

Entity Disambiguation with Hierarchical Topic Models

Saurabh S. Kataria*
Pennsylvania State U.
State College, PA
skataria@ist.psu.edu

Krishnan S. Kumar
Yahoo! Labs
Bangalore, India
krishsk@yahoo-inc.com

Rajeev Rastogi
Yahoo! Labs
Bangalore, India
rrastogi@yahoo-inc.com

Prithviraj Sen
Yahoo! Labs
Bangalore, India
sen@yahoo-inc.com

Srinivasan H Sengamedu
Yahoo! Labs
Bangalore, India
shs@yahoo-inc.com

ABSTRACT

Disambiguating entity references by annotating them with unique ids from a catalog is a critical step in the enrichment of unstructured content. In this paper, we show that topic models, such as *Latent Dirichlet Allocation* (LDA) and its hierarchical variants, form a natural class of models for learning accurate entity disambiguation models from crowd-sourced knowledge bases such as Wikipedia. Our main contribution is a *semi-supervised* hierarchical model called *Wikipedia-based Pachinko Allocation Model* (WPAM) that exploits: (1) All words in the Wikipedia corpus to learn word-entity associations (unlike existing approaches that only use words in a small fixed window around annotated entity references in Wikipedia pages), (2) Wikipedia annotations to appropriately bias the assignment of entity labels to annotated (and co-occurring unannotated) words during model learning, and (3) Wikipedia's category hierarchy to capture co-occurrence patterns among entities. We also propose a scheme for pruning spurious nodes from Wikipedia's crowd-sourced category hierarchy. In our experiments with multiple real-life datasets, we show that WPAM outperforms state-of-the-art baselines by as much as 16% in terms of disambiguation accuracy.

Categories and Subject Descriptors

H.2.8 [Database applications]: Data mining

General Terms

Algorithms, Theory, Performance

*Work done while first author was interning at Yahoo!Labs.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD'11, August 21–24, 2011, San Diego, California, USA.
Copyright 2011 ACM 978-1-4503-0813-7/11/08 ...\$10.00.

Keywords

Entity Resolution, Disambiguation, Topic Models

1. INTRODUCTION

Even though the world wide web is a veritable knowledge base for everything under the sun, a large chunk of it still exists in the form of unstructured text generated with little or no curation. One important step to instilling structure into such free-form text is to collect all references to entities within it and annotate them with ids from an existing catalog of entities. Doing this will not only enable merge operations with other pieces of similarly annotated text but also promises to aid semantic search, information extraction and integration, document classification, and a host of other applications. For instance, if we know that a user is browsing a page about Michael Jordan the basketball player, then we can show the user additional articles related to only Michael Jordan the sportsperson and not Michael I. Jordan the machine learning researcher.

The crucial task in annotating entity references is to decide which entity from the catalog a particular reference is associated with. In this paper, we refer to this problem as the *entity disambiguation* problem.

1.1 Leveraging Wikipedia for Entity Disambiguation

With over 3.4 million crowd-sourced entities, Wikipedia [2] is clearly a formidable resource, one that can serve as a comprehensive reference catalog for large-scale entity disambiguation. Each Wikipedia entity has a separate page, and a vast network of internal links annotate words in the body of pages with entities that they refer to. The copious annotations constitute valuable training data, and prior work [9, 17, 11, 19, 14] has used them to learn models for disambiguating entities at scale. Thus, Wikipedia has enabled a paradigm of *weak supervision* where freely available annotations are used to train machine-learned models.

One way to broadly categorize the various entity disambiguation approaches proposed in the literature is based on the sources of evidence that they utilize. The *local context* of a reference forms one major source of evidence used in previous work [9, 17, 11, 14]. For instance, there exist a number of Columbus's, e.g., explorer, film director, etc., but

if the words “ocean” or “ship” appear in the vicinity, then that should increase our belief that the reference is to the explorer. Existing approaches use local context evidence for entity disambiguation in two steps. First, they learn the context for each entity from the local context of annotated references to the entity (embedded in other Wikipedia pages). Subsequently, to resolve a reference, they compare the local context of the reference with the (learned) context for each candidate entity to determine their compatibility.

Co-occurrence patterns form yet another major source of evidence used in prior work [11, 19, 14] where the hypothesis is that certain entities often appear together and disambiguating one reference should help to decide which entities the other references within the same document are referring to. For example, there are eight Michael Jordan’s (basketball player, researcher, actor, etc.) and three Charles Barkley’s (basketball player, politician, etc.) in Wikipedia. But if a document contains mentions of both Michael Jordan and Charles Barkley, then we can be fairly certain that both references are to basketball players.

1.2 Shortcomings of Existing Approaches

Prior research works [9, 17, 14], when computing local context compatibility, have used a range of window sizes (e.g., 55 words in [9] and 7 words in [17]) around each reference for determining the words that are a part of its local context. Finding a single optimal value for the window size parameter is a non-trivial task – a window size that is too small can cause important words to be excluded from the local context while irrelevant words may become part of the context with too large a window size.

Furthermore, even though freely available knowledge bases such as Wikipedia contain a large number of annotations, they are still at best, only *partially* annotated datasets. (Usually, only the first occurrence of a relevant reference on a page is annotated as advocated by Wikipedia’s manual of style [4].) Existing disambiguation methods completely ignore un-annotated references to an entity in a Wikipedia page, and so the local context of these references is not included in the context for the entity.

1.3 Problem Definition

We denote the set of all Wikipedia entity pages by \mathcal{W} . Consider an arbitrary set of documents \mathcal{D} . These can be news articles, product or restaurant reviews, blog posts, or crawled web pages. The entity disambiguation problem is to label the entity references in documents from \mathcal{D} with Wikipedia entities in \mathcal{W} . In this paper, we assume that entity references are given to us. Thus, we do not address the problem of identifying entity references in text which is a separate problem that can be handled using named-entity recognizers as described in [11], dictionary lookups, etc.

1.4 Our Contributions

Our main contribution is a *weakly semi-supervised hierarchical topic model* for entity disambiguation called *Wikipedia-based Pachinko Allocation Model* (WPAM). WPAM is a hierarchical variant of the popular *Latent Dirichlet Allocation* (LDA) [7] topic model, and is inspired by the *Pachinko Allocation Model* (PAM) recently proposed in [16]. However, unlike PAM which is completely unsupervised, WPAM is semi-supervised. WPAM extensively leverages Wikipedia pages, annotations, and category information to provide a

form of weak supervision when training models. To the best of our knowledge, our work is the first to apply topic models in conjunction with Wikipedia for large-scale entity disambiguation. There is a body of work [5, 8, 25] that employs topic models for entity resolution but, like PAM, these are also unsupervised and do not exploit Wikipedia resources.

Topic models associate a latent topic with each word in a document. One of our key ideas is to associate a distinct topic with each Wikipedia entity. Thus, using topic models to label words with entity-specific topics provides us with a principled approach to entity disambiguation. Unlike existing disambiguation schemes (described earlier), topic models are oblivious to window size settings – their internal machinery naturally selects words that frequently co-occur with each entity (across the entire document corpus) to learn word-entity mappings. The selected words can be from anywhere within a document including the neighborhood of un-annotated references. Thus, topic models circumvent the issue of setting a window size and still learn high-quality word-entity associations, and this can boost disambiguation accuracy. Early topic models like LDA relied primarily on co-occurrences at the word level for labeling. Subsequent *hierarchical* versions such as PAM are capable of capturing not only word-entity associations but also co-occurrence patterns among entities using a topic hierarchy.

WPAM uses Wikipedia’s category hierarchy [3] as the topic hierarchy to disambiguate entities with hierarchical topic models. Wikipedia’s category hierarchy contains entities as leaves and each entity is assigned to (one or more) parent categories. Furthermore, semantically related entities (that are likely to occur together in documents) are grouped under one or more relevant categories. For instance, Michael Jordan and Charles Barkley are both assigned to the category “African American basketball players” (among a number of others) which is a sub-category of “American basketball players” and “African-American sportspeople”. Thus, WPAM uses a flexible, semantically rich topic hierarchy that captures entity correlations much better compared to the rigid hierarchies with a fixed number of levels used by different PAM variants (e.g., four-level PAM [16]).

With millions of fine-grained topics, one per entity, the space of possible word-topic assignments is enormous and this can confound PAM. Thus, a key challenge here is to be able to guide the topic models so that they annotate words with the correct entities. To this end, we develop *weakly semi-supervised* techniques that exploit Wikipedia’s crowd-sourced annotations for WPAM model learning. During learning, when selecting an entity to label a word, our techniques introduce a bias in favor of entities that frequently appear in annotations for either the word or the document containing it. This bias originating from annotated words also spreads to co-occurring un-annotated words, and recursively through them to more words. Thus, since un-annotated portions of documents also play a role in propagating entity labels, WPAM’s learning techniques are semi-supervised.

Finally, due to the lack of curation and the crowd-sourced manner in which it is produced, the Wikipedia hierarchy contains some spurious categories that can mislead our WPAM model. For example, Michael Jordan is listed under the categories “Living People” and “1963 Births”. We develop techniques to prune such irrelevant categories containing uncorrelated entities which co-occur infrequently.

To gauge the effectiveness of our topic model-based en-

tity disambiguation approach, we conduct an extensive experimental study using two real-life datasets: (1) Held-out subset of annotated Wikipedia data, and (2) Pre-annotated news articles from New York Times (NYT). Our experimental results indicate that our WPAM model is able to achieve an impressive disambiguation accuracy of 70% compared to 54% for the current state-of-the-art method of [14].

The rest of the paper is organized as follows. In Section 2, we develop weakly semi-supervised techniques for LDA and use Wikipedia annotations to learn accurate disambiguation models based on entity context alone. In Section 3, we develop weakly semi-supervised techniques for our WPAM hierarchical topic model that exploits both Wikipedia annotations and its category hierarchy to learn accurate models of disambiguation based on both entity context and co-occurrence. In Section 4, we experiment with real-life Wikipedia and NYT datasets to demonstrate the efficacy of our proposed techniques. We take a closer look at relevant related work in Section 5, and Section 6 concludes the paper.

2. WEAKLY SEMI-SUPERVISED TOPIC MODELS

Topic models represent documents as mixtures of (latent) topics, where each topic is a probability distribution over words. The document-topic and topic-word distributions are learned automatically from the data in an unsupervised manner with no human labeling or prior knowledge required.

Our entity disambiguation models build upon the popular LDA model which has been extensively used for classification of short text segments [23], unsupervised entity resolution [5, 8, 25], prediction of movie ratings from reviews [6], and extraction of ratable aspects of objects from online reviews [27]. In this section, we first show how to use LDA to assign entity labels to document words by mapping each entity to a separate topic. We then present techniques that leverage Wikipedia annotations to bias prior document-topic and topic-word distributions. The result is a semi-supervised version of LDA which is different from the unsupervised models previously used for entity resolution [5, 8, 25].

2.1 Latent Dirichlet Allocation

LDA is a probabilistic generative model that assumes Dirichlet priors on document-topic and topic-word distributions. Consider a collection of M documents containing words from the vocabulary of terms $\{1, \dots, T\}$, and let $\{1, \dots, K\}$ be a set of topics. The LDA model is defined by two parameters: (1) the multinomial distribution $\vec{\theta}_m = P(z|d = m)$ over topics for each document m , and (2) the multinomial distribution $\vec{\phi}_k = P(w|z = k)$ over words for each topic k . LDA’s document generation process is as follows:

- For each topic k , sample word distribution $\vec{\phi}_k \sim \text{Dir}(\vec{\beta})$.
- For each document m
 - Sample topic distribution $\vec{\theta}_m \sim \text{Dir}(\vec{\alpha})$.
 - For each word w_i in document m
 - * Sample a topic $z_i \sim \text{Mult}(\vec{\theta}_m)$.
 - * Sample a word $w_i \sim \text{Mult}(\vec{\phi}_{z_i})$.

Above, $\text{Dir}(\vec{\alpha})$ and $\text{Dir}(\vec{\beta})$ are Dirichlet distributions with hyper-parameters $\vec{\alpha}$ and $\vec{\beta}$, respectively. And $\text{Mult}(\vec{\theta}_m)$ and $\text{Mult}(\vec{\phi}_{z_i})$ are multinomial distributions. In the following

subsections, we will use \vec{w} to denote the vector of words contained in the documents, and \vec{z} to denote the corresponding topics for the words.

2.2 Inference Using Gibbs Sampling

The problem of statistical inference involves estimating the probability distribution $\vec{\phi}_k$ over words associated with each topic k , the distribution over topics $\vec{\theta}_m$ for each document m , and often, the topic responsible for generating each word. Instead of directly estimating $\vec{\phi}_k$ and $\vec{\theta}_m$, we use the approach of [12] that first constructs the posterior distribution $P(\vec{z}|\vec{w})$ and then estimates $\vec{\phi}_k$ and $\vec{\theta}_m$ from this posterior distribution.

To efficiently estimate the posterior distribution, [12] uses Gibbs sampling which is a simple and widely applicable *Markov chain Monte Carlo* (MCMC) algorithm for sampling from complex high-dimensional distributions. Starting with a random topic assignment \vec{z} , the Gibbs sampling algorithm iterates through each word in the document corpus. In each step, the algorithm samples a topic assignment for a word w_i conditioned on the topic assignments of all other words. More formally, in each Gibbs sampling step, the algorithm replaces z_i by a topic drawn from the distribution $P(z_i|\vec{z}_{-i}, \vec{w})$, where \vec{z}_{-i} is \vec{z} without the i^{th} component.

For a word $w_i = t$ in document m , the conditional probability that $z_i = k$ is given by (a detailed derivation of the formula below can be found in [13]):

$$P(z_i = k|\vec{z}_{-i}, \vec{w}, \vec{\alpha}, \vec{\beta}) \quad (1)$$

$$\propto \frac{n_{t,-i}^{(k)} + \beta_t}{\sum_{t'=1}^T (n_{t',-i}^{(k)} + \beta_{t'})} \cdot \frac{n_{k,-i}^{(m)} + \alpha_k}{\sum_{k'=1}^K (n_{k',-i}^{(m)} + \alpha_{k'})}$$

Above, $n_{t,-i}^{(k)}$ is the number of times term t is assigned topic k excluding the current assignment; similarly, $n_{k,-i}^{(m)}$ is the number of words in document m that are assigned topic k excluding the current assignment. The Gibbs sampling equation (1) is fairly intuitive – the first term is the probability of term t under topic k and the second term is the probability of topic k in document m . Observe that due to the second term, the topic assignment for a word is heavily influenced by the topic assignments for the remaining words in the document. Therefore, co-occurring words do influence each other’s topic assignments and will very likely be assigned the same topic by the Gibbs sampling algorithm.

The sequence of samples obtained from Gibbs sampling form a Markov chain that converges to the posterior distribution $P(\vec{z}|\vec{w})$. Thus, after an initial burn-in period, we can get a representative set of samples from the distribution by collecting Gibbs samples at regularly spaced intervals. These can then be used to estimate the distributions $\vec{\phi}_k$ and $\vec{\theta}_m$ as described in [12, 13].

In our implementation, we estimate the $\vec{\alpha}$ and $\vec{\beta}$ Dirichlet hyper-parameters using Minka’s fixed-point iteration [21].

2.3 Using LDA for Entity Disambiguation

We use LDA to develop an unsupervised algorithm for disambiguating the entity references in document set \mathcal{D} . Essentially, our disambiguation algorithm runs Gibbs sampling with a separate topic per Wikipedia entity. Thus, Gibbs sampling assigns entity labels to all the words in the documents, effectively disambiguating all the entity references.

Our disambiguation algorithm has two phases: *training* and *labeling*. In the training phase, we run Gibbs sampling only on the collection of Wikipedia pages \mathcal{W} . Let $\vec{z}_{\mathcal{W}}$ denote the topic assignments for words in \mathcal{W} at the end of the training phase. Next, in the labeling phase, we run an *incremental* Gibbs sampling algorithm on $\mathcal{W} \cup \mathcal{D}$ that only samples topics for words in \mathcal{D} while keeping the topics for words in \mathcal{W} fixed at $\vec{z}_{\mathcal{W}}$. Let $\vec{z}_{\mathcal{D}}$ be the topic assignments for words in \mathcal{D} . Then, in each incremental Gibbs sampling step during the labeling phase, only $\vec{z}_{\mathcal{D}}$ changes but $\vec{z}_{\mathcal{W}}$ stays constant. A topic for word w_i in \mathcal{D} is sampled according to the Gibbs sampling equation (1). Note that the count $n_{t,-i}^{(k)}$ in the equation is the total number of times term t is assigned topic k across $\vec{z}_{\mathcal{D}}$ and $\vec{z}_{\mathcal{W}}$.

A major scalability challenge is that with a separate topic per Wikipedia entity, the number of topics K is equal to the number of entities in Wikipedia which can be quite large. As a result, in each Gibbs sampling step, the cost of sampling a topic assignment for w_i from its conditional distribution can be prohibitive since we need to compute the conditional probabilities for all K topics according to Equation (1). The key observation we make here is that in general only a few entities are relevant to each document. And these are typically entities with a surface form that matches a keyword in the document. (The surface forms for an entity are the anchor text appearing within links to the entity embedded in Wikipedia pages.) Thus, for each document, we identify the entities with matches, and only consider topics corresponding to these entities when sampling a topic assignment for a word in the document. This simple optimization results in a substantial speedup, enabling us to scale to lots of topics.

2.4 Weakly Semi-Supervised LDA

The standard LDA topic model is completely unsupervised, and determines topics for words entirely based on the co-occurrence patterns of words across documents. However, with millions of fine-grained topics, the space of topic assignments to words is vast, and this makes the task of finding a “good” topic assignment extremely challenging.

In this section, we leverage the prior Wikipedia annotations of words with entities to provide a form of weak supervision to the basic LDA model with the objective of improving its labeling accuracy. Let \mathcal{A} denote the portions of Wikipedia pages containing only annotated words along with their entity labels. The key idea underlying our approach is to bias the topic-word distributions $\vec{\phi}_k$ in favor of words that are frequently annotated with (the entity corresponding to) topic k in \mathcal{A} , and the document-topic distributions $\vec{\theta}_m$ in favor of topics (corresponding to entities) that frequently occur in document m ’s annotations in \mathcal{A} .

We consider the Wikipedia annotations \mathcal{A} as multinomial observations that bias the distributions of parameters $\vec{\theta}_m$ and $\vec{\phi}_k$. Recall that $\vec{\theta}_m$ and $\vec{\phi}_k$ have Dirichlet priors with hyper-parameters $\vec{\alpha}$ and $\vec{\beta}$, respectively. $P(\vec{\phi}_k | \vec{\beta}, \mathcal{A})$, the posterior distribution of $\vec{\phi}_k$ conditioned on annotations \mathcal{A} , can be shown to be $\text{Dir}(\vec{\phi}_k | \vec{\beta} + \vec{\delta}^{(k)})$. Here, $\delta_t^{(k)}$ is the number of times term t is assigned topic k in \mathcal{A} . Similarly, we can show that $\vec{\theta}_m$ has a Dirichlet posterior with hyper-parameters $\vec{\alpha} + \vec{\delta}^{(m)}$ where $\delta_k^{(m)}$ is the number of annotated words in document m that are assigned topic k in \mathcal{A} .

We finally get the conditional probability that $z_i = k$ (for

word $w_i = t$ in document m) given the observed Wikipedia annotations \mathcal{A} as:

$$P(z_i = k | \vec{z}_{-i}, \vec{w}, \vec{\alpha}, \vec{\beta}, \mathcal{A}) \quad (2)$$

$$\propto \frac{n_{t,-i}^{(k)} + \beta_t + \delta_t^{(k)}}{\sum_{t'=1}^T (n_{t',-i}^{(k)} + \beta_{t'} + \delta_{t'}^{(k)})} \cdot \frac{n_{k,-i}^{(m)} + \alpha_k + \delta_k^{(m)}}{\sum_{k'=1}^K (n_{k',-i}^{(m)} + \alpha_{k'} + \delta_{k'}^{(m)})}$$

Thus, we incorporate knowledge of Wikipedia annotations in the new Gibbs sampling equation (2) above by adding the prior counts $\delta_t^{(k)}$ and $\delta_k^{(m)}$ observed in \mathcal{A} to the counts $n_{t,-i}^{(k)}$ and $n_{k,-i}^{(m)}$, respectively. This has the effect of biasing the topic assignment for a word (in each Gibbs sampling step) in favor of topics that frequently appear in annotations \mathcal{A} for either the word or the document containing it. Furthermore, this topic bias spreads to other occurrences of the word in the document corpus as well as to co-occurring words, and recursively through them to more words. These words that propagate topic assignment biases can be un-annotated words not contained in \mathcal{A} , and hence our LDA model with biased $\vec{\phi}_k$ and $\vec{\theta}_m$ distributions is semi-supervised. Note that this bias propagation can be quite powerful in practice; in fact, previous research on semi-supervised learning [18, 22] has shown that if we can avail of large amounts of unlabeled data and combine that with the labeled training data then the accuracy of machine-learned models can be dramatically improved. Our experimental results corroborate this with semi-supervised LDA improving disambiguation accuracy by as much as 30.2% over standard LDA.

3. WEAKLY SEMI-SUPERVISED HIERARCHICAL TOPIC MODEL

LDA selects topics based on correlations among words, but it does not explicitly model correlations among topics. However, real-world documents are generally topically coherent, and so a model that can capture topic correlations as well can lead to higher disambiguation accuracy. As pointed out in Section 1, exploiting topic correlations can help to resolve an ambiguous Michael Jordan reference to the basketball player (as opposed to the researcher) if the document also contains a reference to Charles Barkley the basketball player. LDA may have difficulty disambiguating such references because it can combine arbitrary sets of topics.

Hierarchical topic models extend LDA to capture both word and topic correlations in a single unified framework. A hierarchical topic model takes as input a topic hierarchy which can be an arbitrary DAG over topics with interior nodes representing a correlation among child topics. It assigns each word a root-to-leaf topic path with a preference for annotating a document’s words with overlapping paths that share subpaths starting at the root. Thus, the topics for each document are close by in the topic hierarchy, and therefore highly correlated.

In Section 3.1, we describe our *weakly semi-supervised hierarchical topic model* which we call *Wikipedia-based Pachinko Allocation Model* (WPAM). WPAM builds on the *Pachinko Allocation Model* (PAM) proposed in [16] but differs from [16] in two important respects. First, [16] focuses on a fixed four-level topic hierarchy – a problem with such a rigid structure is that it may not accurately model real-world topic correlations. In contrast, WPAM leverages the Wikipedia category hierarchy that has a flexible DAG structure with

a different number of levels in different regions of the hierarchy. The Wikipedia hierarchy groups semantically related entities (and categories) under one or more relevant categories – thus, it is quite effective at capturing topic correlations. A second key difference between PAM and WPAM is that PAM is unsupervised. WPAM, on the other hand, is a weakly semi-supervised model that uses Wikipedia annotations as training data to bias topic assignments.

The Wikipedia hierarchy contains over 0.34 million categories and is generated by human volunteers. Consequently, it is not perfect and contains irrelevant categories such as “Living People”, “People with Year of Birth Missing”, etc. In Section 3.2, we describe a scheme for cleansing the hierarchy of extraneous category nodes.

3.1 Wikipedia-based Pachinko Allocation Model

WPAM uses Wikipedia’s category hierarchy as the topic hierarchy which we denote by H . Non-leaf topics in H are Wikipedia categories, and leaf topics correspond to Wikipedia entities. We denote the set of children of a non-leaf topic k by $c(k)$. WPAM assigns a root-to-leaf path $z_i = \langle z_{i1}, z_{i2}, \dots, z_{il_i} \rangle$ from H to each word w_i in the document corpus. Here, z_{i1} is always the root, and topic $z_{i(j+1)}$ is a child of z_{ij} . The WPAM model is defined by the following parameters: (1) For each document m and non-leaf topic k , a multinomial distribution $\vec{\theta}_{mk}$ over k ’s children $c(k)$, and (2) For each leaf topic k , a multinomial distribution $\vec{\phi}_k$ over words.

The model parameters $\vec{\theta}_{mk}$ have a Dirichlet prior with hyper-parameters $\vec{\alpha}_k$, and the parameters $\vec{\phi}_k$ have a Dirichlet prior with hyper-parameters $\vec{\beta}$. As shown in Section 2.4, with the Wikipedia annotations \mathcal{A} , $\vec{\phi}_k$ has a Dirichlet posterior with hyper-parameters $\vec{\beta} + \vec{\delta}^{(k)}$ which we denote by $\vec{\beta}^{(k)}$ (recall that $\delta_t^{(k)}$ is the number of times term t is assigned topic k in \mathcal{A}). Similarly, if $\delta_{kk'}^{(m)}$ is the number of topic paths in document m in \mathcal{A} containing k' as a subtopic of k , then $\vec{\theta}_{mk}$ can be shown to have a Dirichlet posterior with hyper-parameters $\vec{\alpha}_k + \vec{\delta}_k^{(m)}$ which we denote by $\vec{\alpha}_k^{(m)}$.

A slight problem we face here is that Wikipedia annotations only specify entities and not paths in the topic hierarchy H . Furthermore, since H is a DAG, there can be multiple paths in H from the root to the leaf topic corresponding to an entity. This makes it difficult to obtain exact $\delta_{kk'}^{(m)}$ counts for topic, subtopic pair occurrences in \mathcal{A} . So we approximate $\delta_{kk'}^{(m)}$ as follows. First, for each annotation $a \in \mathcal{A}$ in document m , we compute the fraction of paths in H from the root to the entity specified in a that pass through topic k and its subtopic k' . The fraction essentially represents the probability that a topic path for annotation a passes through the topic, subtopic pair (k, k') . Our $\delta_{kk'}^{(m)}$ estimate is then the sum of the fraction values for all the annotations in document m . Note that we can compute the number of paths between any two nodes of a DAG efficiently using dynamic programming, and this can be used to calculate the path fraction value for each annotation.

WPAM uses the Dirichlet posterior distribution for each $\vec{\phi}_k$ and $\vec{\theta}_{mk}$ to generate documents, and so it is a semi-supervised model. The generative process for a document collection under WPAM is as follows:

- For each leaf topic k , sample word distribution $\vec{\phi}_k \sim \text{Dir}(\vec{\beta}^{(k)})$. Here, $\vec{\beta}^{(k)} = \vec{\beta} + \vec{\delta}^{(k)}$.
- For each document m

- For each non-leaf topic k , sample topic distribution $\vec{\theta}_{mk} \sim \text{Dir}(\vec{\alpha}_k^{(m)})$. Here, $\vec{\alpha}_k^{(m)} = \vec{\alpha}_k + \vec{\delta}_k^{(m)}$.
- For each word w_i in document m
 - * Sample a root-to-leaf path $z_i = \langle z_{i1}, z_{i2}, \dots, z_{il_i} \rangle$ of length l_i from the topic hierarchy H . For each topic z_{ij} in the topic path, sample child $z_{i(j+1)} \sim \text{Mult}(\vec{\theta}_{mz_{ij}})$.
 - * Sample a word $w_i \sim \text{Mult}(\vec{\phi}_{z_{il_i}})$.

As before, we can use Gibbs sampling to compute the topic path assignments \vec{z} for document words in \vec{w} . Essentially, Gibbs sampling draws samples from the posterior distribution $P(\vec{z}|\vec{w})$ by sequentially sampling topic paths for each z_i from $P(z_i|\vec{z}_{-i}, \vec{w})$. For a word $w_i = t$ in document m , the conditional probability that $z_i = \langle k_1, k_2, \dots, k_{l_i} \rangle$ is given by:

$$P(z_i = \langle k_1, k_2, \dots, k_{l_i} \rangle | \vec{z}_{-i}, \vec{w}, \{\vec{\alpha}_k\}, \vec{\beta}, \mathcal{A}) \quad (3)$$

$$\propto \frac{n_{t,-i}^{(k_{l_i})} + \beta_t^{(k_{l_i})}}{\sum_{t'=1}^T (n_{t',-i}^{(k_{l_i})} + \beta_{t'}^{(k_{l_i})})} \cdot \prod_{j=1}^{l_i-1} \frac{n_{k_j k_{j+1}, -i}^{(m)} + \alpha_{k_j k_{j+1}}^{(m)}}{\sum_{k' \in c(k_j)} (n_{k_j k', -i}^{(m)} + \alpha_{k_j k'}^{(m)})}$$

Above, $n_{t,-i}^{(k_{l_i})}$ is the number of occurrences of term t with leaf topic k_{l_i} excluding the current assignment, and $n_{k_j k', -i}^{(m)}$ is the number of times topic k' appears as a subtopic of topic k_j in a topic path of document m excluding the current assignment. In Equation (3) above, the first term is the probability of term t under leaf topic k_{l_i} and the second term is the probability of path $\langle k_1, k_2, \dots, k_{l_i} \rangle$ in document m . Observe that the second term induces a clustering effect among the topic paths of a document with a preference for paths with common subpaths from the root. Thus, words in a document are assigned entity labels that are close by in H , and hence correlated.

Our entity disambiguation algorithm first runs WPAM’s Gibbs sampling on the collection of Wikipedia documents \mathcal{W} . In the labeling phase, it disambiguates each reference in \mathcal{D} by running incremental Gibbs sampling on $\mathcal{W} \cup \mathcal{D}$. Each reference is finally labeled with the entity at the leaf of the topic path assigned to it. Note that, as in Section 2.3, we can significantly speed up our sampling algorithm by only considering the restriction of H to topics and entities mentioned in the document. In our experiments, we found that exploiting the Wikipedia category hierarchy enables WPAM to achieve up to 26.1% higher disambiguation accuracy.

3.2 Pruning Noisy Categories from Hierarchy

As mentioned earlier, due to the crowd-sourced manner in which it is produced, the Wikipedia hierarchy contains some spurious categories. These categories can cause WPAM to incorrectly infer topic correlations, and assign wrong entity labels to words. In this subsection, we propose a scheme to detect such spurious non-leaf topics, and prune them from the topic hierarchy H .

Now, good topics are the ones whose child topics are correlated and frequently co-occur in documents. Thus, our scheme for detecting spurious topics defines a goodness measure for each topic based on the co-occurrence of its children, and then deletes the topics with low goodness scores. A natural candidate for the goodness measure is *entropy* since it increases with the number of subtopics per document and as the co-occurrence among subtopics grows. Consider the

set of Wikipedia pages \mathcal{W} . Let \bar{z} be the topic paths in H assigned to words in \mathcal{W} by WPAM’s Gibbs sampling algorithm when training the WPAM model on \mathcal{W} . Then, for a non-leaf topic k , we define the entropy $E(k)$ as:

$$E(k) = - \sum_{m \in \mathcal{W}} \sum_{k' \in c(k)} \frac{n_{kk'}^{(m)}}{\sum_{k' \in c(k)} n_{kk'}^{(m)}} \log \frac{n_{kk'}^{(m)}}{\sum_{k' \in c(k)} n_{kk'}^{(m)}} \quad (4)$$

(Recall that $n_{kk'}^{(m)}$ is the number of times topic k' appears as a subtopic of topic k in a topic path of document m .) It is easy to see that if each document contains very few of topic k ’s children, then the value of $E(k)$ is low. In fact, in the extreme case, when only one subtopic of k appears per document, k ’s entropy is 0. On the other hand, if many of k ’s subtopics appear in each document, then the value of $E(k)$ is higher. The maximum value of course is $|\mathcal{W}| \cdot \log |c(k)|$ when all of k ’s subtopics are uniformly distributed within each document. Thus, the entropy $E(k)$ is effective at capturing subtopic co-occurrence counts for topic k , and so we use it as the goodness measure for topics.

Our scheme for pruning spurious topics from the Wikipedia hierarchy H iteratively deletes topics k with the lowest entropy scores $E(k)$. During each iteration, it breaks ties between topics with identical entropy scores by selecting the topic k that occurs most frequently in topic paths. The rationale here is that frequent topics have higher confidence levels. A topic k is deleted from hierarchy H by performing the following two modifications: (1) For every child k' of k , new edges are added from the parents of k to k' (thus parents of k become parents of k ’s children), and (2) Topic k and all the edges incident on k are deleted.

The stopping criterion for deleting topics from H is once the disambiguation accuracy of models trained with the new hierarchy (after a deletion) starts to decrease. In our experiments, we measure the disambiguation accuracy for a hierarchy H by first learning on a training dataset and then testing the learned model on a validation set.

4. EXPERIMENTAL RESULTS

In this section, we show that our WPAM model performs better than the state-of-the-art baselines proposed in [14] on real-life datasets. More specifically, we show that WPAM improves disambiguation accuracy by as much as 16% compared to [14]. We also show that our semi-supervised LDA model has 30.2% higher accuracy compared to standard LDA, and our WPAM model outperforms the best performing and standard LDA models by as much as 26.1% and 43%, respectively. We finally show that hierarchy pruning improves disambiguation performance by 6%.

4.1 Experimental Setup

Datasets: We created two datasets – one from Wikipedia and another from the New York Times corpus. The first dataset (called **WIKI**) is an extract of Wikipedia containing people belonging to seven categories – tennis, basketball, football and baseball players along with actors, musicians, and scholars. This dataset contains 58,577 people, each with an entity page, 627,370 intra-Wikipedia links/annotations and 13,035,881 tokens (after stop word removal). The second dataset is a subset of the New York Times annotated corpus [1]. The full corpus contains 1,855,658 annotated articles published in the New York Times, with 2,372,244 annotations. In some of our experiments, we train on WIKI

and test on a subset of the New York Times corpus containing entities that occur in WIKI too. To do this, we first need to match the entity names that occur in the WIKI dataset with those that occur in the New York Times corpus. We use exact string matching to identify matching entity names. The subset of the New York Times corpus that contains references to these matched entities is what we use to perform our experiments and we refer to this extract as NYT in the sequel. NYT contains 9151 unique person names, 65,012 references, 47,581 documents, and 23,283,125 tokens after stopword removal.

The experiments are performed on the above datasets in unlabeled and held-out modes. In the “unlabeled” mode, we present the dataset to the algorithms after hiding a subset of the annotations present in the dataset. Subsequently, we estimate performance based on the predictions on the hidden references. In the “held-out” mode, all references in the test data are unannotated. The datasets are called WIKI-U, WIKI-H, and NYT-H.

WIKI-U: In WIKI-U (WIKI data in the unlabeled mode), the training data consists of 4 splits each containing 75% of the pages in WIKI. In the remaining 25%, we hide 30% of the annotations. 100% of the data is used for training and precision is computed on the references with hidden annotations. We report the average precision over the 4 splits.

WIKI-H: In WIKI-H (WIKI data in held-out mode), the splits are the same as WIKI-U. The hidden portions of the pages (that is, 30% of 25% of the pages) are treated as separate held-out test pages. In other words, these parts are not used for training and are used only to evaluate performance.

NYT-H: In this case, WIKI is used for training and NYT is used for testing.

Performance Metrics: In our experiments, we use precision as the primary metric of comparison: precision is the fraction of entity references that are annotated correctly. In some applications, instead of the best matching entity, a (short) ranked list of entities is used to capture the correct entity. The metric used in these cases is “Precision@k” which is defined as the fraction of references for which the correct entity appears in the *top-k* candidate list for that reference. In our experiments, following prior work [14], we compare words appearing in the text with surface forms of entities in Wikipedia and choose the longest match to identify references.

Outline of Experiments: We perform the following experiments.

- Baseline, LDA variants, PAM, and WPAM on WIKI-U and WIKI-H.
- Baseline, LDA variants, PAM, and WPAM on NYT-H.
- Hierarchy pruning.
- Train and test execution times.

Baselines: We use the following two baselines from [14]. Local Context (LC): Given a local context window around a reference and an entity, LC first builds a feature vector by comparing the window with the entity’s textual metadata (extracted from Wikipedia) using string similarity metrics. The feature vector is then input to a support vector machine (SVM) and the entity with the highest SVM score is the disambiguation result.

Collective Disambiguation (CD): CD uses all the local contextual windows for references in a document and attempts to assign an entity to each reference such that the sum of

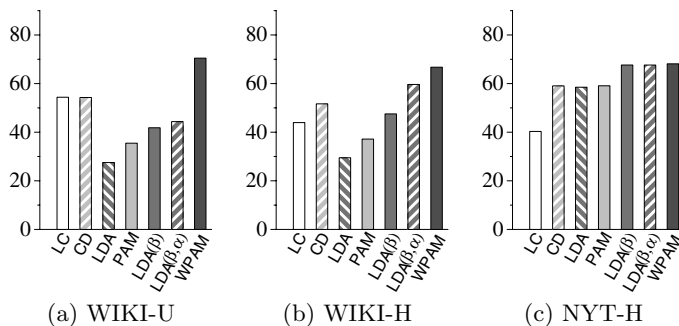


Figure 1: Precision of baselines, LDA variants, PAM, and WPAM.

the average SVM scores and the average *relatedness scores* [19] among pairs of assigned entities is maximized. For this optimization, in addition to the greedy hill climbing implemented in [14], we also use more sophisticated inference algorithms such as *loopy belief propagation* (LBP) [28]. The results we report were obtained using max-product LBP (using exponentiated relatedness as entries in clique potentials).

LDA variants: Section 2.4 introduced weakly semi-supervised LDA models. To estimate the gains in performance achieved by biasing the distributions of parameters $\vec{\theta}_m$ and $\vec{\phi}_k$, we experiment with two variants of weakly semi-supervised LDAs. Recall that semi-supervised learning was achieved by adding precounts $\vec{\delta}^{(k)}$ and $\vec{\delta}^{(m)}$ to Dirichlet hyperparameters $\vec{\beta}$ and $\vec{\alpha}$, respectively. We refer to LDA with only precounts in $\vec{\beta}$ as LDA(β), and with precounts in both $\vec{\alpha}$ and $\vec{\beta}$ as LDA(β, α).

For WPAM, we found that instead of initializing the Gibbs sampling algorithm with a random topic assignment, initializing word topics in a document with more frequently occurring candidate entities gives better performance. We report results with the latter initialization scheme.

Platform: The experiments were run on a Linux system with 32GB main memory and 8 64-bit 1.87GHz 4-core Intel Xeon processors each with 4MB cache.

4.2 Performance Comparison

Figure 1 shows the results of testing on WIKI-U, WIKI-H, and NYT-H. On all three datasets, WPAM outperforms all the remaining models. Furthermore, LDA, PAM, LDA(β), LDA(β, α), and WPAM show a steady progression of improving performance demonstrating the benefits of increasing levels of supervision and exploitation of the category hierarchy.

On the WIKI-U, WIKI-H, and NYT-H datasets, WPAM shows 16%, 15%, and 9% improvements, respectively, over CD. In fact, on the NYT-H dataset, all three proposed topic models, LDA(β), LDA(β, α) and WPAM, perform significantly better than the LC and CD baselines. This is because LC and CD use only limited local context for disambiguation compared to topic models. Furthermore, the superior performance of WPAM can be attributed to (1) the use of semi-supervised learning enabled by Wikipedia annotations, and (2) the Wikipedia hierarchy to capture entity co-occurrence patterns. We investigate the individual benefits due to each of these two aspects next.

To gauge the gains due to semi-supervision, we compare the precision of LDA(β, α) and the standard LDA models. LDA(β, α) outperforms standard LDA on WIKI-U, WIKI-

H, and NYT-H by 16.8%, 30.2%, and 9.1%, respectively. Similar gains are achieved by WPAM compared to PAM. This clearly demonstrates that unsupervised topic models have difficulty handling a large number of fine-grained topics, and semi-supervised learning techniques that combine unlabeled and labeled data are required.

To measure the efficacy of hierarchical models, we compare the precision of WPAM with the best performing LDA model, LDA(β, α). The difference in precision between WPAM and LDA(β, α) on WIKI-U and WIKI-H are 26.1% and 7.1%, respectively. A possible reason for LDA(β, α) doing almost as well as WPAM on NYT-H could be the smaller number of references contained in an NYT document. On an average, an NYT document contains 1.38 references whereas the corresponding number for a WIKI document is 20.83. This implies that there is less entity co-occurrence to be exploited in NYT. The precision gains are even more striking when we compare WPAM to standard LDA – on WIKI-U, WIKI-H, and NYT-H, WPAM outperforms standard LDA by 43%, 37.2%, and 9.6%, respectively.

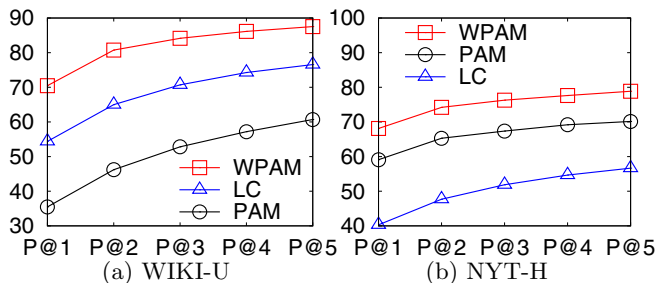


Figure 2: Precision@k results for WPAM, PAM, and LC.

Figure 2 shows Precision@k results, for $k = 1, \dots, 5$, for WPAM, PAM, and LC. We use SVM scores and likelihoods to rank for LC and WPAM, respectively. Figure 2(a) shows that P@5 can reach as high as 87.5%, which is achieved by WPAM on WIKI-U. Also, P@5 for WPAM is almost 17.1% and 1.9% more than that for LDA(β, α) on WIKI-U and NYT-H, respectively (data not shown).

4.3 Hierarchy Pruning

The default Wikipedia hierarchy for all the entities in our dataset consists of 1225 non-leaf nodes, 58577 leaves, and 75910 paths. As mentioned in Section 3.2, not all of the nodes help WPAM, and hence it is necessary to prune the hierarchy. The pruning is done using the entropy-based method as described in Section 3, which not only uses the topic path assignments for document words, but also the performance of the learned WPAM model on a held-out dataset.

We performed pruning with three datasets, each being a 10% random extract of WIKI. Figure 3 depicts the variation of precision with number of nodes deleted from the hierarchy averaged across the splits. The performance of pruning reaches a maximum when the optimal hierarchy is reached, and then drops drastically upon further pruning. Since the performance does not increase monotonically towards the maximum, we use this steep drop as the stopping criterion for pruning.

The reason the performance reaches a maximum at the last stages of pruning is due to the nature of the Wikipedia

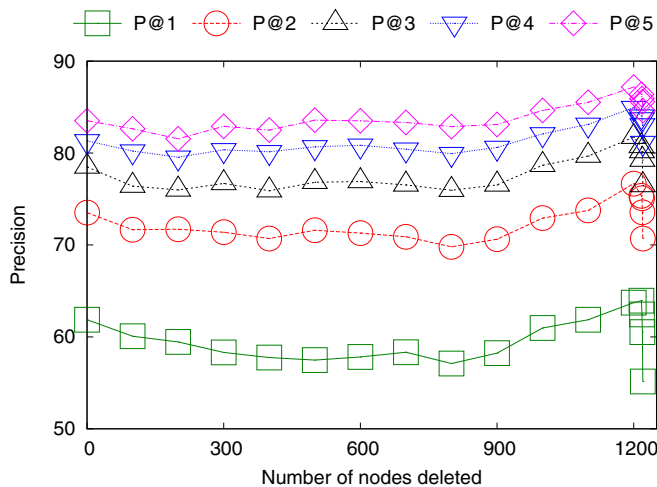


Figure 3: Variation of precision with pruning.

corpus and its corresponding hierarchy. The final pruned hierarchy consists of one root with 8 non-leaf nodes as children thus forming a three-level hierarchy (along with the entities forming the leaves). Hence, dynamic pruning deletes most of the non-leaf nodes (more than 1200) in the hierarchy to obtain optimal performance. Observe that the final pruned hierarchy has 6% higher precision compared to the original hierarchy.

4.4 Training and Test Times

Topic models are known to require excessive amounts of training time. Our code on full WIKI takes two and a half days to train. For a mid-to-large scale application such as ours, this issue exists with the baseline algorithms too. For example, [14] points out that it is not difficult to generate SVM learning problems for LC with millions of constraints. The number of constraints is a function of the number of annotations and entities in the training set. Training an SVM on full WIKI has 6,241,263 constraints even after we restrict the number of candidates per reference to 10 and takes almost a day (using the libsvm [10] JAVA implementation).

Labeling with WPAM’s Gibbs sampling algorithm, however, is much faster compared to the baseline. For our experiments, the labeling time of WPAM is roughly of the order of few tens of milliseconds per reference. In contrast, testing with the baseline models may require comparing each window with each entity in the extreme case costing around 0.2 sec of disambiguation time per reference. Hence labeling using topic models is approximately 10 times faster.

5. RELATED WORK

Due to its scale and coverage, Wikipedia has been the knowledge base of choice for most of the large-scale entity disambiguation approaches proposed in the literature [9, 11, 17, 19, 14]. The early Wikipedia-based disambiguation schemes [9, 17] resolved one reference at a time using only local context information. Subsequent works [11, 19, 14] propose to disambiguate all the references in a page collectively taking into account the relatedness of entities. Of these, [19] performs a limited form of collective disambiguation based on entity co-occurrence patterns – the entities selected to disambiguate the ambiguous references in a page

are the ones that co-occur frequently with (entities for) the unambiguous references in the page.

Unlike [19], [11] performs full-fledged collective disambiguation by also including ambiguous references in each disambiguation decision. It represents each entity as a feature vector consisting of local context information and categories for the entity. It then disambiguates a reference in a page with the entity whose feature vector matches best with the aggregated feature vector for the page (over all possible entity disambiguations for all references in the page). Clearly, a problem with the scheme of [11] is that the aggregate vector for a page may include irrelevant entities.

[14] formulates entity disambiguation as a joint optimization problem that seeks to collectively annotate all the references within a document so that the compatibility of each entity’s context with the local context of its reference and the co-occurrence between entity pairs is maximized. The optimization problem is NP-hard, and the authors resort to a hill-climbing strategy that greedily annotates references with entities that maximize the objective function.

We pointed out the shortcomings of the above-mentioned approaches in Section 1.2 – these include using a fixed-size window around a reference to define its local context (with no consensus on the best window size), and ignoring words that appear in the vicinity of un-annotated references when computing the context for an entity.

Our WPAM entity disambiguation model does not suffer from any of the above-mentioned problems since it is based on hierarchical topic models. Hierarchical models allow diverse sources of evidence like word-entity associations and entity co-occurrence patterns to be combined in a single unified framework for entity disambiguation. Moreover, topic models do not need to use difficult-to-set window parameters – their internal machinery can naturally use words from anywhere in the document, including the vicinity of un-annotated references, to learn high-quality word-entity mappings. WPAM differs from previously proposed hierarchical models like PAM [16] in two aspects: (1) WPAM uses the Wikipedia category hierarchy, and (2) Learning in WPAM is weakly semi-supervised with Wikipedia annotations serving as training data.

Word sense disambiguation and entity resolution are two related areas of research that bear close connections to entity disambiguation. Topic models have been used in both of these areas previously [5, 8, 25]. The main differentiating factor between entity disambiguation and such related fields is the assumption of the presence of a catalog of entities. Word sense disambiguation and entity resolution usually do not assume the presence of any such catalog and researchers in these fields have devoted most of their efforts to developing unsupervised algorithms. Thus, none of [5, 8] or [25] consider including prior annotations/labels for weakly semi-supervised learning the way we do in this paper.

Recently, there have been attempts to incorporate document labels into topic models [6, 15, 24]. Since our aim is to annotate words instead of documents, these approaches are quite different from the topic models we develop in this paper. [27] proposes the Multi-grain LDA topic model that uses local topics at the sentence level to capture ratable aspects like sound quality, battery life, etc. in user reviews. The Multi-grain LDA model is orthogonal to hierarchical topic models that instead model semantic relationships between topics typically at the document level. Similar to us,

[23] runs LDA on Wikipedia pages to discover hidden topics that are then used as additional features to classify short text segments. However, [23] only considers a few hundred coarse-grained topics, and does not exploit Wikipedia’s annotations or category hierarchy for topic inference.

6. CONCLUSION

In this paper, we proposed the weakly semi-supervised hierarchical topic model WPAM for disambiguating entities. WPAM uses all the words in a document, including those in the vicinity of un-annotated references, to learn high-quality word-entity associations. It leverages Wikipedia annotations to appropriately bias the assignment of entity labels to annotated words (and un-annotated words co-occurring with them), and the Wikipedia category hierarchy to capture entity context and co-occurrence patterns in a single unified disambiguation framework. We devised a scheme for pruning spurious categories with poorly correlated sub-categories in the hierarchy. In large-scale experiments with both held out subsets of Wikipedia and the NYT corpus, our WPAM model achieved over 70% disambiguation accuracy compared to about 54% for the existing state-of-the-art baselines. Promising directions for future work include exploring the use of other hierarchical topic models (e.g., hPAM [20]) for entity disambiguation, and learning good topic hierarchies from Wikipedia using non-parametric priors (e.g., hierarchical Dirichlet processes [26]).

7. REFERENCES

- [1] New York Times dataset. <http://www ldc.upenn.edu/Catalog/>.
- [2] Wikipedia. <http://www.wikipedia.org>.
- [3] Wikipedia category hierarchy. <http://en.wikipedia.org/wiki/Special:Categories>.
- [4] Wikipedia manual of style. http://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style.
- [5] I. Bhattacharya and L. Getoor. A latent dirichlet model for unsupervised entity resolution. In *SDM*, 2006.
- [6] D. Blei and J. McAuliffe. Supervised topic models. In *NIPS*, 2007.
- [7] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *JMLR*, 2003.
- [8] J. Boyd-Graber, D. Blei, and X. Zhu. A topic model for word sense disambiguation. In *EMNLP*, 2007.
- [9] R. Bunescu and M. Pasca. Using encyclopedic knowledge for named entity disambiguation. In *European Association for Computational Linguistics*, 2006.
- [10] C.-C. Chang and C.-J. Lin. LIBSVM – A Library for Support Vector Machines. <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>.
- [11] S. Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *EMNLP*, 2007.
- [12] T. Griffiths and M. Steyvers. Finding scientific topics. In *Proceedings of the National Academy of Sciences*, 2004.
- [13] G. Heinrich. Parameter estimation for text analysis. Technical report, 2005.
- [14] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of wikipedia entities in web text. In *KDD*, 2009.
- [15] S. Lacoste-Julien, F. Sha, and M. I. Jordan. DiscLDA: Discriminative learning for dimensionality reduction and classification. In *NIPS*, 2008.
- [16] W. Li and A. McCallum. Pachinko allocation: DAG-structured mixture models of topic correlations. In *ICML*, 2006.
- [17] R. Mihalcea and A. Csomai. Wikify! linking documents to encyclopedic knowledge. In *CIKM*, 2007.
- [18] D. J. Miller and H. S. Uyar. A mixture of experts classifier with learning based on both labelled and unlabelled data. In *NIPS*, 1997.
- [19] D. Milne and I. Witten. Learning to link with Wikipedia. In *CIKM*, 2008.
- [20] D. Mimno, W. Li, and A. McCallum. Mixtures of hierarchical topics with pachinko allocation. In *ICML*, 2007.
- [21] T. P. Minka. Estimating a dirichlet distribution. Technical report, Microsoft Research, 2003.
- [22] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Learning to classify text from labeled and unlabeled documents. In *AAAI*, 1998.
- [23] X.-H. Phan, L.-M. Nguyen, and S. Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *WWW*, 2008.
- [24] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *EMNLP*, 2009.
- [25] L. Shu, B. Long, and W. Meng. A latent topic model for complete entity resolution. In *ICDE*, 2009.
- [26] Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical dirichlet processes. *J. of the American Statistical Association*, 2005.
- [27] I. Titov and R. McDonald. Modeling online reviews with multi-grain topic models. In *WWW*, 2008.
- [28] J. Yedidia, W. Freeman, and Y. Weiss. Generalized belief propagation. In *NIPS*, 2000.