Efficient Tree-Based Topic Modeling

Yuening Hu
Department of Computer Science
University of Maryland, College Park
ynhu@cs.umd.edu

Jordan Boyd-Graber
iSchool and UMIACS
University of Maryland, College Park
jbg@umiacs.umd.edu

Abstract

Topic modeling with a tree-based prior has been used for a variety of applications because it can encode correlations between words that traditional topic modeling cannot. However, its expressive power comes at the cost of more complicated inference. We extend the SPARSELDA (Yao et al., 2009) inference scheme for latent Dirichlet allocation (LDA) to tree-based topic models. This sampling scheme computes the exact conditional distribution for Gibbs sampling much more quickly than enumerating all possible latent variable assignments. We further improve performance by iteratively refining the sampling distribution only when needed. Experiments show that the proposed techniques dramatically improve the computation time.

1 Introduction

Topic models, exemplified by latent Dirichlet allocation (LDA) (Blei et al., 2003), discover latent themes present in text collections. "Topics" discovered by topic models are multinomial probability distributions over words that evince thematic coherence. Topic models are used in computational biology, computer vision, music, and, of course, text analysis.

One of LDA's virtues is that it is a simple model that assumes a symmetric Dirichlet prior over its word distributions. Recent work argues for structured distributions that constrain clusters (Andrzejewski et al., 2009), span languages (Jagarlamudi and Daumé III, 2010), or incorporate human feedback (Hu et al., 2011) to improve the quality and flexibility of topic modeling. These models all use different tree-based prior distributions (Section 2).

These approaches are appealing because they preserve conjugacy, making inference using Gibbs sampling (Heinrich, 2004) straightforward. While straightforward, inference isn't cheap. Particularly for interactive settings (Hu et al., 2011), efficient inference would improve perceived latency.

SPARSELDA (Yao et al., 2009) is an efficient Gibbs sampling algorithm for LDA based on a refactorization of the conditional topic distribution (reviewed in Section 3). However, it is not directly applicable to tree-based priors. In Section 4, we provide a factorization for tree-based models within a broadly applicable inference framework that empirically improves the efficiency of inference (Section 5).

2 Topic Modeling with Tree-Based Priors

Trees are intuitive methods for encoding human knowledge. Abney and Light (1999) used tree-structured multinomials to model selectional restrictions, which was later put into a Bayesian context for topic modeling (Boyd-Graber et al., 2007). In both cases, the tree came from WordNet (Miller, 1990), but the tree could also come from domain experts (Andrzejewski et al., 2009).

Organizing words in this way induces *correlations* that are mathematically impossible to represent with a symmetric Dirichlet prior. To see how correlations can occur, consider the generative process. Start with a rooted tree structure that contains internal nodes and leaf nodes. This skeleton is a prior that generates K topics. Like vanilla LDA, these topics are distributions over words. Unlike vanilla LDA, their structure correlates words. Internal nodes have a distribution $\pi_{k,i}$ over children, where $\pi_{k,i}$ comes from per-node Dirichlet parameterized by β_i.[1] Each leaf node is associated with a word, and each word must appear in at least (possibly more than) one leaf node.

To generate a word from topic k, start at the root. Select a child $x_0 \sim \text{Mult}(\pi_{k,\text{ROOT}})$, and traverse the tree until reaching a leaf node. Then emit the leaf's associated word. This walk replaces the draw from a topic's multinomial distribution over words.

[1] Choosing these Dirichlet priors specifies the direction (i.e., positive or negative) and strength of correlations that appear.

The rest of the generative process for LDA remains the same, with θ, the per-document topic multinomial, and z, the topic assignment.

This tree structure encodes correlations. The closer types are in the tree, the more correlated they are. Because types can appear in multiple leaf nodes, this encodes polysemy. The path that generates a token is an additional latent variable we must sample.

Gibbs sampling is straightforward because the tree-based prior maintains conjugacy (Andrzejewski et al., 2009). We integrate the per-document topic distributions θ and the transition distributions π. The remaining latent variables are the topic assignment z and path l, which we sample jointly:[2]

$$p(z = k, l = \lambda | Z_-, L_-, w) \tag{1}$$
$$\propto (\alpha_k + n_{k|d}) \prod_{(i \to j) \in \lambda} \frac{\beta_{i \to j} + n_{i \to j|k}}{\sum_{j'} (\beta_{i \to j'} + n_{i \to j'|k})}$$

where $n_{k|d}$ is topic k's count in the document d; α_k is topic k's prior; Z_- and L_- are topic and path assignments excluding $w_{d,n}$; $\beta_{i \to j}$ is the prior for edge $i \to j$, $n_{i \to j|t}$ is the count of edge $i \to j$ in topic k; and j' denotes other children of node i.

The complexity of computing the sampling distribution is $O(KLS)$ for models with K topics, paths at most L nodes long, and at most S paths per word type. In contrast, for vanilla LDA the analogous conditional sampling distribution requires $O(K)$.

3 Efficient LDA

The SPARSELDA (Yao et al., 2009) scheme for speeding inference begins by rearranging LDA's sampling equation into three terms:[3]

$$p(z = k | Z_-, w) \propto (\alpha_k + n_{k|d}) \frac{\beta + n_{w|k}}{\beta V + n_{\cdot|k}} \tag{2}$$
$$\propto \underbrace{\frac{\alpha_k \beta}{\beta V + n_{\cdot|k}}}_{s_{\text{LDA}}} + \underbrace{\frac{n_{k|d} \beta}{\beta V + n_{\cdot|k}}}_{r_{\text{LDA}}} + \underbrace{\frac{(\alpha_k + n_{k|d}) n_{w|k}}{\beta V + n_{\cdot|k}}}_{q_{\text{LDA}}}$$

Following their lead, we call these three terms "buckets". A bucket is the *total* probability mass marginalizing over latent variable assignments (i.e., $s_{\text{LDA}} \equiv \sum_k \frac{\alpha_k \beta}{\beta V + n_{\cdot|k}}$, similarly for the other buckets). The three buckets are a smoothing only bucket s_{LDA}, document topic bucket r_{LDA}, and topic word bucket q_{LDA} (we use the "LDA" subscript to contrast with our method, for which we use the same bucket names without subscripts).

Caching the buckets' total mass speeds the computation of the sampling distribution. Bucket s_{LDA} is shared by all tokens, and bucket r_{LDA} is shared by a document's tokens. Both have simple constant time updates. Bucket q_{LDA} has to be computed specifically for each token, but only for the (typically) few types with non-zero counts in a topic.

To sample from the conditional distribution, first sample *which* bucket you need and then (and only then) select a topic *within* that bucket. Because the topic-term bucket q_{LDA} often has the largest mass and has few non-zero terms, this speeds inference.

4 Efficient Inference in Tree-Based Models

In this section, we extend the sampling techniques for SPARSELDA to tree-based topic modeling. We first factor Equation 1:

$$p(z = k, l = \lambda | Z_-, L_-, w) \tag{3}$$
$$\propto (\alpha_k + n_{k|d}) N_{k,\lambda}^{-1} [S_\lambda + O_{k,\lambda}].$$

Henceforth we call $N_{k,\lambda}$ the normalizer for path λ in topic k, S_λ the smoothing factor for path λ, and $O_{k,\lambda}$ the observation for path λ in topic k, which are

$$N_{k,\lambda} = \prod_{(i \to j) \in \lambda} \sum_{j'} (\beta_{i \to j'} + n_{i \to j'|k})$$
$$S_\lambda = \prod_{(i \to j) \in \lambda} \beta_{i \to j} \tag{4}$$
$$O_{k,\lambda} = \prod_{(i \to j) \in \lambda} (\beta_{i \to j} + n_{i \to j|k}) - \prod_{(i \to j) \in \lambda} \beta_{i \to j}.$$

Equation 3 can be rearranged in the same way as Equation 5, yielding buckets analogous to SPARSELDA's,

$$p(z = k, l = \lambda | Z_-, L_-, w) \tag{5}$$
$$\propto \underbrace{\frac{\alpha_k S_\lambda}{N_{k,\lambda}}}_{s} + \underbrace{\frac{n_{k|d} S_\lambda}{N_{k,\lambda}}}_{r} + \underbrace{\frac{(\alpha_k + n_{k|d}) O_{k,\lambda}}{N_{k,\lambda}}}_{q}.$$

Buckets sum both topics and paths. The sampling process is much the same as for SPARSELDA: select *which* bucket and then select a topic / path combination *within* the bucket (for a slightly more complex example, see Algorithm 1).

[2] For clarity, we omit indicators that ensure λ ends at $w_{d,n}$.

[3] To ease notation we drop the $_{d,n}$ subscript for z and w in this and future equations.

Recall that one of the benefits of SPARSELDA was that s was shared across tokens. This is no longer possible, as $N_{k,\lambda}$ is distinct for each path in tree-based LDA. Moreover, $N_{k,\lambda}$ is coupled; changing $n_{i \to j|k}$ in one path changes the normalizers of all cousin paths (paths that share some node i).

This negates the benefit of caching s, but we recover some of the benefits by splitting the normalizer to two parts: the "root" normalizer from the root node (shared by all paths) and the "downstream" normalizer. We precompute which paths share downstream normalizers; all paths are partitioned into cousin sets, defined as sets for which changing the count of one member of the set changes the downstream normalizer of other paths in the set. Thus, when updating the counts for path l, we only recompute $N_{k,l'}$ for all l' in the cousin set.

SPARSELDA's computation of q, the topic-word bucket, benefits from topics with unobserved (i.e., zero count) types. In our case, any non-zero path, a path with *any* non-zero edge, contributes.[4] To quickly determine whether a path contributes, we introduce an **edge-masked count** (EMC) for each path. Higher order bits encode whether edges have been observed and lower order bits encode the number of times the path has been observed. For example, if a path of length three only has its first two edges observed, its EMC is $\overline{11}000000$. If the same path were observed seven times, its EMC is $\overline{11}100111$. With this formulation we can ignore any paths with a zero EMC.

Efficient sampling with refined bucket While caching the sampling equation as described in the previous section improved the efficiency, the smoothing only bucket s is small, but computing the associated mass is costly because it requires us to consider all topics and paths. This is not a problem for *SparseLDA* because s is shared across all tokens. However, we can achieve computational gains with an upper bound on s,

$$s = \sum_{k,\lambda} \frac{\alpha_k \prod_{(i \to j) \in \lambda} \beta_{i \to j}}{\prod_{(i \to j) \in \lambda} \sum_{j'} (\beta_{i \to j'} + n_{i \to j'|k})}$$
$$\leq \sum_{k,\lambda} \frac{\alpha_k \prod_{(i \to j) \in \lambda} \beta_{i \to j}}{\prod_{(i \to j) \in \lambda} \sum_{j'} \beta_{i \to j'}} = s'. \quad (6)$$

A sampling algorithm can take advantage of this by not explicitly calculating s. Instead, we use s' as proxy, and only compute the exact s if we hit the bucket s' (Algorithm 1). Removing s' and always computing s yields the first algorithm in Section 4.

Algorithm 1 SAMPLING WITH REFINED BUCKET

```
for word w in this document do
    sample = rand() *(s' + r + q)
    if sample < s' then
        compute s
        sample = sample *(s + r + q)/(s' + r + q)
        if sample < s then
            return topic k and path λ sampled from s
        sample − = s
    else
        sample − = s'
    if sample < r then
        return topic k and path λ sampled from r
    sample − = r
    return topic k and path λ sampled from q
```

Sorting Thus far, we described techniques for efficiently computing buckets, but quickly sampling assignments within a bucket is also important. Here we propose two techniques to consider latent variable assignments in *decreasing* order of probability mass. By considering fewer possible assignments, we can speed sampling at the cost of the overhead of maintaining sorted data structures. We sort topics' prominence within a document (SD) and sort the topics and paths of a word (SW).

Sorting topics' prominence within a document (SD) can improve sampling from r and q; when we need to sample within a bucket, we consider paths in decreasing order of $n_{k|d}$.

Sorting path prominence for a word (SW) can improve our ability to sample from q. The edge-masked count (EMC), as described above, serves as a proxy for the probability of a path and topic. If, when sampling a topic and path from q, we sample based on the decreasing EMC, which roughly correlates with path probability.

5 Experiments

In this section, we compare the running time[5] of our sampling algorithm (FAST) and our algorithm with the refined bucket (RB) against the unfactored Gibbs sampler (NAÏVE) and examine the effect of sorting.

Our corpus has editorials from New York Times

[4] C.f. observed paths, where *all* edges are non-zero.

[5] Mean of five chains on a 6-Core 2.8-GHz CPU, 16GB RAM

Number of Topics				
	T50	T100	T200	T500
NAIVE	5.700	12.655	29.200	71.223
FAST	4.935	9.222	17.559	40.691
FAST-RB	2.937	4.037	5.880	8.551
FAST-RB-SD	2.675	3.795	5.400	8.363
FAST-RB-SW	2.449	3.363	4.894	7.404
FAST-RB-SDW	2.225	3.241	4.672	7.424
Vocabulary Size				
	V5000	V10000	V20000	V30000
NAÏVE	4.815	12.351	28.783	51.088
FAST	2.897	9.063	20.460	38.119
FAST-RB	1.012	3.900	9.777	20.040
FAST-RB-SD	0.972	3.684	9.287	18.685
FAST-RB-SW	0.889	3.376	8.406	16.640
FAST-RB-SDW	0.828	3.113	7.777	15.397
Number of Correlations				
	C50	C100	C200	C500
NAÏVE	11.166	12.586	13.000	15.377
FAST	8.889	9.165	9.177	8.079
FAST-RB	3.995	4.078	3.858	3.156
FAST-RB-SD	3.660	3.795	3.593	3.065
FAST-RB-SW	3.272	3.363	3.308	2.787
FAST-RB-SDW	3.026	3.241	3.091	2.627

Table 1: The average running time per iteration (S) over 100 iterations, averaged over 5 seeds. Experiments begin with 100 topics, 100 correlations, vocab size 10000 and then vary one dimension: number of topics (top), vocabulary size (middle), and number of correlations (bottom).

from 1987 to 1996.[6] Since we are interested in varying vocabulary size, we rank types by average tf-idf and choose the top V. WordNet 3.0 generates the correlations between types. For each synset in WordNet, we generate a subtree with all types in the synset—that are also in our vocabulary—as leaves connected to a common parent. This subtree's common parent is then attached to the root node.

We compared the FAST and FAST-RB against NAÏVE (Table 1) on different numbers of topics, various vocabulary sizes and different numbers of correlations. FAST is consistently faster than NAÏVE and FAST-RB is consistently faster than FAST. Their benefits are clearer as distributions become sparse (e.g., the first iteration for FAST is slower than later iterations). Gains accumulate as the topic number increases, but decrease a little with the vocabulary size. While both sorting strategies reduce time, sorting topics and paths for a word (SW) helps more than sorting topics in a document (SD), and combining the two is (with one exception) better than either alone.

[6] 13284 documents, 41554 types, and 2714634 tokens.

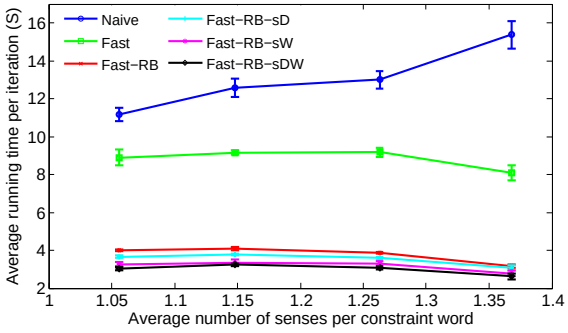

Figure 1: The average running time per iteration against the average number of senses per correlated words.

As more correlations are added, NAÏVE's time increases while that of FAST-RB decreases. This is because the number of non-zero paths for uncorrelated words decreases as more correlations are added to the model. Since our techniques save computation for every zero path, the overall computation decreases as correlations push uncorrelated words to a limited number of topics (Figure 1). Qualitatively, when the synset with "king" and "baron" is added to a model, it is associated with "drug, inmate, colombia, waterfront, baron" in a topic; when "king" is correlated with "queen", the associated topic has "king, parade, museum, queen, jackson" as its most probable words. These represent reasonable disambiguations. In contrast to previous approaches, inference speeds up as topics become more semantically coherent (Boyd-Graber et al., 2007).

6 Conclusion

We demonstrated efficient inference techniques for topic models with tree-based priors. These methods scale well, allowing for faster exploration of models that use semantics to encode correlations without sacrificing accuracy. Improved scalability for such algorithms, especially in distributed environments (Smola and Narayanamurthy, 2010), could improve applications such as cross-language information retrieval, unsupervised word sense disambiguation, and knowledge discovery via interactive topic modeling.

Acknowledgments

We would like to thank David Mimno and the anonymous reviewers for their helpful comments. This work was supported by the Army Research Laboratory through ARL Cooperative Agreement W911NF-09-2-0072 and NSF grant #1018625. Any opinions or conclusions expressed are the authors' and do not necessarily reflect those of the sponsors.

References

Steven Abney and Marc Light. 1999. Hiding a semantic hierarchy in a Markov model. In *Proceedings of the Workshop on Unsupervised Learning in Natural Language Processing*.

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via Dirichlet forest priors. In *Proceedings of International Conference of Machine Learning*.

David M. Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Jordan Boyd-Graber, David M. Blei, and Xiaojin Zhu. 2007. A topic model for word sense disambiguation. In *Proceedings of Emperical Methods in Natural Language Processing*.

Gregor Heinrich. 2004. Parameter estimation for text analysis. Technical report. http://www.arbylon.net/publications/text-est.pdf.

Yuening Hu, Jordan Boyd-Graber, and Brianna Satinoff. 2011. Interactive topic modeling. In *Association for Computational Linguistics*.

Jagadeesh Jagarlamudi and Hal Daumé III. 2010. Extracting multilingual topics from unaligned corpora. In *Proceedings of the European Conference on Information Retrieval (ECIR)*.

George A. Miller. 1990. Nouns in WordNet: A lexical inheritance system. *International Journal of Lexicography*, 3(4):245–264.

Alexander J. Smola and Shravan Narayanamurthy. 2010. An architecture for parallel topic models. *International Conference on Very Large Databases*, 3.

Limin Yao, David Mimno, and Andrew McCallum. 2009. Efficient methods for topic model inference on streaming document collections. In *Knowledge Discovery and Data Mining*.