional distributions and plate diagram for the Mixture of Multinomials model. The observed data for the model is w, where w_d represents the dth document as a vector of word counts. The number of documents is M. The latent cluster or topic variable is k, where k_d is the cluster assignment or topic of the dth document. The number of topics is K. In the model's generative story, the topic assignment is drawn from a categorical distribution with parameter λ, while the words are drawn from a multinomial distribution with parameter ϕ_{k_d}. The modeler gives input about what λ and ϕ should look like in the form of the Dirichlet priors α and β. Further information about the model is given by Walker and Ringger [40].

$$\lambda|\alpha \sim Dirichlet(\alpha), |\lambda| = K$$
$$\forall k \in \{1, ..., K\} : \phi_k|\beta \sim Dirichlet(\beta), |\phi_k| = V$$
$$\forall d \in \{1, ..., M\} : k_d|\lambda \sim Categorical(\lambda)$$
$$\forall d \in \{1, ..., M\} : w_d|k_d, \phi \sim Multinomial(N_d, \phi_{k_d})$$

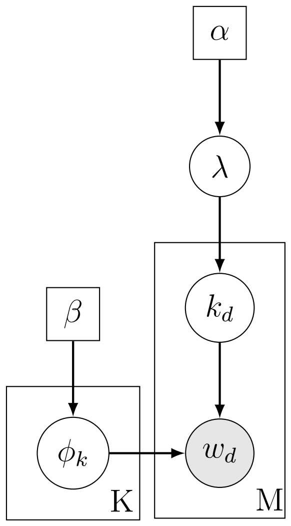


Figure 1.2: The Mixture of Multinomials model, along with the graphical representation of the model

Since the model assigns exactly one mixture component to each document, it is useful not only to gain a high level understanding of the topical content of a collection of documents but also as a method for performing both hard and soft document clustering. Model-based document clustering has become increasingly popular due to higher quality results and better scalability compared to traditional similarity-based approaches [42]. Potential applications include automatic organization of textual data, genre and novelty detection, exploratory analysis and text mining [39]. In particular, we mention web search result clustering, in which search engine results are clustered before being presented to users, either directly as clusters, or by interspersing results from different clusters to enhance diversity of the results. Web search result clustering therefore allows users to sort through different meanings of their query and focus on only the relevant results [14]. The Mixture of Multinomials model is both a simple and effective model for performing document clustering. However, in order for web search result clustering to be useful, the inference algorithm must be able to run very quickly and still produce a high quality model.

Previous work with the Mixture of Multinomials model indicates that a collapsed Gibbs sampler which integrates out the parameters λ and ϕ performs better than the uncollapsed Gibbs sampler [34]. This sampler also outperforms both Expectation Maximization and variational inference in terms of model quality [34, 39, 40]. Due to these results, our experiments with this model will focus on comparing ICM performance with that of Gibbs sampling.

In order to implement Iterated Conditional Modes for the Mixture of Multinomials model, we need to derive the update equations. Walker and Ringger [40] already gave the complete conditional for the collapsed Gibbs sampler; we can simply apply an argmax to the complete conditional for each k_d. Thus, our update equation is given by

$$k_d = \underset{j}{\operatorname{argmax}} \ (\alpha_j + n_j - 1) \frac{\Gamma(\sum_v^V \beta_v + n_{\cdot j \cdot} - n_{d \cdot \cdot})}{\Gamma(\sum_v^V \beta_v + n_{\cdot j \cdot})} \prod_v^V \frac{\Gamma(\beta_v + n_{\cdot jv})}{\Gamma(\beta_v + n_{\cdot jv} - n_{djv})} \tag{1.2}$$

where V is the number of terms in the corpus vocabulary, α and β are the parameters of the Dirichlet priors we gave to the model, n_j is the count of documents assigned to cluster j, and n_{djv} is the count of terms of type v in document d, if document d is assigned to cluster j, with dots indicating marginalization over the indicated dimension. In order to perform inference, we need only iterate over each cluster assignment k_d and update the current k_d using equation 1.2. We repeat this process until no further changes will be made to the model.

$$\forall k \in \mathbb{N} : \phi_k|\beta \sim Dirichlet(\beta), |\phi_k| = V$$
$$\forall d \in \{1, ..., M\} : k_d|\alpha \sim CRP(\alpha)$$
$$\forall d \in \{1, ..., M\} : w_d|k_d, \phi \sim Multinomial(N_d, \phi_{k_d})$$

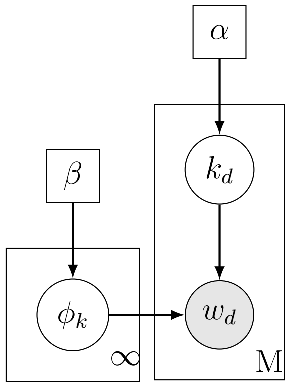
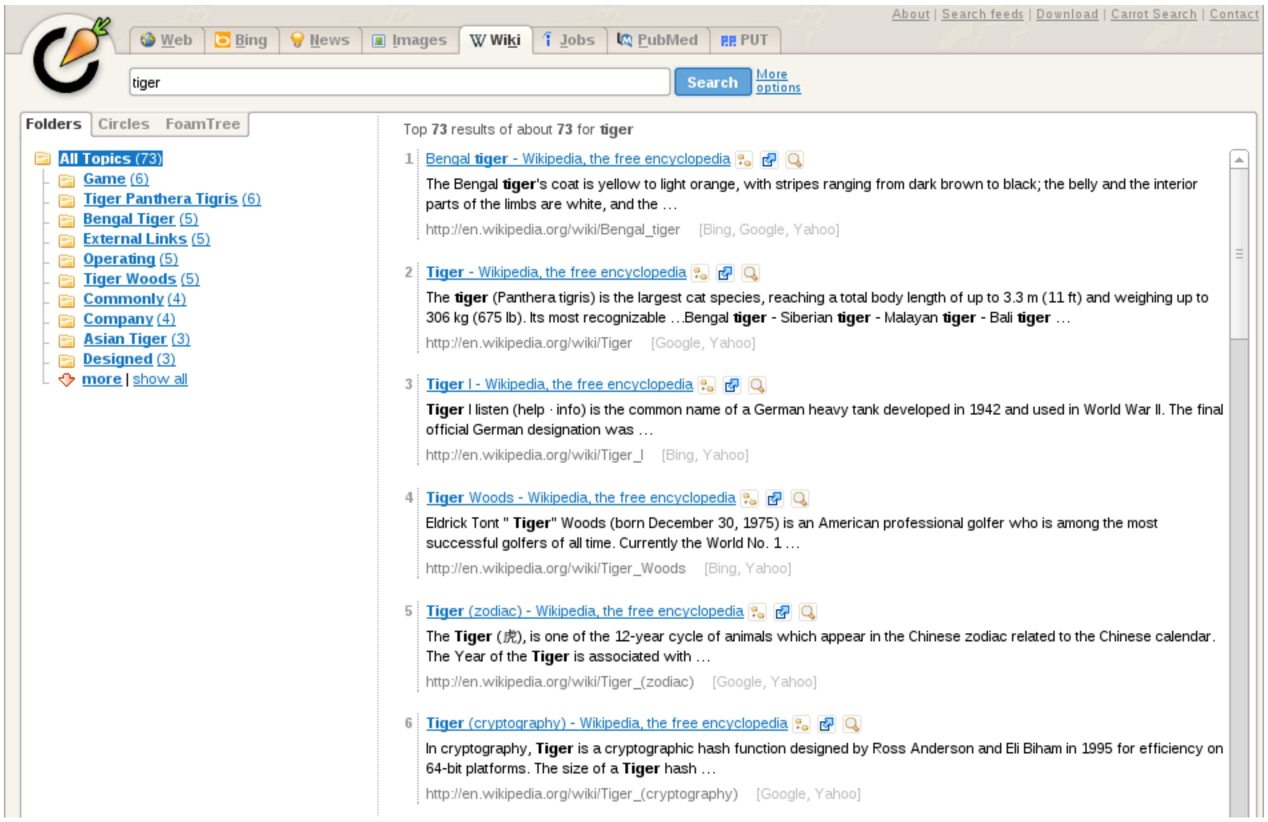


Figure 1.3: The Dirichlet Process Mixture of Multinomials model, along with the graphical representation of the model. Contrast this with figure 1.2.

One disadvantage of the Mixture of Multinomials model is that it requires the number of clusters K to be specified *a priori* . When the number of clusters is known, this is not a problem, but on new data, it can be difficult to determine the optimal number of clusters without sweeping this number with multiple runs. A simple solution to this problem is to extend the Mixture of Multinomials model to be a Dirichlet Process Mixture of Multinomials (hereafter DP-MOM). Essentially all we have to do is replace the Dirichlet prior over the topic distribution with a non-parametric Dirichlet Process prior. As observed by Blackwell and MacQueen [8], we can further simplify the model by integrating over the random mixing measure in the DP mixture, so that the resulting model will have a Chinese Restaurant Process prior over the mixing components. This non-parametric prior allows the model to learn the number of clusters from data. We will discuss this extension in more detail in subsection 2.3.1, but the basic form of the DP-MOM can be seen in figure 1.3. As part of the thesis, we implement both the Mixture of Multinomials and the DP-MOM models with a Gibbs sampler and with ICM.

While the both the Mixture of Multinomials and the DP-MOM models have been applied to document clustering in general, we will focus on a specific type of document clustering known as web search result clustering. Web search result clustering systems take the textual snippets returned from a search engine query and then use clustering on those snippets in order to automatically organize the snippets into semantically related groups. This can help the user quickly narrow the scope of an ambiguous search query.



All Topics (73)
- Game (6)
- Tiger Panthera Tigris (6)
- Bengal Tiger (5)
- External Links (5)
- Operating (5)
- Tiger Woods (5)
- Commonly (4)
- Company (4)
- Asian Tiger (3)
- Designed (3)
- more | show all

Top **73** results of about **73** for **tiger**

1 Bengal **tiger** - Wikipedia, the free encyclopedia
The Bengal **tiger**'s coat is yellow to light orange, with stripes ranging from dark brown to black; the belly and the interior parts of the limbs are white, and the …
http://en.wikipedia.org/wiki/Bengal_tiger [Bing, Google, Yahoo]

2 **Tiger** - Wikipedia, the free encyclopedia
The **tiger** (Panthera tigris) is the largest cat species, reaching a total body length of up to 3.3 m (11 ft) and weighing up to 306 kg (675 lb). Its most recognizable …Bengal **tiger** - Siberian **tiger** - Malayan **tiger** - Bali **tiger** …
http://en.wikipedia.org/wiki/Tiger [Google, Yahoo]

3 **Tiger** I - Wikipedia, the free encyclopedia
Tiger I listen (help · info) is the common name of a German heavy tank developed in 1942 and used in World War II. The final official German designation was …
http://en.wikipedia.org/wiki/Tiger_I [Bing, Yahoo]

4 **Tiger** Woods - Wikipedia, the free encyclopedia
Eldrick Tont " **Tiger**" Woods (born December 30, 1975) is an American professional golfer who is among the most successful golfers of all time. Currently the World No. 1 …
http://en.wikipedia.org/wiki/Tiger_Woods [Bing, Yahoo]

5 **Tiger** (zodiac) - Wikipedia, the free encyclopedia
The **Tiger** (虎), is one of the 12-year cycle of animals which appear in the Chinese zodiac related to the Chinese calendar. The Year of the **Tiger** is associated with …
http://en.wikipedia.org/wiki/Tiger_(zodiac) [Google, Yahoo]

6 **Tiger** (cryptography) - Wikipedia, the free encyclopedia
In cryptography, **Tiger** is a cryptographic hash function designed by Ross Anderson and Eli Biham in 1995 for efficiency on 64-bit platforms. The size of a **Tiger** hash …
http://en.wikipedia.org/wiki/Tiger_(cryptography) [Google, Yahoo]

Figure 1.4: Example screenshot from the Carrot^2 search interface. The query ”tiger” was issued to Wikipedia, and the results can be filtered by selecting a topic on the left.

One example of such a system is the Carrot^2 search framework, shown in figure 1.6. In the figure, the user has issued the query “tiger” to the web search result system. If the user only intended to find results related to the large feline, they may be surprised to see other results related to the famous golfer, the professional sports team, or the German tank. By selecting a more refined topic on the left, the user can quickly filter out irrelevant results. Other systems will instead use web search result clustering to ensure that at least one result from each topic is shown in the top results, thereby guaranteeing that the user will find a relevant result, even if they do not explicitly select a more specific topic manually.

Web search result clustering systems do not typically maintain their own search data, but instead rely on search results from an external search engine results chosen by the user. This allows the web search result clustering to work with arbitrary search results, but also introduces new constraints on runtime. Most importantly, the web search result clustering must be able to perform document clustering quickly enough to facilitate web search. For simple interactions like issuing a web search query, the interaction take less than one second [16]. We aim to show that with the use of ITM on DP-MOM model, we can not only achieve interactive runtime, but also improve the state of the art web search result clustering systems. Our experiments with these models can be found in subsection 2.3.1.

1.5.2 Interactive Topic Model

The Interactive Topic Model [23] extends Latent Dirichlet Allocation so that word groups may form constraints in the form of Dirichlet forests [1]. Figure 1.5 gives the conditional distributions and plate diagram for the model. By placing words into a single Dirichlet forest, we can encourage them to either group into the same topic or to stay in separate topics by setting the parameter η, the prior over word constraints, to the appropriate value. For our purposes, we restrict ourselves to constraints which group words, as separating constraints are much harder to handle due to issues with transitivity [1].

This model is designed to give users the power to interactively drive the model learning. First, the user is shown a summary of the topics. The user can then give feedback by selecting words which should be grouped together within the same topic. Inference then resumes using the new constraints. This process repeats iteratively until the user is satisfied with the model.

$$\forall t \in \{1,...,T\} : \phi_t|\beta,\eta \sim DirichletForest(\beta,\eta), |\phi_t| = V$$

$$\forall d \in \{1,...,M\} : \theta_d|\alpha \sim Dirichlet(\alpha), |\theta_d| = T$$

$$\forall d \in \{1,...,M\}\forall n \in \{1,...,N\} : z_{dn}|\theta_d \sim Categorical(\theta_d)$$

$$\forall d \in \{1,...,M\}\forall n \in \{1,...,N\} : w_{dn}|z_{dn},\phi \sim DirichletTree(\phi_{z_{dn}})$$

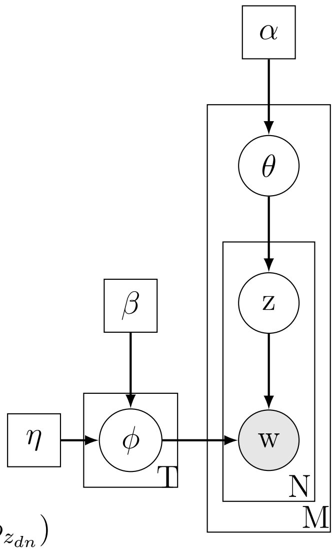
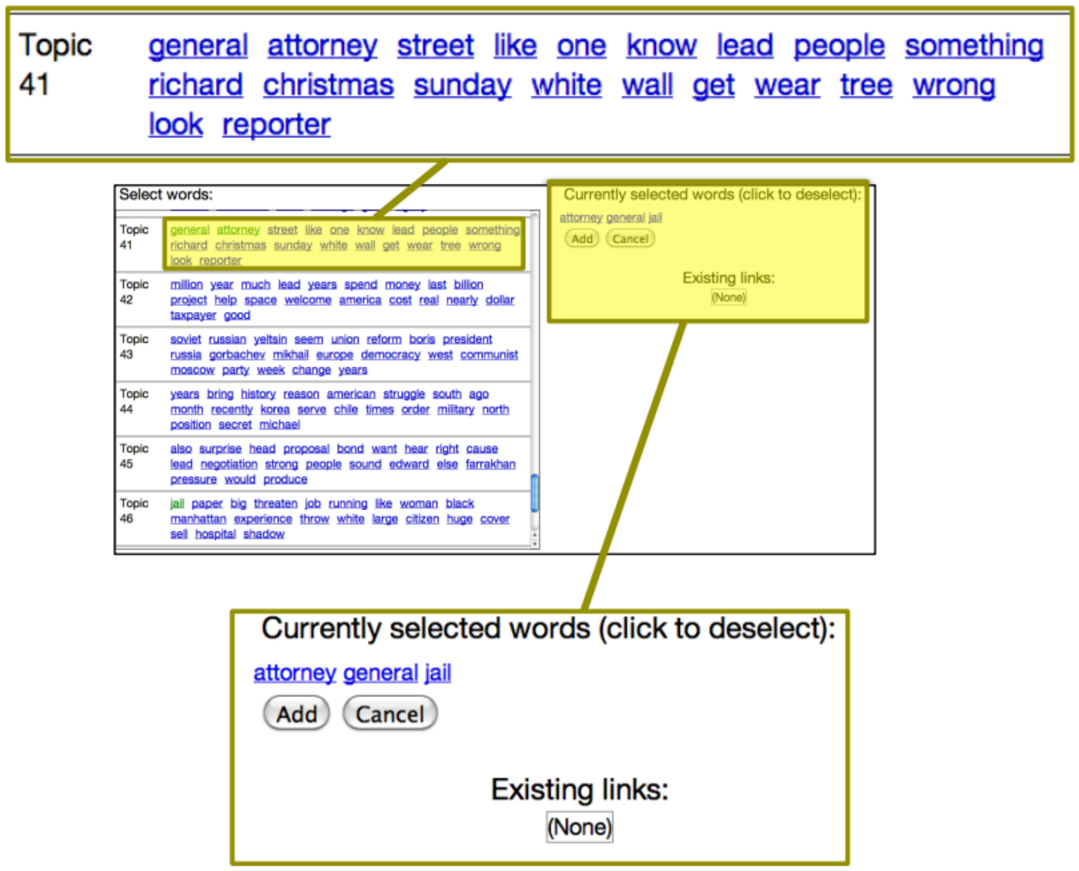


Figure 1.5: The Interactive Topic Model

By including the actual user of the topic model in the inference loop, we can produce models which are tailored to the individual user's needs. Even if we could produce a model which perfectly captures the statistical trends present in the data, each user has different purposes and intentions for the topics and the data. Allowing human guidance allows the ITM to produce models which are of higher quality for the intended analysis of the data.

We note that the run time for this application is critical for delivering a good user experience. Studies have shown that for many applications, if the user is forced to wait too long, cognitive load increases, and users can become frustrated by the experience [16]. In particular, responses such as clicking on links should take no more than a second, while more complex user-initiated actions should take no more than on the order of ten seconds to be completed. Consequently, in order to provide the best possible user experience with the Interactive Topic Model, we require an inference algorithm which can take the user constraint feedback and produce an updated model within seconds.

As with the Gibbs sampler for the ITM described by Hu et al [23], we implement a collapsed version of Iterated Conditional Modes. We also simplify the model by assuming that every token type belongs to a constraint, even if that constraint is of size one. This is


Topic 41 general attorney street like one know lead people something richard christmas sunday white wall get wear tree wrong look reporter

Select words:

Topic 41 general attorney street like one know lead people something richard christmas sunday white wall get wear tree wrong look reporter

Topic 42 million year much lead years spend money last billion project help space welcome america cost real nearly dollar taxpayer good

Topic 43 soviet russian yeltsin seem union reform boris president russia gorbachev mikhail europe democracy west communist moscow party week change years

Topic 44 years bring history reason american struggle south ago month recently korea serve chile times order military north position secret michael

Topic 45 also surprise head proposal bond want hear right cause lead negotiation strong people sound edward else farrakhan pressure would produce

Topic 46 jail paper big threaten job running like woman black manhattan experience throw white large citizen huge cover sell hospital shadow

Currently selected words (click to deselect):

attorney general jail

Existing links:

(None)

Figure 1.6: The ITM user interface for Mechanical Turk experiments in Hu et al. [23]. The user is presented with the top words of each topic, and is then allowed to introduce or remove word-level constraints.

equivalent to Hu's original formulation [23] which is conditioned on whether the token type is part of a constraint or not. The two formulations are equivalent because the terms involving word constraints are constant and do not affect the argmax. After this simplification, the update equation for ICM is:

$$z_{dn} = \underset{j}{\operatorname{argmax}} \ (\alpha_j + n_{dj\cdot\cdot}) \frac{C_{l_{dn}}\beta + n_{\cdot j l_{dn}\cdot}}{V\beta + n_{\cdot j\cdot\cdot}} \frac{\eta + n_{\cdot j l_{dn} w_{dn}}}{C_{l_{dn}}\eta + n_{\cdot j l_{dn}\cdot}} \tag{1.3}$$

where n_{dtlv} is the count of terms of type v in constraint l assigned to topic t in document d, the constraint of the nth word in the dth document is indicated by l_{dn}, and $C_{l_{dn}}$ is the number of word types in constraint l excluding the nth token of the dth document. Dots indicate marginalization over the missing index. The number of token types is V.

In order to use ICM, we need only apply this equation to each topic variable in succession until no further updates are made to the model. We implement ICM as well as the Gibbs sampler described in the original ITM paper. We will not implement variational inference, as no variational update equations have been published for the model (likely because hyper-parameter optimization is not possible in this model). Our experiments with the ITM model are presented in subsection 2.3.2 and chapter 3.

1.5.3 MomResp

The final model we will discuss in this thesis is the MomResp model, used for aggregating labels and annotations from multiple sources. It is a an joint model over document features and multiple, noisy annotations [20]. This model extends the item response model [12] by incorporating the document data to help infer ground-truth labels from crowdsourced annotations of the data. Figure 1.7 gives the conditional distributions and plate diagram for the MomResp model. The observed data for the model is w, where w_d represents the dth document as a vector of word counts. The observed annotations are a, where a_{dj} is the annotation for the dth document given by the jth annotation. The latent cluster assignment for the dth document is k_d. In the model's generative story, the cluster assignments are drawn from a categorical distribution with parameter λ, while the words are drawn from a multinomial distribution with parameter ϕ_{k_d}. Each annotation a_{dj} is drawn from a categorical distribution with parameter γ_{jk_d}. The modeler gives input about the parameters λ, ϕ, and γ in the form of Dirichlet priors α, β, and η, respectively.

Both the Gibbs sampler and ICM for the model have been implemented previously by Felt et al. [20], although the results using ICM with MomResp are as of yet unpublished. We will further discuss the results of the performance of ICM with MomResp compared to Gibbs sampling in subsection 2.3.3.

$$\lambda|\alpha \sim Dirichlet(\alpha), |\lambda| = K$$
$$\forall k \in \{1, ..., K\} : \phi_k|\beta \sim Dirichlet(\beta), |\phi_k| = V$$
$$\forall k \in \{1, ..., K\}\forall j \in \{1, ..., J\} : \gamma_{jk} \sim Dirichlet(\eta), |\gamma_{jk} = K|$$
$$\forall d \in \{1, ..., M\} : k_d|\lambda \sim Categorical(\lambda)$$
$$\forall d \in \{1, ..., M\} : w_d|k_d, \phi \sim Multinomial(N_d, \phi_{k_d})$$
$$\forall d \in \{1, ..., M\}\forall j \in \{1, ..., J\} : a_{dj}|k_d, \gamma \sim Categorical(\gamma_{jk_d})$$

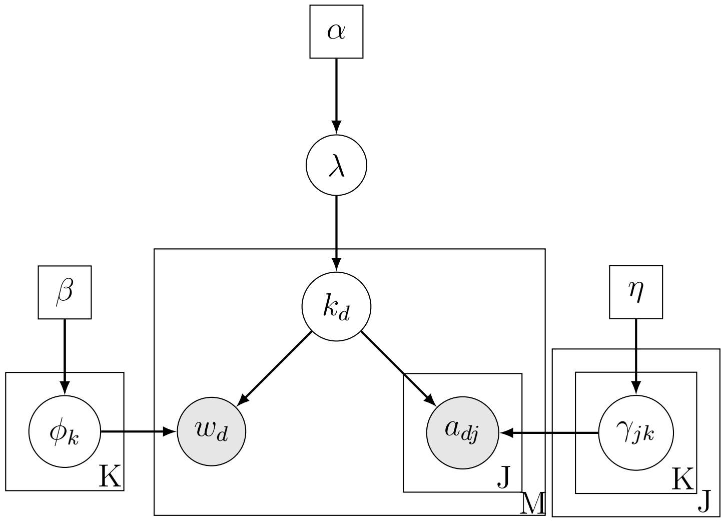


Figure 1.7: The MomResp model, along with the graphical representation of the model.

Chapter 2

Iterated Conditional Modes

Submitted for publication in Proceedings of NAACL 2015

2.1 Introduction

One of the most popular and useful approaches for analysis of large bodies of text documents is probabilistic models. For example, topic models such as Latent Dirichlet Allocation (LDA) can automatically learn topics from a set of documents, giving users a glimpse into the common themes of the data [9]. Other models such as the Mixture of Multinomials can be used to perform document clustering allowing users to automatically organize text data [25, 40].

We are interested in use cases for probabilistic models of text which include human interaction. For example, the Interactive Topic Model (ITM) is a topic model that extends LDA to allow the user to inject model constraints in the form of word groupings while the topics are being learned [23]. By including the user in the training process rather than simply learning the topics offline, the user can fine-tune the resulting topic model to better suit individual user needs and to accommodate a user's domain knowledge. However, if the training algorithm is too slow, the delay between receiving user feedback and presenting the updated model will harm the interaction due to increased cognitive load. Consequently, we require an inference algorithm which is both fast enough to facilitate interaction, and maintains (or improves upon) the accuracy of existing inference techniques.

For models like LDA, we typically perform training by calculating *maximum a posteriori* estimates of the latent topic variables and parameters given observed document data, with the idea that the setting of topic variables and parameters which maximizes the posterior distribution will best explain the observed data. Although various exact methods exist, such as belief propagation [31] and the junction tree algorithm [24], the complexity of exact posterior inference on such models is NP-HARD in general, so we resort to various approximations in order to optimize the posterior distribution [17, 36]. Some popular algorithms for approximate posterior inference include Gibbs sampling and mean field variational inference.

Each of these approximate inference algorithms has some drawbacks. For example, while variational inference is often very fast, it makes simplifying assumptions about the posterior distribution which can seriously degrade the quality of solutions for certain models, such as Mixture of Multinomials [39]. However, for other models such as LDA we can achieve good estimates very quickly [4]. Gibbs sampling provably generates samples from the posterior distribution and unlike variational inference, it is theoretically able to explore the entire support of the posterior manifold. Unfortunately, any reasonable restriction on the run time of the sampler means that we will only be able to explore a localized area of the support. Consequently, for most uses of probabilistic models of text, practitioners run a sampler for a period of time in the hope of finding an area of high probability, and then use the final sample as an approximation for the mode. While for some models, such as Mixture of Multinomials, this technique gives very good results [34, 40, 42], the lack of a convergence criteria can make the technique too slow for applications which require user interaction. For example, Griffiths and Steyvers [22] found that LDA requires hundreds of iterations of sampling before the log-likelihood of the model stabilizes in distribution. In practice, many of the probabilistic models of text we are interested in require similar numbers of sampling iterations.

As an alternative to techniques which introduce strong assumptions for posterior inference (e.g., variational inference) or lack clear and timely convergence criteria (e.g., Gibbs sampling), we will examine the use of Iterated Conditional Modes or ICM [6], This algorithm is able to quickly achieve locally optimal *maximum a posteriori* estimates.

In section 2.2, we will briefly describe the ICM algorithm and compare it with other existing techniques. Then in section 2.3 we will empirically examine the performance of ICM in the context of three very different probabilistic models of text which can be used interactively. We first show that ICM performs well in the context of a non-parametric model by experimenting with a Dirichlet Process Mixture of Multinomials applied to the problem of web search result clustering. We then turn our attention to the Interactive Topic Model [23] to show that ICM improves performance over the previously published Gibbs sampler. Finally, we use ICM in the context of MomResp, a probabilistic model designed to infer true document class labels from noisy crowdsourced judgments [20].

2.2 Iterated Conditional Modes

Suppose we are given a probabilistic model of text with observed data x and unobserved variables θ. For the purpose of this discussion, θ may represent any number of unobserved parameters and latent variables. These parameters and variables can be either continuous or discrete. Like Gibbs sampling, Iterated Conditional Modes (ICM) relies on the fact that while computing a posterior distribution of the form $p(\theta|x)$ may be intractable, computing the complete conditional for a single variable θ_i while holding fixed both x and the rest of the parameters $\theta_{\neg i}$ is feasible in models with local conjugacy. By using the tractable complete conditional distribution $p(\theta_i|\theta_{\neg i}, x)$ we are able to locally maximize the posterior without the need to approximate the posterior with samples.

The general procedure for ICM is very similar to Gibbs sampling. We cycle through each unobserved variable θ_i in the model and update current value of the variable to be the mode of its complete condition distribution. The ICM update is repeated until convergence when the value of each θ_i is already the mode of its complete conditional distribution.

To see that Iterated Conditional Modes will find a local maxima of the posterior distribution, we will demonstrate that the ICM updates monotonically increase the current estimate of the posterior probability $p(\theta|x)$. Since the data x is fixed, the posterior is proportional to the joint distribution over all of the variables and data:

$$p(\theta|x) = \frac{p(\theta, x)}{p(x)} \propto p(\theta, x) \tag{2.1}$$

Using the chain rule, for some i we can then factor the joint distribution as:

$$p(\theta, x) = p(\theta_i|\theta_{\neg i}, x) \cdot p(\theta_{\neg i}|x) \cdot p(x) \tag{2.2}$$

Since x is fixed, we can also write

$$p(\theta, x) \propto p(\theta_i|\theta_{\neg i}, x) \cdot p(\theta_{\neg i}|x) \tag{2.3}$$

While updating the parameter θ_i, $\theta_{\neg i}$ is held fixed, which means that the second term $p(\theta_{\neg i}|x)$ is constant and the factored joint distribution is proportional to the complete conditional, thus:

$$p(\theta|x) \propto p(\theta_i|\theta_{\neg i}, x) \tag{2.4}$$

Since these two expressions are proportional, setting θ_i to the mode of its complete conditional will only increase the value of the posterior probability. Thus our update rule for the variable θ_i is given as:

$$\hat{\theta}_i = \underset{k}{\operatorname{argmax}}\ p(\theta_i = k|\theta_{\neg i}, x) \tag{2.5}$$

Since this update equation monotonically increases the estimate of the posterior probability, and the value of the posterior probability is bounded above by 1, we can use the monotone convergence theorem to conclude that this algorithm will converge to a local maximum in the posterior distribution. Furthermore, the ICM algorithm is able to do so without approximating the whole posterior distribution.

Iterated Conditional Modes is related to Expectation Conditional Maximization [26] in that it employs conditionals to find local maxima in a distribution. However, unlike Expectation Conditional Maximization which maximizes a likelihood, ICM is not a variant of EM, as it takes into account prior distributions when computing the complete conditionals. In fact, we could describe ICM as a particular limit of Gibbs sampling in the same way that K-means can be viewed as the deterministic limit of the EM algorithm [7].

Note however that the efficiency of the ICM algorithm depends entirely on the ability to quickly compute the mode of the conditional distributions. If the complete conditional distribution is a density, this may involve continuous optimization. However, for many probabilistic models of text, computing the mode of the conditional distribution is often easy due to the frequent use of the Dirichlet-Multinomial conjugate pair.

To be clear, Iterated Conditional Modes is a coordinate ascent algorithm, and as such, it can only locally optimize the posterior — there is no guarantee of finding a global maximum. Thus if the posterior manifold contains many sub-par local maxima, then ICM will be sensitive to initialization and may perform poorly. Random restarts may mitigate the problem. Alternatively, an initialization strategy which consistently starts the inference procedure near a good solution will yield better *maximum a posteriori* estimates. The best initialization strategy depends on the model to be optimized. Thus each experiment described below includes its own initialization strategy.

2.3 Experiments

We now demonstrate that Iterated Conditional Modes converges quickly enough to allow for interactive use cases involving various probabilistic models of text, while yielding high quality estimates. In the hopes of demonstrating the general applicability of the technique, we do so on three different models and tasks. The first model is a DP Mixture of Multinomials applied to the task of web search result clustering. We choose this model to show that ICM can work in the context of non-parametric models. The second task is the Interactive Topic Model or ITM. We choose this model to suggest that ICM may be viable for a wide variety of interactive topic modeling applications. Finally, we will apply ICM to the MomResp model, which is a probabilistic model for producing annotated corpora for NLP and machine learning research.

2.3.1 Web Search Result Clustering

As many as 16% of queries issued to search engines contain ambiguous search terms [35]. After issuing a search query with this kind of ambiguity, users may become confused by seemingly unrelated search results, or they may be slowed by the need to narrow the scope of the query. For example, suppose a user issues the query "tiger." The user may be surprised to see results about the large feline, the golfer Tiger Woods and the German tanks used in the 1940s, when only one of those meanings of "tiger" was intended. Web search result clustering helps users deal with query ambiguity by automatically discovering clusters among the search results and presenting the results as clusters [14]. With a web search result clustering system, the user can select the cluster in which they are actually interested, and immediately filter out irrelevant results. An example of such a system is the Carrot^2 search framework, which is available online both as a web service and as a downloadable application.[*]

Client-side web search result clustering systems do not maintain their own search index or data but instead rely on search results (specifically the snippets) returned from an external search engine chosen by the user. This allows users to utilize web search result clustering systems on a wide variety of search engines, both public and private. Since web search result clustering systems are meant to work with arbitrary search results, the computation typically takes place client-side. An important consequence is that web search result clustering systems should be able cluster extremely small amounts of data: rather than tens of thousands of full documents encountered in typical document clustering tasks, a web search result clustering

[*]http://search.carrot2.org

systems uses around 100 documents, each of which consist of no more than a sentence or two. Furthermore, web search result clustering systems must be run quickly enough to facilitate web search. For simple interactions like issuing a web search query, the interaction take less than one second [16]. This stands in contrast to typical document clustering settings which can be run offline possibly using parallel computation resources rather than online using a single commodity machine. Due to these run time constraints, it has been argued that traditional document clustering techniques may not work out of the box [14]. Consequently, various specialized algorithms for the problem of web search result clustering have been published.

The best reported solution to the problem of web search result clustering employs maximal spanning trees to perform word sense induction [19]. The algorithm, referred to as MST, uses the Google Web1T n-gram data set [11] to create a co-occurrence graph on the words in the snippet results and then calculates maximal spanning trees to remove edges from the graph. This process repeats until the desired number of word clusters is formed. Unfortunately, the requirement of large amounts of n-gram data is not amenable to client-side computation. Even just maintaining an up-to-date n-gram data set (so that the system can handle queries related to fast-changing subjects such as recent popular culture) is also necessary but requires web-scale data-gathering resources. Consequently, MST is not a client-side solution.

There are, however, a number of approaches which are amenable to client-side web search result clustering. One such system is Lingo, which was developed for use in the Carrot^2 search framework [30]. Another is KeySRC, which extracts key phrases from snippet data and then uses hierarchical agglomerative clustering on those phrases [5].

Despite the fact that model-based approaches tend to yield higher quality results in the general problem of document clustering [42], no study has applied model-based clustering to the specialized problem of web search result clustering. We rectify the lacuna by applying Iterated Conditional Modes to a Dirichlet Process Mixture of Multinomials model (hereafter referred to as DP-MOM), and we compare our results to those of the previously studied web search result clustering solutions.

Our model-based approach employs a Dirichlet Process (DP) mixture model, a well studied Bayesian non-parametric model [3, 29]. This type of model has been used to perform document clustering, albeit with modifications to include feature selection in the model [41]. Given the scarcity of data in this application, we cannot realistically perform feature selection (although the snippet generation itself might be viewed as feature selection).

While DP mixtures in general are well known, the details for the specific instance of a DP mixture model that we employ (namely DP-MOM) have not been published elsewhere, so we will elaborate here. DP mixture models have a known relationship with the Chinese Restaurant Process (CRP) in that if we integrate over the random mixing measure in the DP mixture, the resulting model will have a CRP prior over the mixture components [8]. Taking advantage of this relationship, the DP-MOM model can be written with the following form:

$$\phi_k|\beta \sim Dirichlet(\beta), k = 1, ..., K$$

$$z_d|\alpha \sim CRP(\alpha), d = 1, ..., M$$

$$w_d|z_d, \phi \sim Multinomial(N_d, \phi_{z_d}), d = 1, ..., M$$

where ϕ_k is the word distribution for topic k with a symmetric Dirichlet prior of β, z_d is the cluster assignment of document d, α is CRP concentration, and w_d gives the observed token counts for document d. M is the number of documents, and N_d is the number of tokens in the dth document.

Following the advice of Neal [29], we derive a collapsed Gibbs sampler by integrating over ϕ. The complete conditional probability for the cluster assignment z_d, given the other assignments $z_{\neg d}$ and data is

$$p(z_d = j | z_{\neg d}, w) \propto \begin{cases} c_j \prod_{v \in V} \frac{\Gamma(\beta + n_{jv} + w_{dv})}{\Gamma(|V|\beta + n_{j\cdot} + w_{d\cdot})} \frac{\Gamma(|V|\beta + n_{j\cdot})}{\Gamma(\beta + n_{jv})}, & \text{if } c_j > 0 \\ \alpha \prod_{v \in V} \frac{\Gamma(\beta + w_{dv})}{\Gamma(|V|\beta + w_{d\cdot})} \frac{\Gamma(|V|\beta)}{\Gamma(\beta)}, & \text{otherwise} \end{cases} \quad (2.6)$$

where V is the set of words in the data, c_j is the count of documents assigned to cluster j, n_{jv} is the number of times the word type v is present in a document assigned to cluster j, and w_{dv} is the number of times word v is found in document d. Dots in the subscripts of these counters indicate marginalization over the missing index. For the sake of space, we omit the derivation, but it is similar to the derivation for the finite Mixture of Multinomials given by Walker and Ringger [40].

The final detail needed for implementing DP-MOM is an initialization strategy. A key advantage of using a non-parametric model is that the model can learn the number of clusters from data, thereby allowing our model to perform well with varying amounts of query term ambiguity. We can either initialize with a large number of clusters and let the model shrink to fit the data or to start with a small number of clusters and grow to fit the data. Our experiments indicate that starting with a single cluster performed the best, so we utilized this initialization strategy for our results.

In order to validate that our model-based approach performs well, we follow the same methodology as Di Marco and Navigli [19] when evaluating the MST algorithm. We experiment with two different datasets: AMBIENT [13] and MORESQUE [19]. Each dataset is a set of search queries issued to the Yahoo! search engine, along with the top 100 search result snippets which have all been manually labeled with topics. The primary difference between the two datasets is that the earlier AMBIENT dataset consists of single word queries, while the MORESQUE dataset extends AMBIENT to queries of length 2–4. Taking both datasets together, we have a total of 158 ambiguous queries, each with between 3 and 15 topics.

Algorithm	AMBIENT	MORESQUE	All
ICM DP-MOM	**.768**	**.570**	**.625**
Gibbs DP-MOM	.758	.544	.604
KEYSRC	.665	.558	.588
Lingo	.628	.527	.555
MST	<u>.815</u>	<u>.867</u>	<u>.852</u>

Table 2.1: Clustering quality results, as measured by the Rand index. We include the MST results for reference though they do not constitute client-side results. Bold indicates the best client-side result, and underline indicates the absolute best result.

Following Di Marco and Navigli [19], we evaluate clustering performance on these ambiguous query datasets with two metrics. The first is the Rand Index [33], a measure of similarity between two clusterings over the same set of elements. The Rand Index can be viewed a kind of accuracy, since it gives the percentage of pairing decisions which were correctly made with respect to a base clustering.

Table 2.1 summarizes the results of the various web search clustering algorithms measured by Rand Index. We see that the MST algorithm performs the best, but we remind the reader that this algorithm far exceeds the computation resource requirements for client-side web search result clustering so it only serves as a baseline. Among previously studied algorithms which respect resource constraints, our model-based approach outperforms existing techniques by a wide margin. Interestingly, Iterated Conditional Modes outperforms Gibbs sampling, indicating that for this model and this task, the extra exploration within a region of high probability from sampling is not as important as the ability to jump to a mode in that region of high probability. Furthermore, our approach is extremely fast. Using a single core of an AMD Phenom II X6 1090T processor, the median time spent using ICM to perform clustering on results for a single query was 1.18 milliseconds.

Our second measure evaluates the diversification produced by a clustering algorithm. As outlined by Di Marco and Navigli [19], we can use the clustering labels to re-rank the search results such that the top search results are more diverse. We measure the diversification

Algorithm	K=3	K=5	K=10	K=15	K=20
ICM DP-MOM	**.517**	**.650**	**<u>.811</u>**	**<u>.885</u>**	**<u>.927</u>**
Gibbs DP-MOM	.508	.635	.795	.873	.917
Yahoo!	.492	.600	.729	.785	.827
KeySRC	.443	.558	.720	.791	.832
MST	<u>.547</u>	<u>.656</u>	.792	.867	.907

Table 2.2: Diversification results on all queries, as measured by S-recall@K. Bold indicate the best client-side result, and underline indicates the best absolute result.

with S-recall@K, which measures the percentage of ground-truth labeled topics present in the top K search results after re-ranking.

Table 2.2 summarizes the results of the various web search result clustering algorithms with respect to S-recall@K. Once again ICM outperforms Gibbs sampling. In both cases, our model-based approach outperforms the baseline ranking taken directly from the Yahoo! search results. Both KeySRC and Lingo actually did worse than the Yahoo! baseline. For $K \leq 5$, MST performs the best. However, for all other values of K, our model-based approach performs the best. This is likely due to the fact that our non-parametric model is able to increase the number of clusters in the presence of highly ambiguous queries, whereas the MST algorithm uses a pre-specified number of clusters.

2.3.2 Interactive Topic Model

We now turn our attention to the Interactive Topic Model or ITM [23]. This model extends LDA by replacing the per-topic categorical distributions over words with a tree-structured Dirichlet-forest distribution. The user interactively injects constraints into the model by placing token types into Dirichlet trees. Depending on the prior for the Dirichlet trees, the constraint can either be a “must link” (positive correlation) or a “cannot link” (negative correlation) type of constraint [2]. Due to issues with transitivity in “cannot link” constraints, we follow Hu et al. [23] and focus on “must link” constraints by setting the Dirichlet trees parameter to be extremely high. A user is able to employ constraints to tell the model to give a particular set of word types similar probability within each individual topic. For the sake of brevity, we omit details about the ITM including the specific distributions for the model and the complete conditionals used to drive the collapsed Gibbs sampler for the model since they are thoroughly explained by Hu et al. [23].

Following Hu et al. [23], the ITM model is trained as follows: first we train a base model with no constraints. This is equivalent to learning a vanilla LDA model. The user is then presented with the outcome of an analysis using the model, possibly by showing them the traditional topic lists wherein a topic is represented by the most probable words in the topic. The user then injects word constraints into the model according to the individual needs of the user or specific domain knowledge. Using the document-level ablation strategy recommended by Hu et al. [23] the topic assignments of any document which contains a newly constrained word are revoked. In order to enforce model consistency, the rest of the topic assignments remain unchanged. Finally, inference is rerun with the new constraints and the updated model is presented to the user. This interactive process is repeated until the user is satisfied with the final state of the model.

We now investigate which inference algorithm performs the interactive model updates best. As demonstrated by Hu et al. [23], Gibbs sampling is certainly an option. Cook and Thomas [16] show that for a complex user-initiated activity such as requesting a model update, we desire a response time which is ten seconds or less or else the human-computer interaction may suffer. Unfortunately, if we perform the recommended 30 iterations of sampling, even at one second per iteration, this can be taxing on the user. Thus we turn to ICM as a faster alternative.

In order to validate the performance of Iterated Conditional Modes on the ITM, we employ an experimental setup similar to that of Hu et al. [23] using the well-known 20 Newsgroups corpus, which consists of roughly 20,000 documents divided into 20 newsgroups. We simulate a user's constraints by selecting words using information gain with respect to the newsgroup labels. After training a base model with 100 iterations of Gibbs sampling for burn-in, we inject the simulated constraints into the model. Finally, we run inference using either

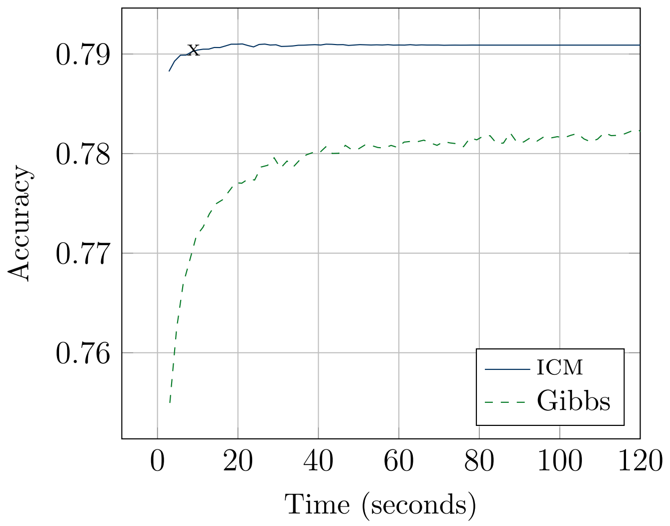


Figure 2.1: Accuracy versus time for both Iterated Conditional Modes and Gibbs sampling with the ITM. The X-mark indicates the median point of convergence for ICM.

ICM or Gibbs sampling from this point. We evaluate the model quality with a classification task. To do so, the corpus is split into a training and test set, and each word along with its assigned topic is used as a feature for the classifier. We report the classification accuracy from a support vector machine trained on the topic-word pairs from the documents in the training set. As with Hu et al. [23], we do not hope to achieve state-of-the-art classification results for this dataset, but we do hope that the classification trends will demonstrate which inference algorithm better drives the model towards the original (withheld) human labels once the simulated constraints have been added.

As shown in Figure 2.1 Iterated Conditional Modes outperforms Gibbs sampling. This indicates that there is more value in reaching a local maximum than there is in the exploration that comes from sampling. More importantly, ICM has the potential to run much faster than Gibbs sampling: rather than running a Gibbs sampler for the recommended 30 iterations, the median number of iterations required for ICM to converge was 9, which allows us to present the updated model to the user within the ten second time frame recommended by Cook and Thomas [16].

2.3.3 MomResp

Microtask markets such as Amazon's Mechanical Turk (mturk.com) allow corpora to be labeled at extremely low cost, a practice known as crowdsourcing. However, the recent emergence of crowdsourcing as the preferred method for labeling document corpora has introduced an important research problem: how to mitigate the inaccuracy of crowdsourced judgments. A common solution is to obtain multiple redundant judgments, or annotations, and aggregate them using a baseline strategy such as *majority vote*.

When annotations are both plentiful and highly accurate, majority vote works well. However, crowdsourced annotations are seldom highly accurate. State-of-the-art solutions are model-based and use standard inference algorithms. For example, the MomResp model presented by Felt et al. [20] describes a joint model over document features and annotations. We include here a sketch of the model, deferring details to the referenced paper. Documents and annotations are both modeled as count vectors with multinomial distributions conditioned on the true but unobserved class label. Parameters include both per-class word distributions and class confusion matrices for each annotator. When annotations are scarce or of low-quality, the MomResp model trained with Gibbs sampling significantly outperforms majority vote in terms of inferred label accuracy. Labels inferred by Iterated Conditional Modes are even more accurate.

In order to validate this claim, we run MomResp with both Gibbs and ICM on synthetic annotations produced for the 20 newsgroups dataset. We draw synthetic annotators from the LOW and CONFLICT annotator pools described by Felt et al. [20]. Each pool consists of 5 annotators. In both pools, annotators give correct judgments with probabilities .5, .4, .3, .2, .1, respectively. In the LOW pool, annotator errors are distributed uniformly across incorrect classes. In the simulated CONFLICT pool, errors are systematic: a confusion matrix is created for each annotator whose diagonal is set to the annotator's accuracy and whose off-diagonal row entries are sampled from a symmetric Dirichlet distribution with parameter 0.1, to encourage sparsity, and then scaled so that each row sums to 1. CONFLICT

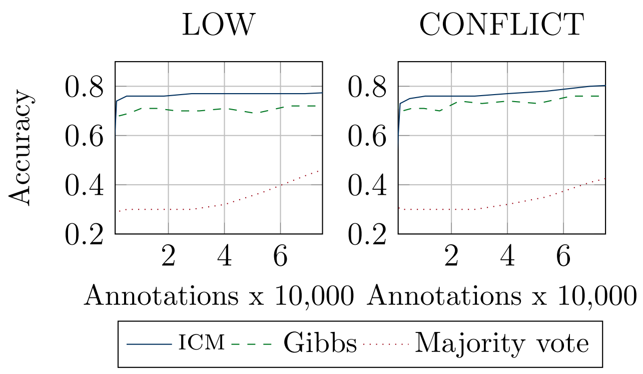


Figure 2.2: Accuracy of inferred labels versus the number of annotations given to the model. At the last plotted point each document has on average nearly 4 annotations. Gibbs and ICM use MomResp. A majority vote baseline is also shown for reference.

errors are produced by corrupting true labels according to this confusion matrix. Documents are annotated in random order without replacement, and after all documents have one annotation, the process is repeated. Simulated annotation continues until we have reached the desired number of annotations. We then initialize MomResp using majority vote to set initial class label values and perform posterior inference using both Gibbs and ICM. We compare the model-inferred class labels with the gold standard class labels for each document in order to compute model accuracy.

Figure 2.2 plots the inferred label accuracy of Gibbs and ICM as well as majority vote for reference. Regardless of the number of annotations, ICM yields better accuracy than Gibbs sampling for both the LOW and the CONFLICT cases. While not shown, this trend hold even cases where majority vote outperforms MomResp.

In addition to inferring more accurate document labels than Gibbs, ICM has a run time which is orders of magnitude faster than that of Gibbs sampling. The median time of convergence was 6.72 seconds. This falls well within the run time recommended by Cook and Thomas [16] for complex user-initiated tasks such as rerunning inference on MomResp given additional annotations. Consequently, if MomResp were to be adapted for an active learning task, then ICM would provide not only accurate posterior inference, but run times which are amenable to active learning.

2.4 Conclusion

Iterated Conditional Modes is a coordinate ascent algorithm that yields locally optimal *maximum a posteriori* estimates for models with tractable complete conditionals. We have shown that ICM identifies *maximum a posteriori* solutions that are superior to those found by Gibbs sampling for three applications: web search result clustering, topic modeling, and crowdsourcing problems. In addition, Iterated Conditional Modes has a termination criterion which is easily identified while it is often difficult to determine when a Gibbs sampler has reached the stationary distribution. Because of the convergence of ICM, we were able to significantly speed up the three applications compared to Gibbs sampling, enabling better human interactivity. These experiments motivate further exploration of this inference technique, particularly in interactive use cases of probabilistic models in which both run time and model quality are crucial.

Chapter 3

Hybrid Maximization/Sampling for the Interactive Topic Model

In subsection 2.3.2, we explored the use of Iterated Conditional Modes with the ITM model and found that ICM can significantly improve upon Gibbs sampling both in terms of runtimes and classification accuracy. One way to reason about the difference in performance in these two algorithms is by framing the inference problem in terms of the tradeoff between exploitation and exploration. With ICM, we are able to exploit the current state of the inference by using hill-climbing to quickly find a local maximum. On the other hand, Gibbs sampling is theoretically able to explore a signicant portion of the posterior manifold through the sampling process. Therefore, given enough time and assuming that the maximal sampled state is tracked, Gibbs sampling will eventually be able to achieve globally optimal *maximum a posteriori* estimates. However, given the limitations on time in the context of interactive use of the ITM model, we find that immediate exploitation is more important.

In this chapter, we will expand upon our previous results in order to better understand the tradeoff between exploration and exploitation during ITM inference. To this end, we will introduce a number of hybrid algorithms which allow us to control the amount of sampling and maximization which occurs during the posterior inference process. We will then utilize an experimental setup similar to 2.3.2 and compare the results of these hybrid algorithms to that of ICM and Gibbs sampling.

3.1 Hybrid Experiments

The hybrid algorithms we will experiment with are COINFLIP, TOPN, and THRESHOLD. Each of these algorithms includes a parameter which allows us to control the degree of exploration and exploitation. As in 2.3.2, we are most interested in characterizing algorithms when the runtime is constrained to enable human interaction. For a complex user-initiated action such as updating the ITM after injecting a topic-word constraint, we require a response time on the order of ten seconds [16].

3.1.1 Coinflip

The first hybrid algorithm we will experiment with is COINFLIP. Like ICM and Gibbs sampling, we iterate through each latent topic variable in the model, and we update the value of that variable. With some probability p, we will update the value of the variable by maximizing the complete conditional for that variable. Otherwise, we update the variable by sampling from the complete conditional. By setting the value of p appropriately, we can control the degree of exploration or exploitation. Note that $p = 0$ is equivalent to Gibbs sampling, and $p = 1$ is equivalent to ICM.

We run our experimental setup (described in 2.3.2) for each value of p in $[0, .25, .5, .75, 1]$. The results of these experiments are summarized in figure 3.1. Within the runtime needed for interactivity, the general trend is that as we increase the amount of exploitation (through maximization), classification accuracy also increases. However, if we allow for increased runtimes, we see that that a hybrid approach eventually yields accuracy superior to that of ICM. In particular, after 40 seconds, we see that COINFLIP with $p = .75$ actually surpasses ICM. Unfortunately, the increased runtime required for COINFLIP to produce superior solutions is not amenable to interactive use cases.

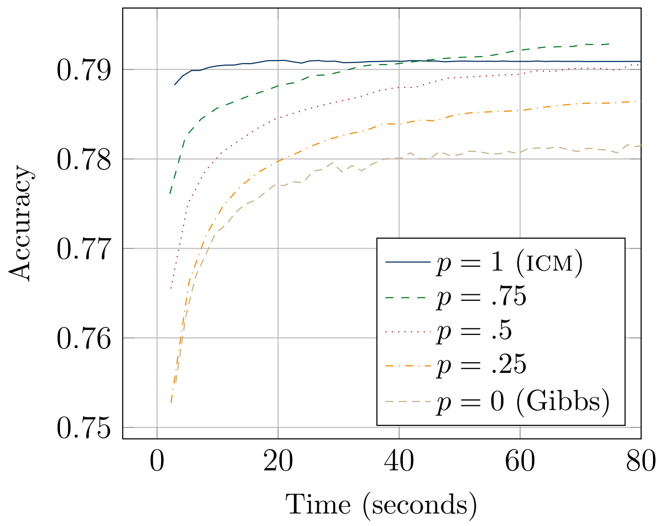


Figure 3.1: Accuracy versus time for the COINFLIP algorithm, with various values for p. Note that for $p = 0$, COINFLIP is equivalent to Gibbs sampling, and that for $p = 1$ COINFLIP is equivalent to ICM. In particular, with enough time, p=0.75 yields the best accuracy.

3.1.2 Top N

The next hybrid inference algorithm for ITM we will use is TopN. Once again we iterate through each latent topic variable in the model and perform an update using a sample from the complete conditional for a particular topic assignment. However, we reject any assignment which is not in the top n assignments when ranked by mass. By setting the value of n appropriately, we can control the amount of exploration during inference. If n is low, we discourage exploration by discarding topic assignments which have low probability. Note that if $n = 1$, then TopN is equivalent to ICM. Alternately, if n is equal to the number of topics in our ITM model, then TopN will be equivalent to Gibbs sampling.

The results of our experiments with ITM using TopN for inference can be seen in Figure 3.2. For values of $2 < n < 20$, the TopN algorithm performs worse than Gibbs sampling. This poor performance can be explained by the fact that when the TopN algorithm performs exploration by sampling from the top n possible topic assignments, it can reinforce the same incorrect assignment by making this assignment multiple times. For example, consider the case where $n = 3$, and for each token in a particular document, the conditional

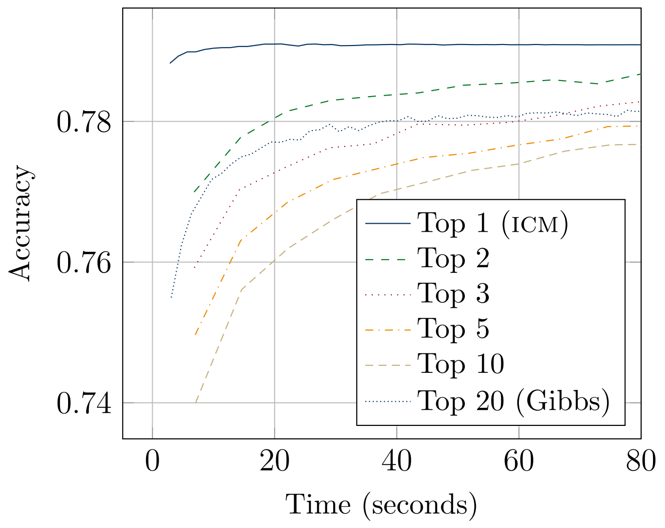


Figure 3.2: Accuracy versus time for the TopN algorithm, with various values for n. Note that for $n = 1$, TopN is equivalent to ICM, and that for $n = 20$, TopN is equivalent to Gibbs sampling.

distribution over topics has the same rank order. Since the topic ranking is the same across this entire document (and possibly others), each time exploration is performed, the same suboptimal topic assignment will be chosen. The next time the documents with this particular topic ranking are updated, the potentially harmful topic assignment will be even more likely. Contrast this behavior with that of COINFLIP, which when exploring allows any possible topic assignment (even if some are far less likely than others).

3.1.3 Threshold

The final hybrid method we will employ is THRESHOLD. This algorithm is similar to Gibbs sampling, except that the sampler will reject any assignment for which the conditional probability of that assignment is less than a threshold t. If no such assignment exists, then we simply select the assignment which maximizes the conditional probability. We can use the value of t to control the amount of exploration or exploitation in the algorithm. If t is low, then we allow more exploration as less likely topic assignments are considered. If t is high, then we encourage exploitation as only topic assignments with high probability are considered. Note that if $t \geq .5$, then THRESHOLD is equivalent to ICM, since there can only ever be a single assignment which has more probability than the threshold. Similarly, if $t = 0$, then THRESHOLD is equivalent to Gibbs sampling, since no assignment will ever be rejected.

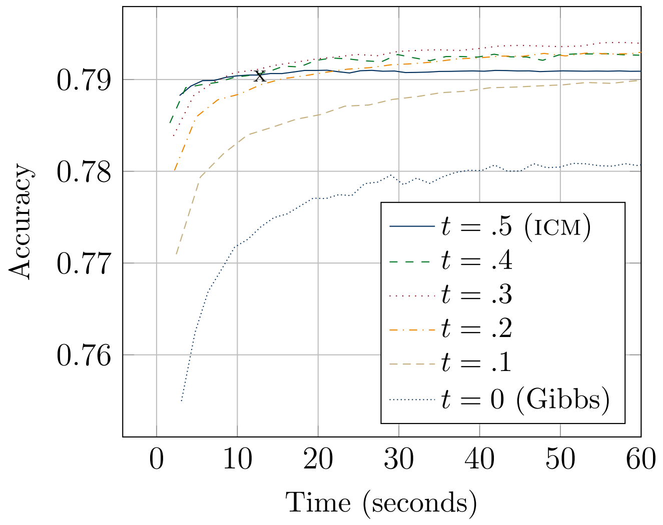


Figure 3.3: Accuracy versus time for the THRESHOLD algorithm, with various values for t. Note that for $t = .5$, THRESHOLD is equivalent to ICM, and that for $t = 0$, THRESHOLD is equivalent to Gibbs sampling. We have also marked the median point of convergence for ICM with an X-mark.

The results for THRESHOLD can be seen in figure 3.3. For $t = .3$ and $t = .4$, THRESHOLD outperforms ICM in terms of accuracy even if we cut off the THRESHOLD algorithm at the median time for ICM convergence. For $t = .2$ and $t = .1$, THRESHOLD eventually achieves better accuracy than ICM, but at the cost of increased runtime.

3.2 Threshold as Initialization

Using the THRESHOLD algorithm, which does indeed discourage low probability topic assignments, we have achieved superior classification accuracy within the runtime constraints of interactive use cases for ITM above the performance of ICM alone. However, note that the use of the THRESHOLD algorithm loses the benefit of a clear stopping criteria due to the sampling in the algorithm.

One simple way to alleviate this problem is to use a sampling algorithm to explore for a number of iterations, and then run ICM to ensure that we converge to a local maximum. In this way, we use the exploration in the sampling algorithm to initialize the state for the ICM hill climbing. When this approach is applied to the ITM, the main concern will be whether this approach will still allow us to keep the runtime small enough for interactive use.

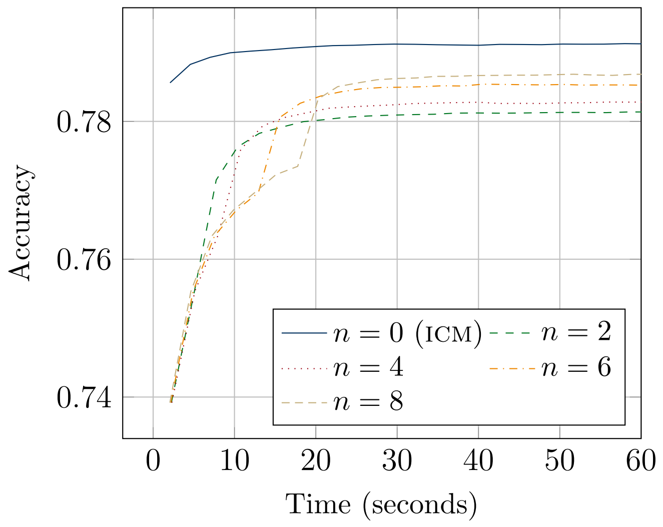


Figure 3.4: Accuracy versus time when using Gibbs sampling as an initializer for ICM. The value n indicates the number of iterations of sampling before we switch inference to use ICM. Note that $n = 0$ is simply ICM.

We can of course simply utilize Gibbs sampling in order to provide our initial exploration of the posterior. To do so, we repeat our previous experimental set up with the ITM. When the time comes to do an interactive update, we run a Gibbs sampler for n iterations, and then finish by running ICM. We experiment with values of $n \in [0, 2, ..., 8]$. The results of these experiments can be seen in figure 3.4.

Most importantly, we see that $n = 0$ (which is equivalent to just using ICM) outperforms any other choice of n. This is because immediately after each new constraint, we revoke certain topic assignments of words and documents which relate to the newly formed constraints (see Hu et al. [23] for more details). In other words, immediately after adding a new constraint, the most important words (namely the newly constrained words) have no topic assignments or document level topic information. Consequently, the Gibbs sampler initially makes assignments for those words which are too close to uniform random. On the other hand, ICM will very quickly choose a favorite topic for the new word and assign nearly all the newly constrained words to the chosen topic. Given that the user constrained these words in such a way to try and pool them into a single topic, the maximization behavior of ICM is likely to yield the correct action.

An alternative to Gibbs sampling for ICM initialization is one of our hybrid maximization-sampling algorithms. We need some amount of maximization to avoid the problems we observed with Gibbs sampling, but some amount of exploration can potentially allow us to exploit a better neighborhood when it comes time to choose a local maximum. To explore this possibility, we turn to THRESHOLD with $t = .3$ since it performed the best of our hybrid algorithms. We repeat the same procedure we previously used with Gibbs sampling, in which THRESHOLD is used for n iterations as an initializer for ICM.

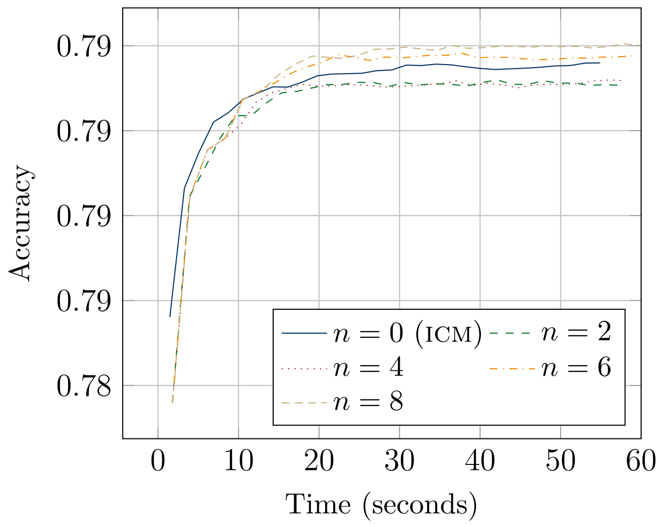


Figure 3.5: Accuracy versus time when using THRESHOLD with $t = .3$ as an initializer for ICM. The value n indicates the number of iterations of sampling before we switch inference to use ICM. Note that $n = 0$ is simply ICM.

The results of this experiment are summarized in figure 3.5. Both $n = 6$ and $n = 8$ demonstrate a slight improvement over simple ICM, indicating that there may be value in initializing with some limited amount of exploration. Furthermore, within the runtime requirements for human interactivity, ICM is still the best option.

3.3 Discussion

We have experimented with various hybrid maximization/sampling algorithms in an effort to understand the trade-off between exploitation and exploration when searching for a maximum a posteriori estimate in the ITM model. We have found that some care is needed when choosing how to hybridize sampling and maximization. For example, the poor performance of TopN indicates that the exploration must be allowed to make varied choices in topic assignment or else the lack of exploration can become harmful. However, using the THRESHOLD algorithm, we were able to improve upon the performance of either Gibbs sampling or ICM alone. Unfortunately, these gains come at a cost of increased runtimes. Thus, for the purpose of interactive uses of the ITM, we still recommend the use of pure ICM.

Chapter 4

Conclusion

While ICM has been seen in the literature before for certain undirected graphical models [6], it has not seen exposure in the literature for probabilistic models of text. Instead the topic modelinprobabilistic literature has concentrated on the use of mean field variational inference for fast inference [4]. However, for models in which there are no published or readily derivable variational updates or in cases when mean field variational inference is known to perform poorly, we submit that ICM is a viable alternative to Gibbs sampling. We have shown that ICM is able to quickly obtain locally optimal *maximum a posteriori* estimates. We have done so on a wide variety of probabilistic models of text for various interactive use cases.

For the ITM, we improved the quality of the model after performing interactive updates while reducing the runtime of the update. With Gibbs sampling, using the recommended 30 iterations of samplings for an update, our implementation of the ITM takes a median time of 43 seconds. In contrast, the median time of convergence for ICM was 11 seconds. With both the improved model quality and drastically improved update times, we are better able to use the ITM for the intended use cases with interactive updates.

For the problem of web search result clustering, we demonstrated that ICM can work with the non-parametric DP-MOM model. Using this model-based approach, we achieved state-of-the-art client-side clustering performance. Using ICM, clustering the search results takes milliseconds, meaning that even on the lowest-grade commodity machine, our approach could be used to create a client-side web search result clustering implementation.

Finally, we demonstrated that ICM works with MomResp, a model for aggregating document labels using crowdsourced annotations. Compared to Gibbs sampling, ICM not only provides more accurate document labels, but also has a run time which is orders of magnitude faster.

The success of ICM on these three models validates our claim that ICM as an inference algorithm is both accurate enough and fast enough to enable human interaction on a wide variety of probabilistic models. Our results suggest that any time a Gibbs sampler is employed to achieve an approximate *maximum a posteriori* estimate, then ICM is likely to be a fast and accurate alternative.

Appendix A

Finite Mixture of Multinomials for Word Search Result Clustering

In subsection 2.3.1, we showed that our approach to word search result clustering utilizing the DP-MOM model outperforms existing techniques which are amenable to client-side use cases. We also demostrated that we outperforms MST with respect to diversification. We claimed that the reason for this is that DP-MOM is able to learn the number of clusters for a particular query, whereas MST specified the number of clusters *a priori* . Consequently, DP-MOM is able to better handle various levels of query ambiguity and therefore performs better on the diversification task.

In this appendix, we will present additional results which reenforce the claim that DP-MOM achieves superior diversification due to the ability to learn the correct number of underlying clusters in the query data. We will do so by comparing the results of DP-MOM with a the finite version of the same model (namely Mixture of Multinomials).

A.1 Tuning Mixture of Multinomials

Before we can proceed with our comparison of Mixture of Multinomials with DP-MOM, we must first tune the model so that it can perform well on the task at hand. We must first select the number of clusters we will use. Since this must be done *a priori* , we tuned this parameter using a separate development data set. The best value on this data set was 5, which matches the optimal number of clusters used by the MST algorithm on the same data [19].

Another decision we must make is the values for the hyperparameters α and β. We now repeat the experiments previously described in subsection 2.3.1 using the Mixture of

Algorithm	AMBIENT	MORESQUE	All
ICM DP-MOM	**.768**	**.570**	**.625**
Gibbs DP-MOM	.758	.544	.604
ICM MOM	.742	.527	.585
Gibbs MOM	.739	.525	.585

Table A.1: Clustering quality results, as measured by the Rand index. We include the MST results for reference though they do not constitute client-side results. Bold indicates the best result.

Multinomials model in place of DP-MOM. As before, we sweep the parameters α and β and average the results across all parameterizations. We do this in order to demonstrate robustness with any reasonable values for the hyper-parameters. We select values for α and β in $[.1, .2, ..., 1]$.

Finally, we must choose a strategy for initialization. With DP-MOM, we found that initializing with a single cluster and allowing the model to grow to the correct number of clusters was best. However we cannot use this approach with the finite mixture, as the zero counts in the initially empty clusters will prevent those clusters from ever being used. Instead, we must initialize the document cluster assignments randomly.

A.2 Results

We now compare the performance of the finite Mixture of Multinomials with DP-MOM on the task of web search result clustering. We will utilize the same experimental setup as in subsection 2.3.1. We start by examining the clustering quality. The results are summarized in Table A.1. As was the case with DP-MOM, with the finite mixture we achieve better clustering quality with ICM than with Gibbs sampling. However, it is also the case that DP-MOM significantly outperforms the finite Mixture of Multinomials.

As before, we also measured the performance of Mixture of Multinomials in terms of diversification. Once again ICM yields superior performance compared to Gibbs sampling on the finite mixture, although the infinite mixture yields better results still.

Algorithm	K=3	K=5	K=10	K=15	K=20
ICM DP-MOM	**.517**	**.650**	**.811**	**.885**	**.927**
Gibbs DP-MOM	.508	.635	.795	.873	.917
ICM MOM	.503	.640	.771	.840	.884
Gibbs MOM	.502	.637	.769	.840	.885

Table A.2: Diversification results on all queries, as measured by S-recall@K. Bold indicate the best client-side result.

A.3 Discussion

Given the structural similarity between DP-MOM and Mixture of Multinomials (essentially the only difference is the prior distribution over document cluster assignments), we assert that the reason for the superior performance of DP-MOM is due to the prior distribution over cluster assignments allowing the model to learn the number of clusters from data.

We quantify the ability of DP-MOM to learn the number of clusters by measuring the mean absolute error in the discovered number of clusters with the true number of clusters. For DP-MOM with ICM, this value is 2.32. Considering that there are anywhere between 3 and 15 topics for a given query, this is a fairly reasonable amount of error given that we initialize with a single cluster. On the other hand, with the number of topics fixed at 5 as was the case with the finite mixture, the mean absolute error in the number of topics is 3.14. While this value is also fairly reasonable, the difference between the two explains the superior performance of DP-MOM, particularly when it comes to diversification at higher values of k.

References

[1] David Andrzejewski, Xiaojin Zhu, and Mark Craven. Incorporating domain knowledge into topic modeling via dirichlet forest priors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 25–32. ACM, 2009.

[2] David Andrzejewski, Xiaojin Zhu, and Mark Craven. Incorporating domain knowledge into topic modeling via dirichlet forest priors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 25–32. ACM, 2009.

[3] Charles E Antoniak et al. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, 1974.

[4] Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. On smoothing and inference for topic models. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 27–34. AUAI Press, 2009.

[5] Andrea Bernardini, Claudio Carpineto, and Massimiliano D'Amico. Full-subtopic retrieval with keyphrase-based search results clustering. In *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on*, volume 1, pages 206–213. IET, 2009.

[6] Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 259–302, 1986.

[7] Christopher M Bishop. *Pattern recognition and machine learning*, volume 1, pages 443–444. Springer New York, 2006.

[8] David Blackwell and James B MacQueen. Ferguson distributions via pólya urn schemes. *The Annals of Statistics*, pages 353–355, 1973.

[9] David M. Blei, Andrew Ng, and Michael Jordan. Latent dirichlet allocation. *JMLR*, 3: 993–1022, 2003.

[10] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23 (11):1222–1239, 2001.

[11] Thorsten Brants and Alex Franz. Web 1T 5-gram version 1. 2006.

[12] Bob Carpenter. Multilevel bayesian models of categorical data annotation. *Unpublished manuscript*, 2008.

[13] Claudio Carpineto, Stefano Mizzaro, Giovanni Romano, and Matteo Snidero. Mobile information retrieval with search results clustering: Prototypes and evaluations. *Journal of the American Society for Information Science and Technology*, 60(5):877–895, 2009.

[14] Claudio Carpineto, Stanislaw Osiński, Giovanni Romano, and Dawid Weiss. A survey of web clustering engines. *ACM Computing Surveys (CSUR)*, 41(3):17, 2009.

[15] Ming-Hui. Chen, Qi-Man Shao, and Joseph G Ibrahim. *Monte Carlo methods in Bayesian computation*. Springer New York, 2000.

[16] Kristin A. Cook and James J. Thomas. Illuminating the path: The research and development agenda for visual analytics. Technical report, Pacific Northwest National Laboratory (PNNL), Richland, WA (US), 2005.

[17] Gregory F Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial intelligence*, 42(2):393–405, 1990.

[18] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[19] Antonio Di Marco and Roberto Navigli. Clustering web search results with maximum spanning trees. In *AI* IA 2011: Artificial Intelligence Around Man and Beyond*, pages 201–212. Springer, 2011.

[20] Paul Felt, Robbie Haertel, Eric Ringger, and Kevin Seppi. Momresp: A bayesian model for multi-annotator document labeling. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA).

[21] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741, 1984.

[22] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, 2004.

[23] Yuening Hu, Jordan L Boyd-Graber, and Brianna Satinoff. Interactive topic modeling. In *ACL*, pages 248–257, 2011.

[24] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[25] Marina Meila and David Heckerman. An experimental comparison of model-based clustering methods. *Machine learning*, 42(1-2):1–2, 2001.

[26] Xiao-Li Meng and Donald B Rubin. Maximum likelihood estimation via the ecm algorithm: A general framework. *Biometrika*, 80(2):267–278, 1993.

[27] Thomas P Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc., 2001.

[28] Kevin P Murphy. *Machine learning: a probabilistic perspective*, page 744. MIT Press, 2012.

[29] Radford M Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000.

[30] Stanisław Osiński, Jerzy Stefanowski, and Dawid Weiss. Lingo: Search results clustering algorithm based on singular value decomposition. In *Intelligent information processing and web mining*, pages 359–368. Springer, 2004.

[31] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[32] Adrian E Raftery and Steven M Lewis. Implementing mcmc. In *Markov chain Monte Carlo in practice*, pages 115–130. Springer, 1996.

[33] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[34] Loïs Rigouste, Olivier Cappé, and François Yvon. Inference and evaluation of the multinomial mixture model for text clustering. *Information processing & management*, 43(5):1260–1280, 2007.

[35] Ruihua Song, Zhenxiao Luo, Ji-Rong Wen, Yong Yu, and Hsiao-Wuen Hon. Identifying ambiguous queries in web search. In *Proceedings of the 16th international conference on World Wide Web*, pages 1169–1170. ACM, 2007.

[36] David Sontag and Daniel M Roy. Complexity of inference in topic models. In *NIPS Workshop on Applications for Topic Models: Text and Beyond*, 2009.

[37] Naonori Ueda and Ryohei Nakano. Deterministic annealing em algorithm. *Neural Networks*, 11(2):271–282, 1998.

[38] Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

[39] Daniel David Walker. *Bayesian text analytics for document collections*. PhD thesis, Brigham Young University, 2012.

[40] Daniel David Walker and Eric K. Ringger. Model-based document clustering with a collapsed gibbs sampler. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 704–712. ACM, 2008.

[41] Guan Yu, Ruizhang Huang, and Zhaojun Wang. Document clustering via dirichlet process mixture model with feature selection. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 763–772. ACM, 2010.

[42] S. Zhong and J. Ghosh. Generative model-based document clustering: a comparative study. *Knowledge and Information Systems*, 8(3):374–384, 2005.