Towards Interactive Construction of Topical Hierarchy: A Recursive Tensor Decomposition Approach

Chi Wang[†], Xueqing Liu[‡], Yanglei Song[‡], Jiawei Han[‡]
[†]Microsoft Research. Redmond, WA 98007
[‡]University of Illinois at Urbana-Champaign. Urbana, IL, 61801
chiw@microsoft.com, {xliu93, ysong44, hanj}@illinois.edu

ABSTRACT

Automatic construction of user-desired topical hierarchies over large volumes of text data is a highly desirable but challenging task. This study proposes to give users freedom to construct topical hierarchies via interactive operations such as expanding a branch and merging several branches. Existing hierarchical topic modeling techniques are inadequate for this purpose because (1) they cannot consistently preserve the topics when the hierarchy structure is modified; and (2) the slow inference prevents swift response to user requests. In this study, we propose a novel method, called STROD, that allows efficient and consistent modification of topic hierarchies, based on a recursive generative model and a scalable tensor decomposition inference algorithm with theoretical performance guarantee. Empirical evaluation shows that STROD reduces the runtime of construction by several orders of magnitude, while generating consistent and quality hierarchies.

Categories and Subject Descriptors

I.7 [**Computing Methodologies**]: Document and Text Processing; H.2.8 [**Database Applications**]: Data Mining

Keywords

Topic Modeling, Ontology Learning, Interactive Data Exploration, Tensor Decomposition

1. INTRODUCTION

Constructing a topic hierarchy for large text collection, such as business documents, news articles, social media messages, and research publications, is helpful for information workers, data analysts and researchers to summarize and navigate them in multiple granularity efficiently. While existing hierarchical topic models can be used to produce such hierarchies as an exploration tool, they still require human curation (*e.g.*, modify the structure and label the topics) to meet the quality requirement for reliable exploitation. The manual work for curation is very expensive. This work focuses on helping with the structure modification task.

The nature of this task is interactive and iterative. On one hand, people use a topic model to explore a dataset when the topics are unknown *a priori*. Thus it is hard to determine the best shape of the hierarchy upfront. On the other hand, as they see the results (inferred topics even with imperfect structure), people have ideas about a more desirable structure, *e.g.*, one topic should be expanded, or multiple topics should be merged. Then they may want to modify part of the hierarchy but preserve other parts that already look good to be labeled. Some modification, such as expanding a topic, is again exploratory and needs help from the machine. It takes multiple iterations of human investigation and algorithm run to finish the construction.

To enable interactive construction of the topic hierarchy, *i.e.*, allowing users to modify the structure on the go, the system needs to satisfy two conditions: *efficiency* and *consistency*. Efficiency is necessary for users to see results quickly and react before they lose the context. Consistency is necessary for confusion-free modification, and has two-fold meanings: when people want to modify certain parts of the hierarchy, the remaining parts should be preserved after each run (single-run consistency); and a system should output undifferentiated results given identical input in multiple runs (multi-run consistency).

Limitation of prior work. Most existing hierarchical topic modeling techniques [10, 17, 20, 14, 2] are based on the extensions of latent Dirichlet allocation (LDA), and are not designed for interactive construction of the hierarchy. First, the inference algorithms for these models are expensive, demanding hundreds or thousands of passes of data. Second, an inference algorithm generates one hierarchy for one corpus in one run of the algorithm. Running the inference algorithm with a slightly modified hierarchy structure does not guarantee preservation of topics in untouched branches. Rerunning the inference algorithm with the same input may result in very different results. Therefore, both single-run and multi-run consistency conditions are violated if we use them for interaction.

Our solution. We consider a strategy of top-down, progressive construction of a topical hierarchy, instead of inferring a complex hierarchical model all at once. Thus the construction can be done via a series of interactive operations, such as expanding a topic, collapsing a topic, merging topics and removing topics. Efficient and consistent algorithms can then be designed for each operation. Users can see the results after each operation, and decide what opera-

KDD'15, August 10-13, 2015, Sydney, NSW, Australia.

DOI: http://dx.doi.org/10.1145/2783258.2783288.

tion to take next. This strategy has several benefits: users can easily control the complexity of the hierarchy; users can see intermediate results and curate the hierarchy in early stages; and it is easier for curators to focus on one simple operation a time.

To support these interactive operations in an efficient and consistent manner, we resort to moment-based inference. Simply put, moment-based inference compresses the original data by collecting important statistics from the documents, *e.g.*, term co-occurrences, and uses these statistics to infer topics. For one advantage, the inference based on the compressed information avoids the expensive, numerous passes of the data. For another advantage, the compression reduces randomness in the data by aggregation. With careful choice of the statistics and the inference method, we can uncover the topics with theoretical guarantee. Modifications to the hierarchy can be supported by manipulating the moments.

We establish a top-down hierarchy construction framework STROD based on these ideas. To the best of our knowledge, it is the first framework towards interactive topical hierarchy construction. The following summarizes our main contributions:

- We propose a new hierarchical topic model such that the modification operations mentioned above can be achieved by several atomic operators to the model.
- We develop a scalable tensor-based recursive orthogonal decomposition (STROD) method for efficient and consistent construction.
- Our experiments demonstrate that our method is several orders of magnitude more efficient than the alternatives, while generating consistent, quality topic hierarchy that is comprehensible to users.

2. RELATED WORK

Statistical topic modeling techniques model a document as a mixture of multiple topics, while every topic is modeled as a distribution over terms. Two important models are probabilistic latent semantic analysis (PLSA) [13] and its Bayesian extension latent Dirichlet allocation (LDA) [5]. They model the generative processes of each term from each document in a corpus, and then infer the unknown distributions that best explain the observed documents.

Hierarchical topic models follow the same generative spirit. Instead of having a pool of flat topics, these models assume an internal hierarchical structure of the topics. Different models use different generative processes to simulate this hierarchical structure, such as nested Chinese Restaurant Process [10], Pachinko Allocation [17], hierarchical Pachinko Allocation [20], recursive Chinese Restaurant Process [14], and nested Chinese Restaurant Franchise [2]. When these models are applied to constructing a topical hierarchy, the entire hierarchy is inferred all at once from the corpus.

The main inference methods for these topic models can be divided into two categories: MCMC sampling [11] and variational inference [5]. They are essentially approximation of the Maximum Likelihood (ML) principle (including its Bayesian version *maximum a posterior*): Find the best parameters that maximize the joint probability specified by a model. There has been a substantial amount of work on speeding up LDA inference, *e.g.*, by leveraging sparsity [22, 30, 16] and parallelization [21, 24, 31], or online learning mechanism [1, 12, 8]. Few of these ideas have been adopted by the hierarchical topic model studies.

These inference methods have no theoretical guarantee of convergence within a bounded number of iterations, and are nondeterministic either due to the sampling or the random initialization. Recently, a new inference method for LDA has been proposed based on a *method of moments*, rather than ML. It is found to have provable error bound and convergence properties in theory [3].

All of the hierarchical topic models follow the bag-of-words assumption, while some other extensions of LDA have been developed to model sequential n-grams to achieve better interpretability [26, 29, 18]. No one has integrated them in a hierarchical topic model. The efficiency and consistency issues will become more challenging in an integrated model. A practical approach is to decouple the topic modeling part and the phrase mining part. Blei and Lafferty [4] have proposed to use a statistical test to find topical phrases, which is time-consuming. A much less expensive heuristic is studied in recent work [6] and shown to be effective.

There are a few alternative approaches to constructing a topical hierarchy. Pujara and Skomoroch [23] proposed to first run LDA on the entire corpus, and then split the corpus heuristically according to the results and run LDA on each split corpus individually. CATHY [28] is a recursive topical phrase mining framework for short, content-representative text. It also decouples phrase mining and topic discovery for efficiency purpose. Though it is not designed for generic text, it bears some similarity with this work such as top-down recursion and compression of documents.

After the hierarchy is constructed from a corpus, people can label these topics and derive topic distributions for each document [25]. Those are not the subject of this paper. Broadly speaking, this work is also related to: hierarchical clustering of documents [9], queries [19], keywords [28] *etc.*; and ontology learning [15], which mines subsumption ('is-a') relationships from text.

3. PROBLEM FORMULATION

Given a corpus, our goal is to construct a user-desired topical hierarchy, *i.e.*, a tree of topics, where each child topic is about a more specific theme within the parent topic.

For easy interaction, the topics need to be visualized in user-friendly forms. Unigrams are often ambiguous, especially across fine-grained topics [27]. We choose to enhance the topic representation with ranked phrases. The ranking should reflect both their popularity and discriminating power for a topic. For example, the top ranked phrases for the database topic can be: "database systems", "query processing", "concurrency control" .... A phrase can appear in multiple topics, though it will have various ranks in them.

Formally, the input data are a corpus of D documents. The d-th document can be segmented into a sequence of l_d tokens. All the unique tokens in this corpus are indexed using a vocabulary of V terms. And $w_{d,j} \in [V], j = 1, \ldots, l_d$ represents the index of the j-th token in document d. A topic t is defined by a probability distribution over terms $\phi_t \in \Delta^{V-1}$, and an ordered list of phrases $\mathcal{P}_t = \{P_{t,1}, P_{t,2}, \ldots\}$, where $P_{t,i}$ is the phrase ranked at i-th position for topic t.

A topical hierarchy is defined as a tree $\mathcal{T}$ in which each node is a topic. Every non-leaf topic t has C_t child topics. We assume C_t is bounded by a small number K, such as 10, because the topical hierarchy is intended for human to

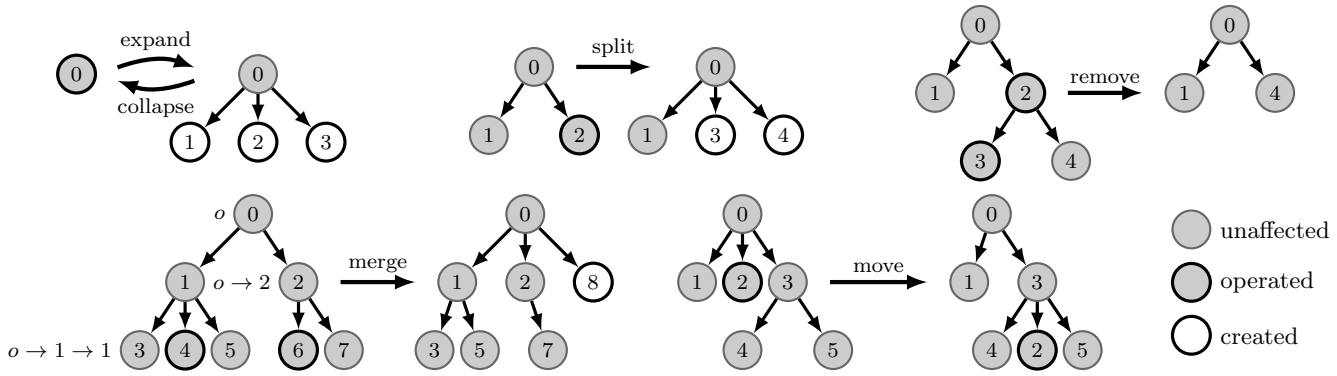


Figure 1: Examples of 6 user operations. A topic can be indexed by a integer (in the circle), or by the path from root

efficiently browse the subtopics for each topic. The number K is named the *width* of the tree $\mathcal{T}$.

A topical hierarchy for a corpus is constructed via a series of user operations. An operation transforms one topic hierarchy $\mathcal{T}$ to another $\mathcal{T}'$. A top-down construction framework supports the following user operations.

1. Expand – for an arbitrary topic t in $\mathcal{T}$, grow a subtree rooted at t.
2. Collapse – for an arbitrary topic t in $\mathcal{T}$, remove all its descendant topics.
3. Split – for an arbitrary topic t in $\mathcal{T}$, split it into k topics.
4. Remove – for an arbitrary set of topics $t_1, \ldots, t_n$ in $\mathcal{T}$, delete these topics.
5. Merge – for an arbitrary set of topics $t_1, \ldots, t_n$ in $\mathcal{T}$, merge these topics as a new topic, whose parent is the least common ancestor of them, and whose children are the union of the children of all merged topics.
6. Move – for an arbitrary topic t in $\mathcal{T}$, move the subtree rooted at t to be under a different parent topic t'.

Figure 1 demonstrates these operations. In these operations, only a few topics are affected, so users can consistently modify the hierarchy and control the change.

For convenience, we index a topic using the top-down path from root to this topic. The root topic is indexed as o. Every non-root topic t is recursively indexed by $\pi_t \to \chi_t$, where π_t is the path index of its parent topic, and $\chi_t \in [C_t]$ is the index of t among its siblings. For example, topic 2 in the 'merge' example of Figure 1 is indexed as $o \to 2$, and topic 3 in the same tree is indexed as $o \to 1 \to 1$. The *level* h_t of a topic t is defined to be its distance to the root. So root topic is in level 0, and topic $o \to 1 \to 1$ is in level 2. The *height* H of a tree is defined to be the maximal level over all the topics in the tree. Clearly, the total number T of topics is upper bounded by $\frac{K^{H+1}-1}{K-1}$.

4. THE STROD FRAMEWORK

We develop a Scalable Tensor Recursive Orthogonal Decomposition (STROD) framework for interactive topical hierarchy construction. In Section 4.1, we propose a new hierarchical topic model, and introduce how the user operations can be achieved by atomic manipulations to the model. In Section 4.2, we present our tensor-based algorithms supporting these operations. Section 4.3 introduces topical phrase mining and ranking based on the inferred model parameters.

4.1 Hierarchical Topic Modeling

Generative hierarchical topic modeling assumes the documents are generated from a latent variable model, and then infers the model parameters from observed documents to recover the topics. Distinct from prior work, we do not infer a hierarchy for one corpus only once. Instead, we allow users to perform consistent modification to the hierarchy. Therefore, we need a model that is convenient for manipulation and supports all the user operations introduced in Section 3.

We first introduce our generative model when the hierarchy structure is fixed, and then discuss atomic operators to manipulate the model structure.

4.1.1 Latent Dirichlet Allocation with Topic Tree

In this subsection we assume the topic hierarchy structure is fixed. Its height is H, and there are τ leaf nodes and $T-\tau$ non-leaf nodes. For ease of explanation we assume all leaf nodes are on the level of H.

Every leaf topic node $t(C_t = 0)$ has a multinomial distribution $\phi_t = p(w = \cdot|t)$ over terms. Every document d paired with a non-leaf node $t(C_t > 0)$ has a multinomial distribution $\theta_{d,t} = p(w = \cdot|d, t)$ over t's child topics: $t \to 1$ through $t \to C_t$. $\theta_{d,t}$ represents the content bias of document d towards t's subtopics. For the 'merge' example in Figure 1, before merge, there are 3 non-leaf topics: $o, o \to 1$ and $o \to 2$. So a document d is associated with 3 multinomial distributions over topics: $\theta_{d,o}$ over its 2 children, $\theta_{d,o\to1}$ over its 3 children, and $\theta_{d,o\to2}$ over its 2 children. Each multinomial distribution $\theta_{d,t}$ is generated from a Dirichlet prior $(\alpha_{t\to1}, \ldots, \alpha_{t\to C_t})$. $\alpha_{t\to z}$ represents the corpus' bias towards z-th child of topic t, and $\alpha_t = \sum_{z=1}^{C_t} \alpha_{t\to z}$.

To generate a token $w_{d,j}$, we first sample a path from the root to a leaf node $o \to z^1_{d,j} \to z^2_{d,j} \to \cdots \to z^H_{d,j}$. The nodes along the path are sampled one by one, starting from the root. Each time one child $z^i_{d,j}$ is selected from all children of $o \to z^1_{d,j} \to \cdots \to z^{i-1}_{d,j}$, from the multinomial $\theta_{d,o\to z^1_{d,j}\to\cdots\to z^{i-1}_{d,j}}$. When a leaf node is reached, the token is generated from its multinomial distribution $\phi_{o\to z^1_{d,j}\to z^2_{d,j}\to\cdots\to z^H_{d,j}}$.

The whole generative process is:

1. For each leaf node t in $\mathcal{T}$, generate its distribution over terms $\phi_t \sim Dir(\beta)$;
2. For each document $d \in [D]$:
   (a) For each non-leaf node t in $\mathcal{T}$, draw a multinomial distribution over its subtopics: $\theta_{d,t} \sim Dir(\alpha_{t\to1}, \ldots, \alpha_{t\to C_t})$;

Table 1: Notations used in our model

Symbol	Description
D	the number of documents in the corpus
V	the number of unique terms in the corpus
H	the height of the topical hierarchy
T	the total number of topics in the hierarchy
τ	the number of leaf topics in the hierarchy
C_t	the number of child topics of topic t
l_d	the length (number of tokens) of document d
π_t	the parent topic of topic t
$\chi_t \in [C_{\pi_t}]$	the index of topic t among its siblings
$w_{d,j} \in [V]$	the j-th token in the document d
$z^i_{d,j}$	the child index of the topic at level i for $w_{d,j}$
ϕ_t	topic t's multinomial distribution over terms
α_t	the Dirichlet hyperparameter of topic t
$\theta_{d,t}$	document d's distribution over t's child topics

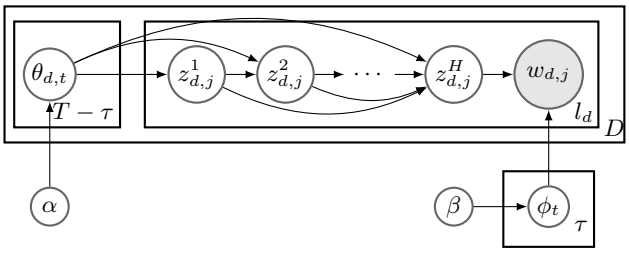


Figure 2: Latent Dirichlet Allocation with Topic Tree

(b) For each token index $j \in [l_d]$ of document d:

i. $i \leftarrow 0$;

ii. While $o \to z^1_{d,j} \to \cdots \to z^i_{d,j}$ is not a leaf node:

A. $i \leftarrow i+1$;

B. Draw subtopic $z^i_{d,j} \sim Multi(\theta_{d,o\to z^1_{d,j}\to\cdots\to z^{i-1}_{d,j}})$;

iii. Generate token $w_{d,j} \sim Multi(\phi_{o\to z^1_{d,j}\to\cdots\to z^i_{d,j}})$.

Its graphical representation is Figure 2. Table 1 collects the notations.

For every non-leaf topic node, we can derive a term distribution by marginalizing their children's term distributions:

$$\phi_t = p(w=\cdot|t) = \sum_{z=1}^{C_t} p(t\to z|t)p(w=\cdot|t\to z) = \sum_{z=1}^{C_t} \frac{\alpha_{t\to z}}{\alpha_t}\phi_{t\to z} \tag{1}$$

So in our model, the term distribution ϕ_t for an internal node in the topic hierarchy can be calculated as a mixture of its children's term distributions. The Dirichlet prior α determines the mixing weight.

When the structure $\mathcal{T}$ is fixed, we need to infer its parameters $\phi(\mathcal{T})$ and $\alpha(\mathcal{T})$ from a given corpus. When the height of the hierarchy $H=1$, our model reduces to the flat LDA model.

4.1.2 Model Structure Manipulation

The main advantage of this model is that it can be consistently manipulated to accommodate user operations.

PROPOSITION 1. *The following atomic manipulation operators are sufficient in order to compose all the user operations introduced in Section 3:*

- $\mathsf{EXP}(t,k)$. *Discover k subtopics of a leaf topic t.*
- $\mathsf{MER}(t_1,t_2)$. *Merge two topics t_1 and t_2 into a new topic t_3 under their least common ancestor t.*
- $\mathsf{MOV}(t_1,t_2)$. *Move the subtree rooted at topic t_1 to be under t_2.*

The following are examples about how to use these manipulation operators to compose the user operations in Figure 1.

- 'Collapse' – applying $\mathsf{MER}(o, o\to 1)$ three times.
- 'Split' – $\mathsf{EXP}(o\to 2, 2)$ followed by $\mathsf{MER}(o, o\to 2)$.
- 'Remove' – $\mathsf{MER}(o\to 2, o\to 2\to 1)$ followed by $\mathsf{MER}(o, o\to 2)$.

Implementation of these atomic operators needs to follow the consistency requirement.

1. Single-run consistency – suppose the topical hierarchy $\mathcal{T}_1$ is altered into $\mathcal{T}_2$ after a user operation, certain nodes are not affected. For example, in the 'merge' operation in Figure 1, node 0,1,2,3,5,7 are not touched. The consistency condition requires that, if we restart step 2-(b) whenever we reach an affected node in step 2-(b)-ii, $\mathcal{T}_1$ and $\mathcal{T}_2$ are equivalent generative models, *i.e.*, generate the same documents in expectation. By this definition, we have the following proposition.

   PROPOSITION 2. *A single run altering $\mathcal{T}_1$ into $\mathcal{T}_2$ is consistent if and only if i) for each unaffected leaf node t, $\alpha_t(\mathcal{T}_1)=\alpha_t(\mathcal{T}_2), \phi_t(\mathcal{T}_1)=\phi_t(\mathcal{T}_2)$; and ii) for each internal node t', $\alpha_{t'}(\mathcal{T}_2)=\sum_{z=1}^{C_{t'}}\alpha_{t'\to z}(\mathcal{T}_2)$.*

2. Multi-run consistency – with identical input across multiple runs, one operator should output nearly identical (undifferentiated to human) α and ϕ.

Section 4.2 presents a moment-based method to compute these operators efficiently and consistently.

4.2 Moment-based Operation

In statistics, the ξ-th order *population moment* of a random variable is the expectation of its ξ-th power. In our problem, the random variable is a token $w_{d,j}$ in a document d. The ξ-th population moment is the expected co-occurrence of terms in ξ token positions. They are related to the model parameters α and ϕ. The method of moments collects empirical moments from the corpus, and estimate α and ϕ by fitting the empirical moments with theoretical moments. As a computational advantage, it only relies on the term co-occurrence statistics. The statistics contain important information compressed from the full data, and require only a few scans of the data to collect.

To compute our three atomic operators, we generalize the notion of population moments. We consider the population moments *conditioned* on a topic t. The first order conditional moment $E_1(t)$ is a vector in $\mathbb{R}^V$. Component x is the expectation of $1_{w=x}$ given that w is drawn from topic t's descendant.

$$E_1(t) = p(w=\cdot|t,\alpha) = \phi_t = \sum_{z=1}^{C_t}\frac{\alpha_{t\to z}}{\alpha_t}\phi_{t\to z} \tag{2}$$

The second order moment $E_2(t) \in \mathbb{R}^{V\times V}$ is a $V\times V$ tensor (hence, a matrix), storing the expectation of the co-occurrences of two terms w_1 and w_2 given that they are

both drawn from topic t's descendants. Integrating over the document-topic distribution θ, we have:

$$E_2(t) = p(w_1 = \cdot, w_2 = \cdot | t, t, \alpha) \tag{3}$$
$$= \sum_{z_1 \neq z_2} \frac{\alpha_{t\to z_1}\alpha_{t\to z_2}}{\alpha_t(\alpha_t+1)} \phi_{t\to z_1} \otimes \phi_{t\to z_2} + \sum_{z=1}^{C_t} \frac{\alpha_{t\to z}(\alpha_{t\to z}+1)}{\alpha_t(\alpha_t+1)} \phi_{t\to z}^{\otimes 2}$$

The operator $\otimes$ denotes an outer product between tensors: if $A \in \mathbb{R}^{s_1 \times \cdots \times s_p}$, and $B \in \mathbb{R}^{s_{p+1} \times \cdots \times s_{p+q}}$, then $A \otimes B$ is a tensor in $\mathbb{R}^{s_1 \times \cdots \times s_{p+q}}$, and $[A \otimes B]_{i_1 \ldots i_{p+q}} = A_{i_1 \ldots i_p} B_{i_{p+1} \ldots i_{p+q}}$.

Likewise, we can derive the third order moment $E_3(t) \in \mathbb{R}^{V \times V \times V}$ (a $V \times V \times V$ tensor) as the expectation of co-occurrences of three terms w_1, w_2 and w_3 given that they are all drawn from topic t's descendants:

$$\begin{aligned} &E_3(t) = p(w_1 = \cdot, w_2 = \cdot, w_3 = \cdot | t, t, t, \alpha) \\ &= \sum_{z_1 \neq z_2 \neq z_3 \neq z_1} \frac{\alpha_{t\to z_1}\alpha_{t\to z_2}\alpha_{t\to z_3}}{\alpha_t(\alpha_t+1)(\alpha_t+2)} \phi_{t\to z_1} \otimes \phi_{t\to z_2} \otimes \phi_{t\to z_3} \\ &+ \sum_{z_1 \neq z_2} \frac{\alpha_{t\to z_1}\alpha_{t\to z_2}(\alpha_{t\to z_1}+1)}{\alpha_t(\alpha_t+1)(\alpha_t+2)} (\phi_{t\to z_1} \otimes \phi_{t\to z_1} \otimes \phi_{t\to z_2} \\ &+ \phi_{t\to z_1} \otimes \phi_{t\to z_2} \otimes \phi_{t\to z_1} + \phi_{t\to z_2} \otimes \phi_{t\to z_1} \otimes \phi_{t\to z_1}) \\ &+ \sum_{z=1}^{C_t} \frac{\alpha_{t\to z}(\alpha_{t\to z}+1)(\alpha_{t\to z}+2)}{\alpha_t(\alpha_t+1)(\alpha_t+2)} \phi_{t\to z}^{\otimes 3} \end{aligned} \tag{4}$$

Equations (2)–(4) characterize the theoretical conditional moments for topic t using model parameters associated with t's children. The empirical conditional moments can be estimated from data and parameters of t's ancestors.

For topic t, we estimate the empirical 'topical' count of term x in document d as:

$$c_{d,x}(t) = c_{d,x} p(t|x) = c_{d,x}(\pi_t) \frac{\alpha_t \phi_{t,x}}{\sum_{z=1}^{C_{\pi_t}} \alpha_{\pi_t \to z} \phi_{\pi_t \to z, x}} \tag{5}$$

Recall that π_t is t's parent. $c_{d,x}(t)$ can be recursively computed through $c_{d,x}(\pi_t)$ and the boundary is $c_{d,x}(o) = c_{d,x}$, *i.e.*, the total counts of term x in document d.

Then we can estimate empirical conditional moments using these empirical topical counts:

$$\begin{aligned} E_1(t) &= \sum_{d=1}^{D} \frac{1}{l_d(t)} c_d(t) \\ E_2(t) &= \sum_{d=1}^{D} \frac{1}{l_d(t)(l_d(t)-1)} [c_d(t) \otimes c_d(t) - diag(c_d(t))] \end{aligned} \tag{6}$$

where $l_d(t) = \sum_{x=1}^{V} c_{i,x}(t)$. These enable fast estimation of empirical moments by passing data once.

The following three subsections discuss the computation of the three atomic operators EXP, MER and MOV with the method of moments.

4.2.1 EXP *Operator*

EXP(t, k) should find k subtopics under topic t, without changing any existing model parameters. So we need an algorithm that returns $(\alpha_{t\to z}, \phi_{t\to z}), z \in [k]$, with $\sum_{z=1}^{k} \alpha_{t\to z} = \alpha_t$. By recursion, we note that only α_o needs to be set by a user. It controls the degree of topical purity of documents. When $\alpha_o \to \infty$, each document is only about one leaf topic.

We employ the method of moments. In Equations (2)–(4), we replace the left hand side with the empirical conditional moments estimated from the data. The right hand side is theoretical moments with $\alpha_{t\to z}, \phi_{t\to z}, z \in [k]$ as unknown variables. Solving these equations yields a solution of the acquired model parameters. The following theorem by Anandkumar *et al.* [3] suggests that we only need to use up to 3rd order moments to find the solution.

THEOREM 1. *Assume M_2 and M_3 are defined as:*

$$M_2 = \sum_{z=1}^{k} \lambda_z v_z^{\otimes 2}, M_3 = \sum_{z=1}^{k} \lambda_z v_z^{\otimes 3} \tag{7}$$

where $\lambda_z > 0, v_z$'s are linearly independent, and $\|v_z\| = 1$. When M_2 and M_3 are given, v_z and λ_z in Equation (7) can be uniquely solved in polynomial time.

To write Equations (2)–(4) in this form, we define:

$$M_2(t) = (\alpha_t + 1) E_2(t) - \alpha_t E_1(t)^{\otimes 2} \tag{8}$$

$$\begin{aligned} U_1(t) &= E_2(t) \otimes E_1(t), \\ U_2(t) &= \Omega(U_1(t), 1, 3, 2), U_3(t) = \Omega(U_1(t), 2, 3, 1) \end{aligned} \tag{9}$$

$$\begin{aligned} M_3(t) = &\frac{(\alpha_t+1)(\alpha_t+2)}{2} E_3(t) + \alpha_t^2 E_1^{\otimes 3} \\ &- \frac{\alpha_t(\alpha_t+1)}{2} [U_1(t) + U_2(t) + U_3(t)] \end{aligned} \tag{10}$$

where $\Omega(A, a, b, c)$ permutes the modes of tensor A, such that $\Omega(A, a, b, c)_{i_1, i_2, i_3} = A_{i_a, i_b, i_c}$. It follows that:

$$M_2(t) = \sum_{z=1}^{C_t} \frac{\alpha_{t\to z}}{\alpha_t} \phi_{t\to z}^{\otimes 2}, M_3(t) = \sum_{z=1}^{C_t} \frac{\alpha_{t\to z}}{\alpha_t} \phi_{t\to z}^{\otimes 3}$$

So they fit Equation (7) nicely, and intuitively. If we decompose $M_2(t)$ and $M_3(t)$, the z-th component is determined by the child's term distribution $\phi_{t\to z}$, and its weight is $\frac{\alpha_{t\to z}}{\alpha_t}$, which is equal to $p(t \to z|t)$.

$M_2(t)$ is a dense $V \times V$ matrix, and $M_3(t)$ is a dense $V \times V \times V$ tensor. Direct application of the tensor decomposition algorithm in [3] is challenging due to the creation of these huge dense tensors. Therefore, we design a more scalable algorithm. The idea is to bypass the creation of $M_2(t)$ and $M_3(t)$ and utilize the *sparsity* and *decoupled decomposition* of the moments. We go over Algorithm 1 to explain it.

Line 1.1 collects the empirical moments with one scan of the data.

Lines 1.2 to 1.6 project the large tensor $M_3 \in \mathbb{R}^{V \times V \times V}$ into a smaller tensor $\widetilde{T} \in \mathbb{R}^{k \times k \times k}$. $\widetilde{T}$ is not only of smaller size, but also can be decomposed into an orthogonal form: $\widetilde{T} = \sum_{z=1}^{k} \widetilde{\lambda_z} \widetilde{v_z}^{\otimes 3}$. $\widetilde{v_z}, z \in [k]$ are orthonormal vectors in $\mathbb{R}^k$. This is assured by the whitening matrix W calculated in Line 1.5, which satisfies $W^T M_2 W = I$. This part contains two major tricks:

1. When calculating W, the straightforward computation requires spectral decomposition of M_2. We avoid explicit creation of M_2, but achieve the equivalent spectral decomposition. We first perform spectral decomposition for $E_2(t) = U \Sigma_1 U^T$, where $U \in \mathbb{R}^{V \times k}$ is the matrix of k eigenvectors, and $\Sigma_1 \in \mathbb{R}^{k \times k}$ is the diagonal eigenvalue matrix. The k column vectors of U form an orthonormal basis of the column space of $E_2(t)$. $E_1(t)$'s representation in this basis is $M_1 = U^T E_1(t)$. According to Equation (8), M_2 can now be written as:

$$M_2 = U[(\alpha_t + 1)\Sigma_1 - \alpha_t M_1 \otimes M_1] U^T$$

So a second spectral decomposition can be performed on $M_2' = (\alpha_t + 1)\Sigma_1 - \alpha_t M_1 \otimes M_1$, as $M_2' = U'\Sigma U'^T$. Then we have $UU'\Sigma(UU')^T$ as M_2's spectral decomposition. The space requirement is reduced from V^2 to $m = \|E_2(t)\|_0 \ll V^2$, because only term pairs ever co-occurring in one document contribute to non-zero elements of $E_2(t)$. The time for spectral decomposition is reduced from $\mathcal{O}(V^2K)$ to $\mathcal{O}(mK)$.

2. The straightforward computation of the tensor product $\widetilde{T} = M_3(t)(W, W, W)$ using explicit $M_3(t)$ and W requires $\mathcal{O}(V^3)$ space and $\mathcal{O}(V^3K + L\hat{l}^2)$ time, where $\hat{l}$ is the maximal document length. We decouple $M_3(t)$ as a summation of multiple tensors, such that the product between each tensor and W is in a decomposable form: either $(v \otimes v \otimes v)(W, W, W)$ or $(v \otimes B)(W, W, W)$, which can be computed as easily as $(W^T v)^{\otimes 3}$ or $(W^T v) \otimes (W^T B W)$.

$$\widetilde{T} = \frac{(\alpha_t + 1)(\alpha_t + 2)}{2} E_3(t)(W, W, W) + \alpha_t^2 (W^T E_1(t))^{\otimes 3} - \frac{\alpha_t(\alpha_t + 1)}{2}[(U_1 + U_2 + U_3)(W, W, W)] \quad (11)$$

$$E_3(t) = \frac{1}{D}\left[A_1 - A_2 - \Omega(A_2, 2, 1, 3) - \Omega(A_2, 2, 3, 1) + 2A_3\right]$$

$$A_1(W, W, W) = \sum_{d=1}^{D} s_d(t)(W^T c_d(t))^{\otimes 3}$$

$$A_2(W, W, W) = \sum_{d=1}^{D} s_d(t)(W^T c_d(t)) \otimes W^T diag(c_d(t)) W$$

$$A_3(W, W, W) = \sum_{x=1}^{V}\sum_{d=1}^{D} s_d(t) c_{d,x}(t)(W_x^T)^{\otimes 3}$$

$$U_1(W, W, W) = \frac{1}{(\alpha_t + 1)}[I + \alpha_t (W^T E_1(t))^{\otimes 2}] \otimes W^T E_1(t)$$

where $s_d(t) = \frac{1}{l_d(t)[l_d(t)-1][l_d(t)-2]}$ and W_x^T is the x-th column of W^T. $U_2(W, W, W)$ and $U_3(W, W, W)$ can be obtained by permuting $U_1(W, W, W)$'s modes. The renovated procedure needs only one pass of data in $O(LK^2)$ time.

Lines 1.7 to 1.14 perform orthogonal decomposition of $\widetilde{T}$ via a power iteration method. The orthonormal eigenpairs $(\widetilde{\lambda_z}, \widetilde{v_z})$ are found one by one. To find one such pair, the algorithm randomly starts with a unit-norm vector v, runs power iteration (Line 1.11) for n times, and records the candidate eigenpair. This process further repeats by N times, starting from various unit-norm vectors. Line 1.12 picks the eigenpair with the largest eigenvalue. After an eigenpair is found, the tensor $\widetilde{T}$ is deflated by the found component (Line 1.14), and the same power iteration is applied to it to find the next eigenpair. After all the k orthonormal eigenpairs $(\widetilde{\lambda_z}, \widetilde{v_z})$ are found, they can be used to uniquely determine the k target components $(\alpha_{t\to z}, \phi_{t\to z})$ (Line 1.13).

Line 1.15 computes the empirical topical counts for the k inferred child topics. It requires one scan of the data.

The decomposition by Algorithm 1 is fast and unique with sufficient data.

THEOREM 2. *Assume M_2 and M_3 are defined as in Equation (7), $\lambda_z > 0$, and v_z's are linearly independent with unit-norm, then Algorithm 1 finds exactly the same set of $(\lambda_z = \frac{\alpha_{t\to z}}{\alpha_t}, v_z = \phi_{t\to z})$ with high probability. The power iteration step of Line 1.11 converges in a quadratic rate.*

Algorithm 1: EXP(t, k)

```
1.1  Compute E_1(t) and E_2(t) according to Equation (6);
1.2  Find k largest orthonormal eigenpairs (σ_z, μ_z), z ∈ [k] of E_2;
1.3  M_1 = U E_1(t) ;   // U = [μ_1, ..., μ_k], Σ_1 = diag(σ_1, ..., σ_k)
1.4  Compute spectral decomposition for
       M_2' = (α_t + 1)Σ_1 − α_t M_1 ⊗ M_1 = U'ΣU'^T;
1.5  X = UU', W = M_1' Σ^{-1/2}, (W^T)^+ = XΣ^{1/2};
1.6  Compute T̃ = M_3(t)(W, W, W) according to Equation (11);
1.7  for z ∈ [k] do
1.8      λ* ← 0 ;        // the largest eigenvalue so far
1.9      for outIter ∈ [N] do
             // N, n are a small constants
1.10         v ← a random unit-norm vector;
1.11         for innerIter ∈ [n] do v ← T̃(I,v,v) / ||T̃(I,v,v)|| ;
1.12         if T̃(v,v,v) > λ* then (λ*, v*) ← (T̃(v,v,v), v);
1.13     α_{t→z} = α_t 1/(λ*)^2, φ_{t→z} = (α_{t→z}/α_t)(W^T)^+ v*;
1.14     T̃ ← T̃ − λ* v* ⊗ v* ⊗ v* ;                  // deflation
1.15 Compute c_d(t → z), z ∈ [k] according to Equation (5);
```

It satisfies single-run consistency. Multi-run consistency is guaranteed if the empirical moments are close to theoretical moments. We empirically evaluate it in Section 5.

The overall time complexity for EXP is $\mathcal{O}(LK^2 + Km + NnK^4)$, which can be regarded linear to the data size since N and n can be as small constants as 10 to 30, and K is a small number like 10 to 50 due to our assumption of human-manageable tree width. It requires only three scans of data.

4.2.2 MER *Operator*

Algorithm 2: MER(t_1, t_2)

```
2.1  Find the least common ancestor t of t_1 and t_2;
2.2  t' ← t_1;
2.3  while t' ≠ t and π_{t'} ≠ t do
2.4      t' ← π_{t'};
2.5      c(t') ← c(t') − c(t_1);
2.6      α_{t'} ← α_{t'} − α_{t_1}
2.7  t' ← t_2;
2.8  for t' ≠ t and π_{t'} ≠ t do
2.9      t' ← π_{t'};
2.10     c(t') ← c(t') − c(t_2);
2.11     α_{t'} ← α_{t'} − α_{t_2}
2.12 if t = t_1 or t = t_2 then t_3 ← t ;
2.13 else
2.14     Create topic node t_3 with parent t;
2.15     c(t_3) ← c(t_1) + c(t_2);
2.16     α_{t_3} ← α_{t_1} + α_{t_2};
2.17 for z ∈ [C_{t_1}] do π(t_1 → z) ← t_3;
2.18 for z ∈ [C_{t_2}] do π(t_2 → z) ← t_3;
2.19 Remove t_1 and t_2 from T, and add t_3 to T;
```

To merge two topics t_1 and t_2, we need to find their least common ancestor t (Line 2.1), subtract the topical counts $c(t')$ and the Dirichlet prior $\alpha_{t'}$ for any other topic t' in the path between t_1 and t_2 (Lines 2.2–2.11), and then create a new node t_3 to sum up the topical counts and Dirichlet prior of t_1 and t_2 (Lines 2.14–2.16) with one exception: when t_1 is t_2's direct ancestor or direct descendant, we can just use

t as the merged topic node (Line 2.12). We then move the children of t_1 and t_2 to be under the merged topic node (Lines 2.17–2.18). Last, we remove t_1 and t_2 and add t_3 to the topical hierarchy (Line 2.19). The complexity for MER is $\mathcal{O}(LH)$.

4.2.3 MOV *Operator*

Algorithm 3: MOV(t_1, t_2)

```
3.1  t ← π_{t_1};
3.2  while t ≠ o do
3.3      c(t) ← c(t) − c(t_1);
3.4      α_t ← α_t − α_{t_1};
3.5      t ← π_t;
3.6  t ← t_2;
3.7  while t ≠ o do
3.8      c(t) ← c(t) + c(t_1);
3.9      α_t ← α_t + α_{t_1};
3.10     t ← π_t;
3.11 Set π_{t_1} ← t_2;
```

To move the subtree rooted at t_1 to be under t_2, we first subtract topical counts and Dirichlet prior from every ancestor of t_1 (Lines 3.1–3.5), and then add them to every ancestor of t_2, including t_2 itself (Lines 3.6–3.10). Finally, we set the parent of t_1 to be t_2. The complexity for MOV is $\mathcal{O}(LH)$.

The implementation of MER and MOV using Algorithms 2 and 3 satisfy both multi-run and single-run consistency requirement.

4.3 Phrase Mining and Ranking

After the term distribution in each topic is inferred, we can then mine and rank topical phrases within each topic. The phrase mining and ranking in STROD adapt CATHY [27] to generic text. Here we briefly present the process.

In this work, a phrase is defined as a frequent consecutive sequence of terms of arbitrary lengths. To filter out incomplete phrases (*e.g.*, 'vector machine' instead of 'support vector machine') and frequently co-occurred terms that do not make up a meaningful phrase (*e.g.*, 'often use'), we use a statistical test to select quality phrases [7], and record the count $c_{d,P}$ of each phrase P in each document d.

After the phrases of mixed lengths are mined, they are ranked with regard to the representativeness of each topic in the hierarchy, based on two factors: *popularity* and *discriminativeness*. A phrase is *popular* for a topic if it appears frequently in documents containing that topic (*e.g.*, 'information retrieval' has better popularity than 'cross-language information retrieval' in the Information Retrieval topic). A phrase is *discriminative* of a topic if it is frequent only in the documents about that topic but not in those about other topics (*e.g.*, 'query processing' is more discriminative than 'query' in the database topic).

We use the topical term distributions inferred from our model to estimate the 'topical count' $c_{d,P}(t)$ of each phrase P in each document d, in a similar way as we estimate the topical count of terms in Equation (5):

$$c_{d,P}(t) = c_{d,P}(\pi_t)p(t|P,\pi_t) = c_{d,P}(\pi_t)\frac{\alpha_t \prod_{x\in P}\phi_{t,x}}{\sum_{z=1}^{C_{\pi_t}} \alpha_{\pi_t\to z}\prod_{x\in P}\phi_{\pi_t\to z,x}}$$

Let the conditional probability $p(P|t)$ be the probability of "randomly choose a document and a phrase that is about topic t, the phrase is P." It can be estimated as $p(P|t) = \frac{1}{D}\sum_{d=1}^{D}\frac{c_{d,P}(t)}{\sum_{P'} c_{d,P'}(t)}$. The popularity of a phrase P in a topic t can be quantified by $p(P|t)$. The discriminativeness can be measured by the log ratio between the probability $p(P|t)$ conditioned on topic t and the probability $p(P|\pi_t)$ conditioned on its parent topic π_t: $\log\frac{p(P|t)}{p(P|\pi_t)}$.

A good ranking function to combine these two factors is their product:

$$r_t(P) = p(P|t)\log\frac{p(P|t)}{p(P|\pi_t)} \tag{12}$$

which has an information-theoretic sense: the pointwise KL-divergence between the two probabilities [27]. Finally, we use $r_t(P)$ to rank phrases in topic t in the descending order.

5. EXPERIMENTS

In this section we first introduce the datasets and the methods used for comparison, and then describe our evaluation on efficiency, consistency, and quality.

Datasets. Our performance study is on four datasets:

- DBLP title: A set of titles of recently published papers in DBLP (`www.dblp.org`). The set has 1.9M titles, 152K unique terms, and 11M tokens.
- CS abstract: A dataset of computer science paper abstracts from Arnetminer (`www.arnetminer.org`). The set has 529K papers, 186K unique terms, and 39M tokens.
- TREC AP news: A TREC news dataset (1998). It has 106K full articles, 170K unique terms, and 19M tokens.
- Pubmed abstract: A dataset of life sciences and biomedical topic. We crawled 1.5M abstracts from Jan. 2012 to Sep. 2013 on Pubmed (`www.ncbi.nlm.nih.gov/pubmed`). The dataset has 98K unique terms and 169M tokens.

We remove English stopwords from all the documents.

Methods for comparison. We mainly evaluate EXP because it dominates the runtime, and its consistency in real-world data is subject to empirical evaluation. We compare the following topical hierarchy construction methods.

- hPAM – parametric hierarchical topic model. The hierarchical Pachinko Allocation Model [20] is a state-of-the-art parametric hierarchical topic modeling approach. hPAM outputs a specified number of supertopics and subtopics, as well as the associations between them.
- nCRP – nonparametric hierarchical topic model. We choose nCRP to represent this category for its relative efficiency. It outputs a tree with a specified height. The number of topics cannot be set exactly. We tune its hyperparameter to generate an approximately identical number of topics as other methods.
- splitLDA – recursively applying LDA, as discussed in Section 2. This heuristic method is more efficienct than the above two methods. We implement splitLDA on top of an efficient single-machine LDA inference algorithm [30].
- CATHY – recursively clustering term co-occurrence networks. CATHY [27] uses a term co-occurrence network to compress short documents and performs topic discovery through an EM algorithm.

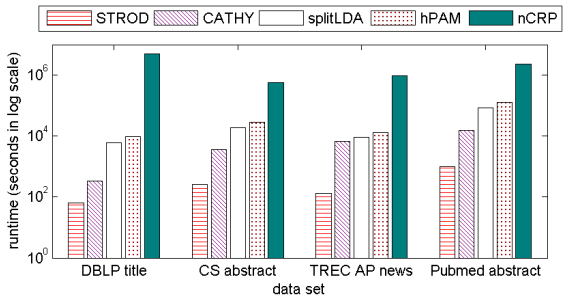
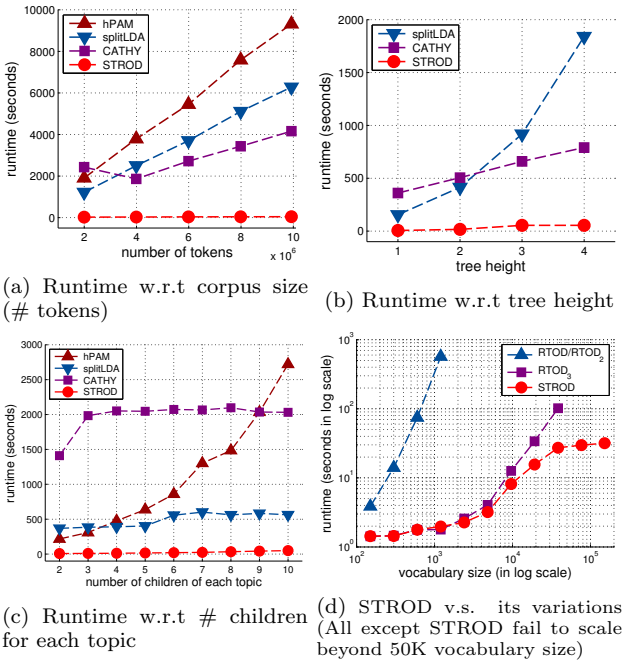


Figure 3: Total runtime on each dataset, $H = 2, C_t = 5$



(a) Runtime w.r.t corpus size (# tokens)

(b) Runtime w.r.t tree height

(c) Runtime w.r.t # children for each topic

(d) STROD v.s. its variations (All except STROD fail to scale beyond 50K vocabulary size)

Figure 4: Runtime with varying scale

- STROD and its variations RTOD, RTOD_2, RTOD_3 – recursively applying our EXP operator to expand the tree. We implement several variations to analyze our scalability improvement techniques: (i) RTOD: recursive tensor orthogonal decomposition without scalability improvement [3]; (ii) RTOD_2: RTOD plus the efficient computation of whitening matrix by avoiding creation of M_2; (iii) RTOD_3: RTOD plus the efficient computation of tensor product by avoiding creation of M_3; and (iv) STROD: Algorithm 1 with the full scale-up technique.

5.1 Efficiency

The first evaluation assesses the efficiency of different algorithms when constructing a topical hierarchy with the same depth and width.

Figure 3 shows the overall runtime in these datasets. STROD is several orders of magnitude faster than the existing methods. On the largest dataset it reduces the runtime from one or more days to 18 minutes in total. CATHY is the second best method in short documents such as titles and abstracts because it compresses the documents into term co-occurrence networks. But it is still more than 100 times slower than STROD due to many rounds of EM iterations. splitLDA and hPAM rely on Gibbs sampling, and the former is faster because it recursively performs LDA, and considers fewer dependencies in sampling. nCRP is two orders of magnitude slower.

We then conduct analytical study of the runtime growth with respect to different factors. Figures 4a–4c show the runtime varying with the number of tokens, the tree height and the tree width. We can see that the runtime of STROD grows slowly, and it has the best performance in all occasions. The margin of our method over others grows quickly when the scale increases. In Figure 4b, we exclude hPAM because it is designed for $H = 2$. We exclude nCRP from all these experiments because it takes too long time to finish (>90 hours with 600K tokens).

Figure 4d shows the performance in comparison with the variations of STROD. Both RTOD and RTOD_2 fail to finish when the vocabulary size grows beyond 1K, because the third-order moment tensor $M_3(t)$ requires $O(V^3)$ space to create. RTOD_3 also has limited scalability because the second order moment tensor $M_2(t) \in \mathbb{R}^{V \times V}$ is dense. STROD scales up easily by avoiding explicit creation of these tensors.

5.2 Consistency

The second evaluation assesses the multi-run consistency of different algorithms. For each dataset, we sample 10,000 documents and run each algorithm 10 times and measure the *variance* among the 10 runs for the same method as follows. Each pair of algorithm runs generate the same number of topics, but their correspondence is generally unknown (STROD makes an exception with its ability to obtain a unique order of subtopics according to learned α). For example, the topic $o \rightarrow 1$ in the first run may be close to $o \rightarrow 3$ in the second run. We measure the KL divergence between all pairs of topical term distributions between the two runs, build a bipartite graph using the negative KL divergence as the edge weight, and then use a maximum matching algorithm to determine the best correspondence (top-down recursively). Then we average the KL divergence between matched pairs as the difference between the two algorithm runs. Finally, we average the difference between all $10 \times 9 = 90$ ordered pairs of algorithm runs as the final variance. We exclude nCRP in this section, since even the number of topics is not a constant after each run.

Table 2 summarizes the results: STROD has lowest variance in all the three datasets. The other three methods based on Gibbs sampling have variance larger than 1 in all datasets, which implies that the topics generated across multiple algorithm runs are considerably different.

We also evaluate the variance of STROD when we vary the number of outer and inner iterations N and n. As shown in Figure 5, the variance of STROD quickly diminishes when the number of outer and inner iterations grow to 10. This validates the theoretical analysis of their fast convergence.

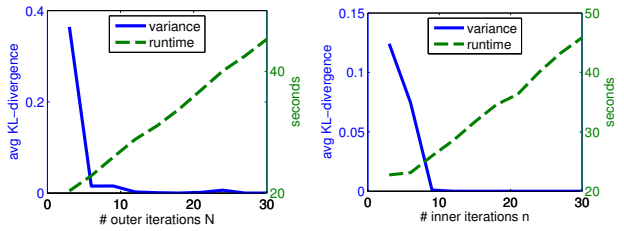
In conclusion, STROD achieves consistent performance with small runtime. It is stable and robust to be used as a hierarchy construction method for large text collections.

5.3 Interpretability

The final evaluation assesses the interpretability of the constructed topical hierarchy, via human judgment. We evaluate hierarchies constructed from DBLP titles and TREC AP news. For simplicity, we set the number of subtopics to

Table 2: The variance of multiple algorithm runs in each dataset

Method	DBLP title	CS abstract	TREC AP news
hPAM	5.578	5.715	5.890
splitLDA	3.393	1.600	1.578
CATHY	17.34	1.956	1.418
STROD	**0.6114**	**0.0001384**	**0.004522**



(a) varying N when $n = 30$ (b) varying n when $N = 30$

Figure 5: The variance and runtime of STROD when varying # outer and inner iterations N and n (CS abstract)

be 5 for all topics. For hPAM, we post-process them to obtain the 5 strongest subtopics for each topic. For all the methods we use the same phrase mining and ranking procedure to enhance the interpretability. We do not include nCRP in this study because hPAM has been shown to have superior performance of it [20].

In order to evaluate the coherence of the hierarchy, we use an *Topic Intrusion (TI)* task which were proposed in [27]: Evaluators are shown a parent topic t and X candidate child topics. $X - 1$ of the child topics are actual children of t in the generated hierarchy, and the remaining child topic is not. Each topic is represented by its top-5 ranked phrases. Evaluators are asked to select the intruder child topic, or to indicate that they are unable to make a choice.

For this study we set $X = 4$. 160 Topic Intrusion questions are randomly generated. We then calculate the agreement of the human choices with the actual hierarchical structure constructed by the various methods. We consider a higher match between a given hierarchy and human judgment to imply a higher quality hierarchy. For each method, we report the F-1 measure of the answers matched consistently by three human judgers with CS background.

Figure 6 summarizes the results. STROD is the best performing method in both datasets. This suggests that the quality of the hierarchy is not compromised by the strong efficiency and consistency of STROD. As the tree goes deeper, splitLDA degrades in quality due to inclusion of irrelevant portion of each document. Compared to splitLDA, STROD does not assign a document entirely to a topic. In addition, STROD has a theoretically guaranteed inference method for expansion, which may also account for the superior quality.

A subset of the hierarchy constructed from CS abstract by 'Expand' is presented in Figure 7. For each non-root node, we show the top ranked phrases. Node $o \rightarrow 1$ is about 'data', while its children involve database, data mining and bioinformatics. The lower the level is, the more specific the topic is, and the more multigrams emerge ahead of unigrams in general. This initial hierarchy helps users quickly see the main topics without going through all the documents. They can then use other operators to make small changes to the hierarchy to confidently and continuously refine the quality.

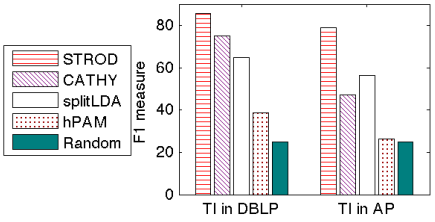


Figure 6: Topic intrustion study

6. DISCUSSIONS

In this work, we tackle the efficiency and consistency challenge of interactive topical hierarchy construction from large-scale text data. We design a novel moment-based framework to build the hierarchy recursively. Our framework divides the construction task into simpler operations in which users can be interactively involved. To support these operations, we design a new model for topical hierarchy which can be learned recursively. For consistent inference, we extend a theoretically guaranteed tensor orthogonal decomposition technique to this model. Utilizing the special structure of the tensor in our task, we scale up the algorithm significantly. By evaluating our approach on a variety of datasets, we demonstrate a prominent computational advantage. Our algorithm generates consistent and quality topic hierarchy 100-1000 times faster than the state of the art, and the margin grows when the corpus size increases.

This invention opens up numerous possibilities for future work. On the application side, it is foundation for building new systems to support explorative generation of textual data catalogs. Existing choice is either fully manual or fully automatic. The former is high quality but labor-expensive, and the latter is the opposite. By adding interaction capability to automated methods, there is hope to reduce human effort and meanwhile allow users to have quality control. On the methodology side, the advantage of STROD can be further fulfilled by parallelization and adaptation to dynamic text collections.

7. ACKNOWLEDGMENTS

Research was sponsored in part by the U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), National Science Foundation IIS-1017362, IIS-1320617, and IIS-1354329, HDTRA1-10-1-0120, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (`www.bd2k.nih.gov`), and MIAS, a DHS-IDS Center for Multimodal Information Access and Synthesis at UIUC.

8. REFERENCES

[1] A. Ahmed, Q. Ho, C. H. Teo, J. Eisenstein, E. P. Xing, and A. J. Smola. Online inference for the infinite topic-cluster model: Storylines from streaming text. In *AISTATS*, 2011.

[2] A. Ahmed, L. Hong, and A. Smola. Nested chinese restaurant franchise process: Applications to user tracking and document modeling. In *ICML*, 2013.

[3] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15:2773–2832, 2014.

[4] D. M. Blei and J. D. Lafferty. Visualizing Topics with Multi-Word Expressions. *arXiv:0907.1013*, 2009.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

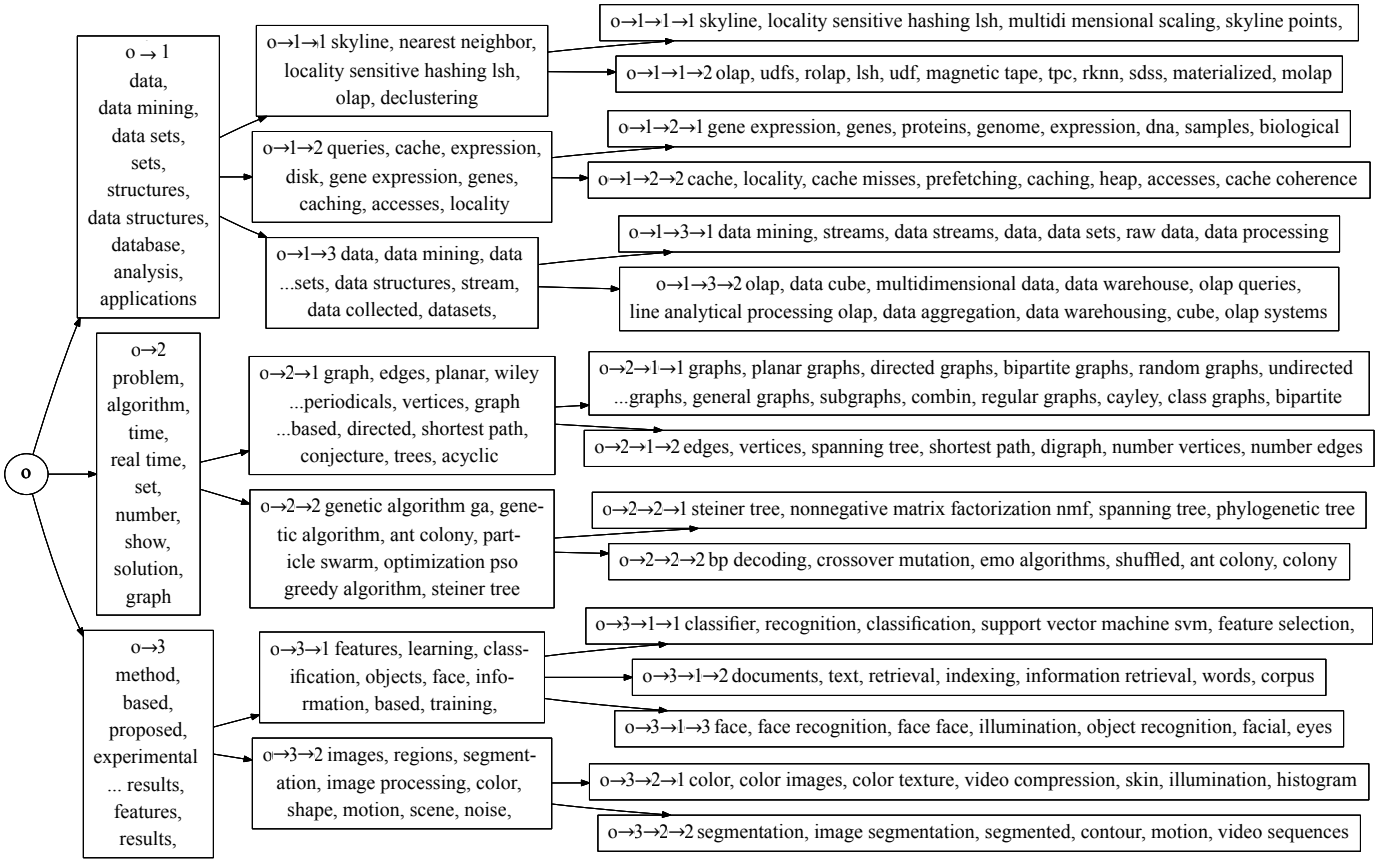


Figure 7: Sample of hierarchy generated by STROD (two phrases only differing in plural/single forms are shown only once)

[6] M. Danilevsky, C. Wang, N. Desai, J. Guo, and J. Han. Automatic construction and ranking of topical keyphrases on collections of short documents. In *SDM*, 2014.

[7] A. El-Kishky, Y. Song, C. Wang, C. R. Vossand, and J. Han. Scalable topical phrase mining from text corpora. *VLDB*, 2015.

[8] J. Foulds, L. Boyles, C. DuBois, P. Smyth, and M. Welling. Stochastic collapsed variational bayesian inference for latent dirichlet allocation. In *KDD*, 2013.

[9] B. C. Fung, K. Wang, and M. Ester. Hierarchical document clustering using frequent itemsets. In *SDM*, 2003.

[10] T. Griffiths, M. Jordan, J. Tenenbaum, and D. M. Blei. Hierarchical topic models and the nested chinese restaurant process. *NIPS*, 2004.

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

[12] M. Hoffman, D. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

[13] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1-2):177–196, Jan. 2001.

[14] J. H. Kim, D. Kim, S. Kim, and A. Oh. Modeling topic hierarchies with the recursive chinese restaurant process. In *CIKM*, 2012.

[15] D. Lawrie and W. B. Croft. Discovering and comparing topic hierarchies. In *Proc. RIAO*, 2000.

[16] A. Li, A. Ahmed, S. Ravi, and A. J. Smola. Reducing the sampling complexity of topic models. In *KDD*, 2014.

[17] W. Li and A. McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *ICML*, 2006.

[18] R. V. Lindsey, W. P. Headden, III, and M. J. Stipicevic. A phrase-discovering topic model using hierarchical pitman-yor processes. In *EMNLP-CoNLL*, 2012.

[19] X. Liu, Y. Song, S. Liu, and H. Wang. Automatic taxonomy construction from keywords. In *KDD*, 2012.

[20] D. Mimno, W. Li, and A. McCallum. Mixtures of hierarchical topics with pachinko allocation. In *ICML*, 2007.

[21] D. Newman, A. Asuncion, P. Smyth, and M. Welling. Distributed algorithms for topic models. *The Journal of Machine Learning Research*, 10:1801–1828, 2009.

[22] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *KDD*, 2008.

[23] J. Pujara and P. Skomoroch. Large-scale hierarchical topic models. In *NIPS Workshop on Big Learning*, 2012.

[24] A. Smola and S. Narayanamurthy. An architecture for parallel topic models. *Proceedings of the VLDB Endowment*, 3(1-2):703–710, 2010.

[25] D. Sontag and D. Roy. Complexity of inference in latent dirichlet allocation. In *NIPS*, pages 1008–1016, 2011.

[26] H. M. Wallach. Topic modeling: beyond bag-of-words. In *ICML*, 2006.

[27] C. Wang, M. Danilevsky, N. Desai, Y. Zhang, P. Nguyen, T. Taula, and J. Han. A phrase mining framework for recursive construction of a topical hierarchy. In *KDD*, 2013.

[28] C. Wang, X. Yu, Y. Li, C. Zhai, and J. Han. Content coverage maximization on word networks for hierarchical topic summarization. In *CIKM*, 2013.

[29] X. Wang, A. McCallum, and X. Wei. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *ICDM*, 2007.

[30] L. Yao, D. Mimno, and A. McCallum. Efficient methods for topic model inference on streaming document collections. In *KDD*, 2009.

[31] K. Zhai, J. Boyd-Graber, N. Asadi, and M. L. Alkhouja. Mr. lda: A flexible large scale topic modeling package using variational inference in mapreduce. In *WWW*, 2012.