Latent Tree Models for Hierarchical Topic Detection

Peixian Chen\textsuperscript{a}, Nevin L. Zhang\textsuperscript{a,}\textsuperscript{*}, Tengfei Liu\textsuperscript{b}, Leonard K. M. Poon\textsuperscript{c}, Zhourong Chen\textsuperscript{a}

\textsuperscript{a}Department of Computer Science and Engineering
The Hong Kong University of Science and Technology, Hong Kong
\textsuperscript{b}Ant Financial Services Group, Shanghai
\textsuperscript{c}Department of Mathematics and Information Technology
The Hong Kong Institute of Education, Hong Kong

Abstract

We propose a novel method for hierarchical topic detection where topics are obtained by clustering documents in multiple ways. Specifically, we model document collections using a class of graphical models called \textit{hierarchical latent tree models (HLTMs)}. The variables at the bottom level of an HLTM are observed binary variables that represent the presence/absence of words in a document. The variables at other levels are discrete latent variables, with those at the second level representing word co-occurrence patterns and those at higher levels representing co-occurrence of patterns at the level below. Each latent variable gives a soft partition of the documents, and document clusters in the partitions are interpreted as topics. Latent variables at high levels of the hierarchy capture long-range word co-occurrence patterns and hence give thematically more general topics, while those at low levels of the hierarchy capture short-range word co-occurrence patterns and give thematically more specific topics. Compared with LDA-based topic models, a key advantage of HLTMs is that they, as graphical models, explicitly model the dependence and independence structure among topics and words, which is conducive to the discovery of meaningful topics and topic hierarchies.

\textsuperscript{*}Corresponding author

\textit{Email address:} lzhang@cse.ust.hk (Nevin L. Zhang)
1. Introduction

The objective of hierarchical topic detection (HTD) is to, given a corpus of documents, obtain a tree of topics with more general topics at high levels of the tree and more specific topics at low levels of the tree. It has a wide range of potential applications. For example, a topic hierarchy for posts at an online forum can provide an overview of the variety of the posts and guide readers quickly to the posts of interest. A topic hierarchy for the reviews and feedbacks on a business/product can help a company gauge customer sentiments and identify areas for improvements. A topic hierarchy for recent papers published at a conference or journal can give readers a global picture of recent trends in the field. A topic hierarchy for all the articles retrieved from PubMed on an area of medical research can help researchers get an overview of past studies in the area. In applications such as those listed here, the problem is not about search because the user does not know what to search for. Rather the problem is about summarization of thematic contents and topic-guided browsing.

Topic detection has been one of the most active research areas in Machine Learning in the past decade. The most commonly used method is latent Dirichlet allocation (LDA) [1]. In LDA, a topic is a probability distribution over a vocabulary. It describes the relative frequency that an author would use each word when writing about the topic. Each document is regarded as a mixture of multiple topics and is characterized by a probability distribution over the topics. A variety of extensions of LDA have been proposed, including correlated topic models [2], dynamic topic models [3][4][5], and topic models with side information [6][7]. In particular, there are methods that explore hierarchical structures among topics [8][9][10][11]. We will refer to these extensions collectively as LDA-based methods.

Two representative LDA-based methods for HTD are nest Chinese Restaurant Process (nCRP) [10] and nest Hierarchical Dirichlet Process (nHDP) [12]. They produce trees of topics where topics at higher levels of a tree (closer to the root) appear more often than those at lower levels of the tree. To do so, they assume that there is a true topic tree behind data. A prior distribution is placed over all possible trees using nCRP, and an assumption is made as to how documents are generated from
the true topic tree, which, together with data, gives a likelihood function over all possible trees. The true topic tree is estimated by combining the prior and the likelihood in posterior inference. The two methods differ from each other in how documents are generated from the true topic tree. In the nCRP algorithm [10], topics in one document are assumed to be from one path down the tree, while in the nHDP algorithm [12] the topics in one document can be from multiple paths, i.e., a subtree within the entire topic tree.

In the LDA models, the observed variables are token variables. A token variable stands for a location in a document, and its possible values are all the words in a vocabulary. In this paper, we propose a novel model for HTD where the observed variables are word variables. Each word variable has two possible values ‘0’ and ‘1’, which represent the absence and presence of the word in a document respectively. A document is represented as a binary vector over the vocabulary. From such data, our method learns a tree model, where all the word variables are at the bottom level and there are multiple levels of binary latent variables on top of them. The model is called a hierarchical latent tree model (HLTM) and our method is hence called hierarchical latent tree analysis (HLTA). The latent variables right above the word variables represent word co-occurrence patterns and those at higher levels represent co-occurrence of patterns at the level below. In this sense, HLTA produces a hierarchy of co-occurrence patterns.

HLTA also gives a hierarchy of topics in the following sense. Each latent variable in the tree model represents a soft partition of the documents and document clusters in the partitions can be interpreted as topics. Latent variables at the higher levels of the hierarchy capture long-range word co-occurrence patterns and hence give thematically more general topics, while latent variables at low levels of the hierarchy capture short-range word co-occurrence patterns and hence give thematically more specific topics.

A key advantage of HLTA over the LDA-based methods is that, due to the use of word variables instead of token variables, it explicitly models dependence and independence relationships among words and topics using model structures. This is conducive to the discovery of meaningful topics and topic hierarchies. In our empirical studies, HLTA found substantially better topics and topic hierarchies than
Figure 1: Two latent tree models. The model $m_1$ has only one latent variable $Y$, and it is a latent class model. It gives one soft partition of data. The model $m_2$ has two latent variables $Y$ and $Z$. It gives two related partitions of data.

the LDA-based methods, despite the obvious handicap of not using word frequencies.

The rest of the paper is organized as follows. In next section we illustrate, with a toy example, the kind of models that HLTA produces and how topic hierarchies can be extracted from such models. Then we explain how HLTA builds the model structures in Section 3 and how it estimates model parameters in Section 4. In Section 5 we present the results obtained by HLTA on large real-world datasets, and in Section 6 we compare HLTA with LDA-based methods empirically. Finally, related works are discussed in Section 7 and conclusions are drawn in Section 8.

2. HLTA and Hierarchical Topic Detection

Technically, HLTA is based on a class of probabilistic graphical models called latent tree models.

2.1. Latent Tree Models

A latent tree model (LTM) is a Markov random field over an undirected tree, where the leaf nodes represent observed variables and the internal nodes represent latent variables [13][14]. Two examples are shown in Figure 1. For this subsection, all the variables are assumed to take a finite number of possible values. The number of possible values of a variable is called its cardinality.

Parameters of an LTM consist of potentials associated with edges and nodes. In this paper, the potentials for an LTM are picked as follows: Root the model at an arbitrary
latent node, direct the edges away from the root, and specify a marginal distribution for the root and a conditional distribution for each of the other nodes given its parent. Take the model in Figure 1(b) as an example. If $Y$ is chosen as the root, the parameters are the distributions $P(Y)$, $P(A \mid Y)$, $P(B \mid Y)$, $P(D \mid Y)$, $P(C \mid Z)$ and $P(E \mid Z)$. Because of the way the potentials are picked, LTMs can be viewed as tree-structured Bayesian networks [15], and the product of all the potentials defines a joint distribution over all the variables. Different ways to pick the root give different Bayesian networks that are equivalent in the sense that they represent exactly the same class of distributions over the observed variables [13].

LTMs with a single latent variable as Figure 1(a) are known as latent class models (LCMs) [16]. They are a type of finite mixture models (FMMs) [17] and are widely used for cluster analysis of categorical data [18]. The model in Figure 1(a) is an LCM and it defines the following mixture distribution over the observed variables:

$$P(A, \cdots, E) = \sum_{i=1}^{|Y|} P(Y = y_i) P(A, \cdots, E \mid Y = y_i)$$

where $|Y|$ is the cardinality of $Y$ and $y_i$ is the $i_{th}$ state of $Y$. The model stipulates that there are $|Y|$ clusters in data, the size of the $i_{th}$ cluster is $P(Y = y_i)$, and the distribution of the observed variables in the $i_{th}$ cluster is $P(A, \ldots, E \mid Y = y_i)$, which is determined by the model parameters.

The model in Figure 1(b) has two latent variables. It gives two different and related mixture distributions:

$$P(A, \cdots, E) = \sum_{i=1}^{|Y|} P(Y = y_i) P(A, \cdots, E \mid Y = y_i),$$

$$P(A, \cdots, E) = \sum_{j=1}^{|Z|} P(Z = z_j) P(A, \cdots, E \mid Z = z_j).$$

The model stipulates that there are two ways to partition data. One way is to divide data into $|Y|$ clusters based on $Y$, and another is to divide data into $|Z|$ clusters based on $Z$.

LTMs are a generalization of LCMs. An LCM has a single latent variable and it gives one soft partition of data. In this sense, it is a tool for uni-dimensional clustering. In contrast, an LTM typically has multiple latent variables and it gives multiple soft partitions of data. Hence LTMs are a tool for multidimensional clustering [13, 14, 19].
2.2. Hierarchical Latent Tree Analysis of Text Data

The input to HLTA is a collection of documents, each represented as a binary vector over the vocabulary. The output is a latent tree model where the variables are organized hierarchically, and is hence called a hierarchical latent tree model (HLTM). One example is shown in Figure 2, which is learned from a subset of the 20 Newsgroup data. The variables at the bottom level, level 0, are observed binary variables that represent the presence/absence of words in a document. The variables at other levels are discrete latent variables. For the rest of the paper, all variables are assumed to be binary.

The latent variables at level 1 were introduced during data analysis to model word co-occurrence patterns. For example, $Z_{11}$ captures the probabilistic co-occurrence of the words nasa, space, shuttle and mission; $Z_{12}$ captures the probabilistic co-occurrence of the words orbit, earth, solar and satellite; $Z_{13}$ captures the probabilistic co-occurrence of the words lunar and moon.

Latent variables at level 2 were introduced during data analysis to model the co-occurrence of the patterns at level 1. For example, $Z_{21}$ represents the probabilistic co-occurrence of the patterns $Z_{11}$, $Z_{12}$ and $Z_{13}$.

---

1http://qwone.com/jason/20Newsgroups/
2.3. Topic Hierarchy Extraction

Topics can be extracted from an HLTM. Take the model shown in Figure 2 as an example. There are 14 latent variables and each of them gives a soft partition of the data. The clusters in the partitions can be interpreted as topics. Table 1 shows the information about the partition given by \( Z_{21} \). The probability values are calculated from the model parameters, which are learned from the dataset along with the model structure.

There are two clusters in the partition \( Z_{21} = s_0 \) and \( Z_{21} = s_1 \). The cluster \( Z_{21} = s_1 \) consists of 5% of the documents. In this cluster, the words such as space, nasa and orbit occur with relatively high probabilities. It is clearly a meaningful topic and might be labeled “NASA”. The other cluster \( Z_{21} = s_0 \) consists of 95% of the documents. In this cluster, the words occur with low probabilities. We interpret it as a background topic.

Note that the document clusters given by \( Z_{21} \) are interpreted based on the word variables in the subtree rooted at \( Z_{21} \). This is because \( Z_{21} \) was introduced during data analysis to model the correlations among those variables. Also note that, in Table 1, the word variables are ordered according to their mutual information (MI) with \( Z_{21} \). The words listed at the front have the highest MI with \( Z_{21} \). They are the best ones to characterize the difference between the two clusters because their occurrence probabilities in the two clusters differ the most. If one is to choose only the top, say 5, words to characterize the topic \( Z_{21} = s_1 \), then the best words to pick are space, nasa, orbit, earth and shuttle.

If the background topics are ignored, each latent variable gives us exactly one topic. As such, the model in Figure 2 gives use 14 topics, which are shown in Table 2. In general, latent variables at high levels of the hierarchy capture long-range word co-occurrence patterns and hence give thematically more general topics, while those at low levels of the hierarchy capture short-range word co-occurrence patterns and give thematically more specific topics. In our running example, the topic given by \( Z_{22} \) (windows, card, graphics, video, dos) consists of a mixture of words about several aspects of computers, while the subtopics are each concerned with only one aspect: \( Z_{14} \) (card, video, driver), \( Z_{15} \) (dos, windows), and \( Z_{16} \) (graphics,
Table 1: Document partition given by latent variable $Z_{21}$.

|     | s0 (0.95) | s1 (0.05) |
|-----|-----------|-----------|
| space | 0.04      | 0.58      |
| nasa  | 0.03      | 0.43      |
| orbit | 0.01      | 0.33      |
| earth | 0.01      | 0.33      |
| shuttle | 0.01    | 0.24      |
| moon  | 0.02      | 0.26      |
| mission | 0.01   | 0.21      |

display, image).

3. Model Structure Construction

We present the HLTA algorithm in this and the next section. The inputs to HLTA include a collection of documents and several algorithmic parameters. The outputs include an HLTM and a topic hierarchy extracted from the HLTM. Topic hierarchy extraction has already been explained in Section 2.3, and we will hence focus on how to learn the HLTM. In this section we will describe the procedures for constructing the model structure, and in the next section we will discuss parameter estimation issues.

3.1. Top-Level Control

The top-level control of HLTA is given in Algorithm 1 and the subroutines are given in Algorithm 2-5. In this subsection, we illustrate the top-level control using the toy dataset mentioned on Section 2.2, which involves 30 word variables.

In the first pass through the loop, the subroutine BUILDISLANDS is called (line 3). It partitions all word variables into several clusters (Figure 3 bottom), such that the words in each cluster tend to co-occur and the co-occurrences can be properly modeled using a single latent variable. A latent variable is introduced for each cluster to model the co-occurrence of the words inside it. This results in a number of LCMs. We
Table 2: Topic hierarchy given by the model in Figure 2

| Topic hierarchy | Probability |
|-----------------|-------------|
| space nasa orbit earth shuttle | 0.05 |
| orbit earth solar satellite | 0.06 |
| space nasa shuttle mission | 0.05 |
| moon lunar | 0.03 |
| team games players season hockey | 0.14 |
| team season | 0.14 |
| players baseball league | 0.11 |
| games win won | 0.09 |
| hockey nhl | 0.08 |
| windows card graphics video dos | 0.24 |
| card video driver | 0.12 |
| windows dos | 0.15 |
| graphics display image | 0.10 |
| computer science | 0.09 |

The next step is to link up the islands (line 4). This is done by first estimating the mutual information (MI) between every pair of latent variables, and then finding the maximum spanning tree. The result is the model in the middle of Figure 3.

In the subroutine HARDASSIGNMENT, inference is carried out to compute the posterior distribution of each latent variable for each document. The document is assigned to the state with the maximum posterior probability, resulting in a dataset over the level-1 latent variables (line 10).

In the second pass through the loop, the level-1 latent variables are partitioned into 3 groups and one island is formed for each cluster. The islands are linked up to form the model shown at the top of Figure 3. At line 8, the model at the top of Figure 3 (referred to as $m_1$) is stacked on the model in the middle (referred to as $m$) to give rise to the hierarchical model in Figure 2. While doing so, the subroutine STACKMODELS cuts off the links among the level-1 latent variables.

One of the input parameters to HLTA is $\tau$, which is a user-specified upper bound on the number of top-level latent variables. In this example, we set $\tau = 5$. The number of
Algorithm 1 HLTA($D$, $\tau$, $\mu$, $\delta$, $\kappa$)

**Inputs:** $D$—Collection of documents, $\tau$—Upper bound on the number of top-level topics, $\mu$—Upper bound of the island size $\delta$—Threshold used in UD-test, $\kappa$—Number of EM steps on final model.

**Outputs:** An HLTM and a topic hierarchy.

1: $D_1 \leftarrow D$, $\mathcal{L} \leftarrow \emptyset$, $m \leftarrow \text{null}$.  
2: **repeat**  
3: $\mathcal{L} \leftarrow \text{BUILDISLANDS} (D_1, \delta, \mu)$;  
4: $m_1 \leftarrow \text{BRIDGEISLANDS} (\mathcal{L}, D_1)$;  
5: **if** $m = \text{null}$ **then**  
6: $m \leftarrow m_1$;  
7: **else**  
8: $m \leftarrow \text{STACKMODELS} (m_1, m)$;  
9: **end if**  
10: $D_1 \leftarrow \text{HARDASSIGNMENT} (m, D)$;  
11: **until** $|\mathcal{L}| < \tau$.  
12: Run EM on $m$ for $\kappa$ steps.  
13: **return** $m$ and topic hierarchy extracted from $m$.

Nodes at the top level in our current model is 3, which is below the threshold $\tau$. Hence the loop is exited. At line 12 the EM algorithm [22] is run on the final hierarchical model for $\kappa$ steps to improve its parameters, where $\kappa$ is another user specified input parameter. In our experiments, we set $\kappa = 50$.

3.2. Discovering Co-occurrence of Words

The first step, and the key step, of HLTA is to partition the word variables into clusters such that the words in each cluster tend to co-occur and the co-occurrences can be properly modeled using a single latent variable. This is achieved using the BUILDISLANDS subroutine, which is based on a statistical test called the *uni-dimensionality test (UD-test)* [19].
3.2.1. Uni-Dimensionality Test

Conceptually, a set of variables is said to be uni-dimensional if the correlations among them can be properly modeled using a single latent variable. Operationally, we rely on the uni-dimensionality test (UD-test) to determine whether a set $S$ of variables is uni-dimensional.

The UD-test is built upon a model selection criterion called Bayesian information criterion (BIC) \(^{23}\). The BIC score of a model is given by

$$BIC(m \mid D) = \log P(D \mid m, \theta^*) - \frac{d}{2} \log |D|, \quad (2)$$

where $\theta^*$ is the maximum likelihood estimate of the model parameters, $d$ is the number of free model parameters, and $|D|$ is the sample size. The first term in the BIC score
Algorithm 2 BuildIslands($D$, $\delta$, $\mu$)

1: $\mathcal{V} \leftarrow$ variables in $D$, $\mathcal{M} \leftarrow \emptyset$.
2: while $|\mathcal{V}| > 0$ do
3:   $m \leftarrow$ OneIsland($D$, $\mathcal{V}$, $\delta$, $\mu$);
4:   $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$;
5:   $\mathcal{V} \leftarrow$ variables in $D$ but not in any $m \in \mathcal{M}$;
6: end while
7: return $\mathcal{M}$.

measures how well the model fits the data, and the second term is a penalty term which discourages overfitting.

To perform UD-test on a set $S$ of observed variables, we first learn two latent tree models $m_1$ and $m_2$ for $S$ and then compare their BIC scores. The model $m_1$ is the model with the highest BIC score among all LTMs with a single latent variable, and the model $m_2$ is the model with the highest BIC score among all LTMs with two latent variables. Figure 1 shows what the two models might look like when $S$ consists of five observed variables $A − E$. We conclude that $S$ is uni-dimensional if the following inequality holds:

$$BIC(m_2 | D) - BIC(m_1 | D) < \delta,$$

where $\delta$ is a user-specified threshold. In other words, $S$ is considered uni-dimensional if the best two-latent variable model is not significantly better than the best one-latent variable model.

The BIC score $BIC(m | D)$ is a large sample approximation of the marginal loglikelihood $\log P(D|m)$ [23, 24]. The difference $BIC(m_2 | D) - BIC(m_1 | D)$ is hence a large approximation of the logarithm of the Bayes factor $P(D|m_2) / P(D|m_1)$ for comparing $m_1$ and $m_2$ [25]. According to the cut-off values for the Bayes factor, there is positive, strong, and very strong evidence favoring $m_2$ when the quantity is larger than 1, 3 and 5 respectively. In our experiments, we set $\delta = 3$.  

12
Algorithm 3 ONEISLAND($\mathcal{D}$, $\mathcal{V}$, $\delta$, $\mu$)

1: if $|\mathcal{V}| \leq 3$, $m \leftarrow$ LEARNLCM($\mathcal{D}$, $\mathcal{V}$), return $m$.
2: $\mathcal{S} \leftarrow$ two variables in $\mathcal{V}$ with highest MI;
3: $X \leftarrow \arg \max_{A \in \mathcal{V} \setminus \mathcal{S}} MI(A, \mathcal{S})$;
4: $\mathcal{S} \leftarrow \mathcal{S} \cup X$;
5: $\mathcal{V}_1 \leftarrow \mathcal{V} \setminus \mathcal{S}$;
6: $\mathcal{D}_1 \leftarrow$ PROJECTDATA($\mathcal{D}$, $\mathcal{S}$);
7: $m \leftarrow$ LEARNLCM($\mathcal{D}_1$, $\mathcal{S}$).
8: loop
9: $X \leftarrow \arg \max_{A \in \mathcal{V}_1} MI(A, \mathcal{S})$;
10: $W \leftarrow \arg \max_{A \in \mathcal{S}} MI(A, X)$;
11: $\mathcal{D}_1 \leftarrow$ PROJECTDATA($\mathcal{D}, \mathcal{S} \cup \{X\}$), $\mathcal{V}_1 \leftarrow \mathcal{V}_1 \setminus \{X\}$;
12: $m_1 \leftarrow$ PEM-LCM($m$, $\mathcal{S}$, $X$, $\mathcal{D}_1$);
13: if $|\mathcal{V}_1| = 0$ return $m_1$.
14: $m_2 \leftarrow$ PEM-LTM-2L($m$, $\mathcal{S} \setminus \{W\}$, $\{W, X\}$, $\mathcal{D}_1$);
15: if $BIC(m_2|\mathcal{D}_1) - BIC(m_1|\mathcal{D}_1) > \delta$ then
16: return $m_2$ with $W$, $X$ and their parent removed.
end if
18: if $|\mathcal{S}| \geq \mu$, return $m_1$.
19: $m \leftarrow m_1$, $\mathcal{S} \leftarrow \mathcal{S} \cup \{X\}$.
20: end loop

3.2.2. Building Islands

The subroutine BUILDISLANDS (Algorithm 2) builds islands one by one. It builds the first island by calling another subroutine ONEISLAND (Algorithm 3). Then it removes the variables in the island from the dataset, and repeats the process to build other islands. It continues until all variables are grouped into islands.

The subroutine ONEISLAND (Algorithm 3) requires a measurement of how closely correlated each pair of variables are. In this paper, mutual information is used for the purpose. The mutual information $I(X;Y)$ [20] between the two variables $X$ and $Y$ is defined as follows:

$$I(X;Y) = \sum_{X,Y} P(X,Y) \log \frac{P(X,Y)}{P(X)P(Y)},$$

(4)
where the summation is taken over all possible states of $X$ and $Y$. The distributions $P(X,Y)$, $P(X)$ and $P(Y)$ are estimated from data. Also we define the mutual information between a variable $X$ and a set of variables $S$ as $\text{MI}(X,S) = \max_{A \in S} \text{MI}(X,A)$.

The subroutine ONEISLAND maintains a working set $S$ of observed variables. Initially, $S$ consists of the pair of variables with the highest MI (line 2), which will be referred to as the seed variables for the island. Then the variable that has the highest MI with those two variables is added to $S$ as the third variable (line 3 and 4). Then other variables are added to $S$ one by one. At each step, we pick the variable $X$ that has the highest MI with the current set $S$ (line 9), and perform UD-test on the set $S \cup \{X\}$ (lines 12, 14, 15). If the UD-test passes, $X$ is added to $S$ (line 19) and the process continues. If the UD-test fails, one island is created and the subroutine returns (line 16). The subroutine also returns when the size of the island reaches a user-specified upper-bound $\mu$ (line 18). In our experiments, we set $\mu = 10$ or 15 according to the size of the vocabulary.

The UD-test requires two models $m_1$ and $m_2$. In principle, they should be the best models with one and two latent variables respectively. For the sake of computational efficiency, we construct them heuristically in this paper. For $m_1$, we choose the LCM where the latent variable is binary and the parameters are optimized by a fast subroutine PEM-LCM that will be described in the next section. Let $W$ be the variable in $S$ that has the highest MI with the variable $X$ to be added to the island. For $m_2$, we choose the model where one latent variable is connected to the variables in $S \setminus \{W\}$ and the second latent variable connected to $W$ and $X$. Both variables are binary and the model parameters are optimized by a fast subroutine PEM-LTM-2L that will be described in the next section.

Let us illustrate the ONEISLAND subroutine using an example in Figure 4. The pair of variables nasa and space have the highest MI among all variables, and they are hence the seed variables. The variable shuttle has the highest MI with the pair among all other variables, and hence it is chosen as the third variable to start the island (Figure 4a). Among all the other variables, mission has highest MI with the three variables in the model. To decide whether mission should be added to the group,
the two models $m_1$ and $m_2$ in Figure 4b are created. In $m_2$, shuttle is grouped with the new variable because it has the highest MI with the new variable among all the three variables in Figure 4a. It turns out that $m_1$ has higher BIC score than $m_2$. Hence the UD-test passes and the variable mission is added to the group. The next variable to be considered for addition is moon and it is added to the group because the UD-test passes again (Figure 4c). After that, the variable lunar is considered. In this case, the BIC score of $m_2$ is significantly higher than that of $m_1$ and hence the UD-test fails (Figure 4d). The subroutine ONEISLAND hence terminates. It returns
an island, which is the part of the model \( m_2 \) that does not contain the last variable lunar (Figure 4e). The island consists of the four words nasa, space, shuttle and mission. Intuitively, they are grouped together because they tend to co-occur in the dataset.

3.2.3. Bridging Islands

After the islands are created, the next step is to link them up so as to obtain a model over all the word variables. This is carried out by the BRIDGEISLANDS subroutine and the idea is borrowed from [26]. The subroutine first estimates the MI between each pair of latent variables in the islands, then constructs a complete undirected graph with the MI values as edge weights, and finally finds the maximum spanning tree of the graph. The parameters of the newly added edges are estimated using a fast method that will be described in the next section.

Let \( m \) and \( m' \) be two islands with latent variables \( Y \) and \( Y' \) respectively. The MI \( I(Y; Y') \) between \( Y \) and \( Y' \) is calculated using Equation (4) from the following joint distribution:

\[
P(Y, Y' | D, m, m') = C \sum_{d \in D} P(Y | m, d)P(Y' | m', d)
\]

where \( P(Y | m, d) \) is the posterior distribution of \( Y \) in \( m \) given data case \( d \), \( P(Y' | m', d) \) is that of \( Y' \) in \( m' \), and \( C \) is the normalization constant. In our running example, the output of BRIDGEISLANDS is shown in Figure 3 (middle). It is said to be a flat LTM because each latent variable is connected to at least one observed variable. In contrast, in a hierarchical LTM, only level-1 latent variables are connected to observed variables, and latent variables at higher levels are not.

3.3. Discovering Co-occurrence of Patterns

After discovering and modeling the word co-occurrence patterns (represented by level-1 latent variables), the next step is to partition the patterns into clusters such that the patterns in each cluster tend to co-occur and the co-occurrences can be properly modeled using a single latent variable.
To do so, HLTA first calls the subroutine HARDASSIGNMENT to convert the level-1 latent variables into observed variables, and to project the dataset onto those variables to obtain a new dataset $D_1$ (line 10). It then runs the subroutines BUILDISLANDS and BRIDGEISLANDS on the new dataset.

In our running example, the result of those operations is the flat LTM shown at the top of Figure 3. The level-1 latent variables are partitioned into three clusters. The clusters reveal co-occurrence of patterns. For example, the patterns $Z_{14}$, $Z_{15}$, $Z_{16}$ and $Z_{17}$ are about different aspects of computers. It is natural to expect that they tend to co-occur in documents.

In the first pass through the loop, HLTA discovers word co-occurrence patterns. In the second pass through the loop, HLTA discovers co-occurrence of patterns. The subroutine STACKMODELS combines those findings to obtain a two-level hierarchy of co-occurrence relationships. As mentioned in Section 3.1 the process can continue to obtain a hierarchy of many levels.

Technically, STACKMODELS takes two flat LTMs $m_1$ and $m$ as inputs, where the observed variables in $m_1$ are the latent variables in $m$. It creates a new model by stacking $m_1$ on top of $m$ and cuts off the links among the latent variables in $m$. The parameter values for the new model are copied from $m_1$ and $m$.

4. Parameter Estimation

HLTA conceptually consists of a model construction phase (lines 2-11) and a parameter estimation phase (line 12). During the first phase, a large number of intermediate models are generated. Whether HLTA can scale up depends on whether the parameters of the intermediate models and the final model can be estimated efficiently.

In this section, we present a fast method called progressive EM for estimating the parameters of the intermediate models. It is the result of marrying two different parameter estimation methods: the EM algorithm and the method of moments. We also discuss how to estimate the parameter of the final model efficiently when the sample size is very large.
We start by briefly reviewing the EM algorithm and the method of moments in the context of latent tree models.

4.1. The EM Algorithm

Let $X$ and $H$ be respectively the sets of observed and latent variables in an LTM $m$, and let $V = \mathbf{X} \cup \mathbf{H}$. Assume one latent variable is picked as the root and all edges are directed away from the root. For any $V$ in $V$ that is not the root, the parent $\text{pa}(V)$ of $V$ is a latent variable and can take values ‘0’ or ‘1’. For technical convenience, let $\text{pa}(V)$ be a dummy variable with only one possible value when $V$ is the root. Enumerate all the variables as $V_1, V_2, \cdots, V_n$. We denote the parameters of $m$ as

$$\theta_{ijk} = P(V_i = k | \text{pa}(V_i) = j),$$

where $i \in \{1, \cdots, n\}$, $k$ is value of $V_i$ and $j$ is a value of $\text{pa}(V_i)$. Let $\theta$ be the vector of all the parameters.

Given a dataset $D$, the loglikelihood function of $\theta$ is given by

$$l(\theta | D) = \sum_{d \in D} \sum_{H} \log P(d, H | \theta).$$

The maximum likelihood estimate (MLE) of $\theta$ is the value that maximizes the loglikelihood function.

Due to the presence of latent variables, it is intractable to directly maximize the loglikelihood function. An iterative method called the Expectation-Maximization (EM) algorithm is usually used in practice. EM starts with an initial guess $\theta^{(0)}$ of the parameter values, and then produces a sequence of estimates $\theta^{(1)}, \theta^{(2)}, \cdots$. Given the current estimate $\theta^{(t)}$, the next estimate $\theta^{(t+1)}$ is obtained through an E-step and an M-step. In the context of latent tree models, the two steps are as follows:

- **The E-step:**

$$n_{ijk}^{(t)} = \sum_{d \in D} P(V_i = k, \text{pa}(V_i) = j | d, m, \theta^{(t)})$$

- **The M-step:**

$$\theta_{ijk}^{(t+1)} = \frac{n_{ijk}^{(t)}}{\sum_k n_{ijk}^{(t)}}$$
Note that the E-step requires the calculation of \( P(V_i, pa(V_i)|d, m, \theta^{(t)}) \) for each data case \( d \in \mathcal{D} \) and each variable \( V_i \). For a given data case \( d \), we can calculate \( P(V_i, pa(V_i)|d, m, \theta^{(t)}) \) for all variables \( V_i \) in linear time using message propagation [27].

EM terminates when the improvements in loglikelihood \( l(\theta^{(t+1)}|\mathcal{D}) - l(\theta^{(t)}|\mathcal{D}) \) falls below a predetermined threshold or when the number of iterations reaches a predetermined limit. To avoid local maxima, multiple restarts are usually used.

4.2. Method of Moments

The basic idea of method of moments (MoM) [28] is to relate model parameters to population moments and to estimate the parameters by solving equations. In 1996, Chang derived equations for phylogenetic trees that relate model parameters with order 2 and 3 moments, i.e., joint distributions of two and three observed variables [29]. Since then, similar equations have been derived for other models [30]. In particular, Anandkumar et al. showed that equations in [29] are true for latent tree models, of which phylogenetic trees are a subclass [31]. Zhang et al. gave an alternative proof of the result in terms of variable elimination in a class of generated Markov random fields that have negative potentials [32].

Consider the LCM \( m_1 \) in Figure 1 where all the variables have the same number of states. The conditional distribution \( P(A|Y) \) can be regarded as a square matrix and can be denoted as \( P_{A|Y} \). Similarly, let \( P_{AC} \) be the matrix representation of the joint distribution \( P(A, C) \). For a fixed value \( b \) of \( B \), let \( P_{b|Y} \) be the vector presentation of \( P(B=b|Y) \) and let \( P_{AbC} \) be the matrix representation of \( P(A, B=b, C) \). The following theorem is borrowed from [32].

**Theorem 1.** Let \( Y \) be the latent variable in an LCM and \( A, B, C \) be three of the observed variables. Assume all variables have the same cardinality and the matrices \( P_{A|Y} \) and \( P_{AC} \) are invertible. Then we have

\[
P_{A|Y} \text{diag}(P_{b|Y}) P_{A|Y}^{-1} = P_{AbC} P_{AC}^{-1},
\]

where \( \text{diag}(P_{b|Y}) \) is a diagonal matrix with components of \( P_{b|Y} \) as the diagonal elements.
Figure 5: Progressive parameter estimation using MoM: $P(A|Y)$, $P(B|Y)$, $P(D|Y)$ and $P(Y)$ are first estimated in the part of the model shaded in (a); $P(C|Z)$ and $P(E|Z)$ are then estimated in the part shaded in (b); finally $P(E|Y)$ is estimated from the part shaded in (c) and $P(Z|Y)$ is computed using $P(Z|Y)$ and $P(E|Z)$.

Theorem 1 can be used to estimate the conditional distribution $P(B|Y)$ as follows: First, obtain the empirical distributions $\hat{P}(A, B, C)$ and $\hat{P}(A, C)$ from data; Then, for each value $b$ of $B$, form the matrix on the right hand side of Equation (8) and find its eigenvalues; Finally, use those eigenvalues as the estimates of $P(B = b|Y)$ and obtain an estimate of $P(B|Y)$ by properly aligning the labels of the states of $Y$ for different $b$’s. This method is sometimes called the spectral method because it involves finding eigenvalues of matrices.

The technique can be used to estimate all the parameters in a model in multiple steps such that each step involves only a small part of the model. We illustrate this using the model $m_2$ from Figure 1 with $Y$ picked as the root. The process is called progressive parameter estimation and is shown in Figure 5. We start by focusing on the part of the model shaded in Figure 5a. Here we can estimate $P(B|Y)$ as explained in the previous paragraph. By swapping the roles of the variables, we can also estimate $P(A|Y)$ and $P(D|Y)$. Note that the labels of the states of $Y$ need to be properly aligned in different distributions since matrices obtained may be permuted when Equation (8) is carried out. The marginal distribution $P(Y)$ can be obtained from $P(A|Y)$, $P(D|Y)$ and $\hat{P}(A, D)$ as follows:

$$\text{diag}(P_Y) = P^{-1}_{A|Y}P_{AD}(P^{-1}_{D|Y})^T.$$  

Next we focus on the part of the model shaded in Figure 5b and consider it to be an LCM with latent variable $Z$ and observed variables $A$, $C$, and $E$. Here we can
estimate $P(C|Z)$ and $P(E|Z)$. It is also possible to estimate $P(A|Z)$, although it is unnecessary. Note that it is not possible to estimate $P(Z|Y)$.

To estimate $P(Z|Y)$, we need to consider the part of model shaded in Figure 5b and the part shaded in Figure 5c. We can estimate $P(E|Z)$ in 5b and $P(E|Y)$ in 5c. An estimate of $P(Z|Y)$ can then be obtained using the following formula:

$$P_{Z|Y} = P_{E|Z}^{-1}P_{E|Y}$$

4.3. Progressive EM

Both EM and MoM have their pros and cons. Being an iterative algorithm, EM can be trapped in local maxima. It is also time-consuming and does not scale up well. MoM is not iterative and hence local maximum is not an issue. It is very fast for two reasons. First, it allows the parameters to be estimated in a progressive fashion where we need to consider only a small part of the model at each time. Second, the parameters are determined by finding eigenvalues of matrices with fixed dimensions (in this paper $2 \times 2$ matrices), which takes constant time.

On the other hand, MoM relies on relationships between model parameters and empirical marginals that are approximately true under two conditions: (1) the model $m$ under consideration matches the structure of the “true” model behind data and (2) the sample size is sufficiently large. MoM is not robust because the first condition is almost always violated in practice. It often produces poor estimates and even insensible estimate (negative values for probability). EM does not require the assumptions. By maximizing the likelihood function, it tries to find the parameter values so as to make the model $m$ match the “true” distribution behind data as closely as possible. It is more robust than MoM and never produces negative estimates for probability parameters.

Progressive EM combines the desirable features of MoM and EM. It estimates all the parameters in multiple steps and, in each step, it considers a small part of the model and runs EM in the submodel to maximize the local likelihood function. The idea is illustrated in Figure 6 using the model $m_2$ from Figure 1. We first run EM in the part of the model shaded in Figure 6a to estimate $P(Y)$, $P(A|Y)$, $P(B|Y)$ and $P(D|Y)$,
Figure 6: Progressive EM: EM is first run in the submodel shaded in (a) to estimate the distributions $P(Y), P(A|Y), P(B|Y)$ and $P(D|Y)$, and then, EM is run in the submodel shaded in (b), with $P(Y), P(B|Y)$ and $P(D|Y)$ fixed, to estimate the distributions $P(Z|Y), P(C|Z)$ and $P(E|Z)$.

Algorithm 4 PEM-LCM($m, S, X, D$)

1: $Y \leftarrow$ the latent variable of $m$;
2: $S_1 \leftarrow \{X\} \cup$ two seed variables in $S$;
3: While keeping the other parameters fixed, run EM in the part of $m$ that involves $S_1 \cup Y$ to only estimate $P(X|Y)$.
4: return $m$

and then run EM in the part of the model shaded in Figure 6b, with $P(Y), P(B|Y)$ and $P(D|Y)$ fixed, to estimate $P(Z|Y), P(C|Z)$ and $P(E|Z)$.

4.4. Progressive EM and HLTA

We use progressive EM to estimate the parameters for the intermediate models generated by HLTA, specifically those generated by subroutine ONEISLAND (Algorithm 3). It is carried out by the two subroutines PEM-LCM and PEM-LTM-2L.

At lines 1 and 7 ONEISLAND needs to estimate the parameters of an LCM with three observed variables. It is done using EM. Next, it enters a loop. At the beginning, we have an LCM $m$ for a set $S$ of variables. The parameters of the LCM have been estimated earlier (line 7 at beginning or line 12 of previous pass through the loop). At lines 9 and 10 ONEISLAND finds the variable $X$ outside $S$ that has maximum MI with $S$, and the variable $W$ inside $S$ that has maximum MI with $X$.

At line 12 ONEISLAND adds $X$ to the $m$ to create a new LCM $m_1$. The parameters of $m_1$ are estimated using the subroutine PEM-LCM (Algorithm 4), which
Algorithm 5 PEM-LTM-2L($m, S \setminus \{W\}, \{W, X\}, D$)

1: $Y \leftarrow$ the latent variable of $m$
2: $m_2 \leftarrow$ model obtained from $m$ by adding $X$ and a new latent variable $Z$, connecting $Z$ to $Y$, connecting $X$ to $Z$, and re-connecting $W$ (connected to $Y$ before) to $Z$;
3: $S_1 \leftarrow \{W, X\} \cup$ two seed variables in $S$;
4: While keeping the other parameters fixed, run EM in the part of $m_2$ that involves $S_1 \cup Y \cup Z$ to only estimate $P(W|Z)$, $P(X|Z)$ and $P(Z|Y)$.
5: return $m_2$

is an application of progressive EM. Let us explain PEM-LCM using the intermediate models shown in Figure 4. Let $m$ be the model shown on the left of Figure 4c and $S = \{\text{nasa, space, shuttle, mission, moon}\}$. The variable $X$ to be added to $m$ is lunar, and the model $m_1$ after adding lunar to $m$ is shown on the left of Figure 4d. The only distribution to be estimated is $P(\text{lunar}|Y)$, as other distributions have already been estimated. PEM-LCM estimates the distribution by running EM on a part of the model $m_1$ in Figure 4 (left), where the variables involved are in rectangles. The variables nasa and space are included in the submodel, instead of other observed variables, because they were the seed variables at line 2 of Algorithm 3.

At line 19 ONEISLAND adds $X$ to the $m$ to create a new LTM $m_2$ with two latent variables. The parameters of $m_2$ are estimated using the subroutine PEM-LTM-2L (Algorithm 5), which is also an application of progressive EM. In our running example, let moon be the variables $W$ that has the highest MI with lunar among all variables in $S$. Then the model $m_3$ is as shown on the right hand side of Figure 4d. The distributions to be estimated are: $P(Z|Y)$, $P(\text{moon}|Z)$ and $P(\text{lunar}|Z)$. PEM-LTM-2L estimates the distributions by running EM on a part of the model $m_2$ in Figure 7 (right), where the variables involved are in rectangles. The variables nasa and space are included in the submodel, instead of shuttle and mission, because they were the seed variables at line 2 of Algorithm 3.

---

1When one of the seed variables is $W$, use the other seed variable and the variable picked at line 2 of Algorithm 3.
Figure 7: Parameter estimation during island building: To determine whether the variable \textit{lunar} should be added to the island $m_1$ in Figure 4c, two models are created. We need to estimate only $P(\text{lunar}|Y)$ for the model on the left, and $P(Z|Y)$, $P(\text{moon}|Z)$ and $P(\text{lunar}|Z)$ for the model on the right. The estimation is done by running EM in the parts of the models where the variable names are in rectangles.

There is also a parameter estimation problem inside the subroutine \textsc{BridgedIslands}. After linking up the islands, the parameters for edges between latent variables must be estimated. We use progressive EM for this task also. Consider the model in the middle of Figure 2. To estimate $P(Z_{11}|Z_{12})$, we form a sub-model by picking two children of $Z_{11}$, for instance \textit{nasa} and \textit{space}, and two children of $Z_{12}$, for instance \textit{orbit} and \textit{earth}. Then we estimate the distribution $P(Z_{11}|Z_{12})$ by running EM in the submodel with all other parameters fixed.

4.5. Stochastic EM

The parameters of the final model are optimized by running EM for $\kappa$ steps starting from the estimates obtained by progressive EM in the model construction phase. To scale up to very large datasets, we run each iteration of EM on a randomly sampled subset of data cases instead of the entire dataset. The idea is borrowed from stochastic gradient descent [33] and the method is hence called stochastic EM.

In implementation, we assume that $\kappa$ minibatches have been randomly sampled from the dataset to be analyzed. A minibatch can be the entire dataset if the sample size is not very large. We learn the hierarchical model structure from the first
minibatch, and optimize the parameters of the final model by running stochastic EM on all minibatches, with each EM iteration on a different minibatch.

4.6. Complexity Analysis

Let \( n \) be the number of observed variables and \( B \) be the batch size. HLTA requires the computation of empirical MI for each pair of observed variable from the first minibatch. This takes \( O(n^2 B) \) time.

When building islands for the observed variables, HLTA generates roughly \( 2n \) intermediate models. Progressive EM is used to estimate the parameters of the intermediate models. It is run on submodels with 3 or 4 observed variables. The projection of a dataset onto 3 or 4 binary variable consists of only 8 or 16 distinct cases no matter how large the original sample size is. Hence progressive EM takes constant time, which we denote using \( c \). This is the key reason why HLTA can scale up. The projection takes \( O(B) \) time for each submodel. Hence the total time for island building is \( O(2n^2 B + c) \).

To bridge the islands, HLTA needs to estimate the MI between every pair of latent variables and runs progressive EM to estimate the parameters for the edges between the islands. A loose upper bound for the running time of this step is \( n^2 B + n(B + c) \). The total number of variables (observed and latent) in the resulting flat model is upper bounded by \( 2n \). Inference on the model takes \( 2n \) time for each data case. Hence the hard assignment step takes \( O(4nB) \). So, the total time for the first pass through the loop in HLTA is \( O(2n^2 B) + O(3n(B + c)) + O(4nB) = O(2n^2 B + O(n(3c + 7B)) \) if we ignore the last term.

As we move up one level, the number of “observed” variables is decreased by at least half. Hence, the total time for the model construction phase is upper bounded by \( O(4n^2 B) + O(2n(3c + 7B)) \).

The total number of variables (observed and latent) in the final model is upper bounded by \( 2n \). Hence, one EM iteration takes \( O(4nB) \) time and the final parameter optimization steps takes \( O(4nB\kappa) \) times.

The total running time of HLTA is \( O(4n^2 B) + O(6nBc) + O(4nB\kappa) \), where we ignore the term \( 14nB \) because it is dominated by the last term. Roughly speaking, the
The nodes $Z_{601}$ to $Z_{605}$ are top-level latent variables. The subtrees rooted at different top-level latent variables are coded using different colors. Edge width represents mutual information.

three terms are respectively the times for computing MI between variables, for creating and bridging islands, and for optimizing the parameters of the final model.

5. Empirical Results

In this section, we empirically demonstrate that HLTA is capable of discovering meaningful topic hierarchies and is able to handle large datasets. In the next section, we will compare HLTA with the LDA-based methods. HLTA is implemented in Java. The code can be found at http://www.cse.ust.hk/faculty/lzhang/topic/index.htm, along with the datasets used in this paper and the full details of the results obtained on them.
Table 3: Part of topic hierarchy obtained by HLTA on the NIPS dataset.

1. \(0.22\) mixture gaussian mixtures em covariance 
   1.1. \(0.23\) em maximization ghahramani expectation 
   1.2. \(0.23\) mixture gaussian mixtures covariance 
   
   1.3. \(0.23\) generative dis generative generative 
   1.4. \(0.27\) variance noise exp variances deviation 
      1.4.1. \(0.28\) variance exp variances deviation cr 
      1.4.2. \(0.44\) noise noisy robust robustness mea 

2. \(0.38\) trained hidden layer hinton train 
   2.1. \(0.38\) trained hidden train task learn 
   2.2. \(0.42\) layer units net backpropagation layers 
   2.3. \(0.26\) morgan kaufmann advances san touretzky 
   
   2.4. \(0.29\) rumelhart connectionist mcclelland 
   2.5. \(0.29\) hinton propagation williams back boltzmann 

3. \(0.26\) images image pixel pixels object 
   3.1. \(0.25\) images image features detection face 
   3.2. \(0.24\) camera video imaging false tracked 
   3.3. \(0.24\) pixel pixels intensity intensities 
   3.4. \(0.17\) object objects shape views plane 
   
   3.5. \(0.20\) rotation invariant translation invariance 
   3.6. \(0.26\) nearest neighbor kohonen neighbors 

4. \(0.15\) speech word speaker language phoneme 
   4.1. \(0.16\) word language vocabulary words sequence 
   4.2. \(0.11\) spoken acoustics utterances speakers 
   4.3. \(0.10\) string strings grammar symbol symbols 
   4.4. \(0.06\) retrieval search semantic searching 
   
   4.5. \(0.14\) phoneme phonetic phonemes waibel lang 
   4.6. \(0.15\) speech speaker acoustic hmm hmrms 

5. \(0.12\) reinforcement sutton barto actions policy 
   5.1. \(0.14\) actions action environment environments 
   5.2. \(0.11\) sutton barto massachusetts moore td 
   5.3. \(0.10\) reinforcement policy reward rewards singh 
   
   5.4. \(0.17\) control controller dynamic trajectory 
   5.5. \(0.12\) robot arm robotics robots sensor 
   
   5.6. \(0.38\) convergence update converge iteration 
   5.7. \(0.37\) minimizing optimization minimize
5.1. Results on the NIPS Dataset

We first report the results on a small dataset, namely the NIPS dataset. It consists of 1955 articles published at the NIPS conference from 1988 to 1999\footnote{http://www.cs.nyu.edu/~roweis/data.html}. A vocabulary of 10,000 words was selected using average TF-IDF for the analysis. The entire dataset was used in the model construction phase and at every EM iteration in the parameter estimation phase. The limit on the number of top-level latent variables was set at $\tau = 20$, and EM was run for $\kappa = 50$ steps on the final model.

The analysis took around 320 minutes on a desktop machine. The result is an HLTM with 6 levels of latent variables and 13 latent variables at the top level. Figure\textsuperscript{8} shows a part of the model structure. Five top-level latent variables are included in the figure. The level-5 and level-3 latent variables in the subtrees rooted at the five top-level latent variables are also included. Each level-3 latent variable is connected to four word variables in its subtrees. Those are the word variables that have the highest MI with the latent variable among all word variables in the subtree.

The structure is interesting. Most words in the $Z_{601}$ subtree are concerned with \textit{mixture models}; most words in the $Z_{602}$ subtree are concerned with \textit{artificial neural networks}; most words in the $Z_{603}$ subtree are concerned with \textit{image analysis}; most words in the $Z_{604}$ subtree are concerned with \textit{speech recognition}. $Z_{605}$ has three branches, which are respectively about \textit{reinforcement learning}, \textit{robot control}, and \textit{optimization}. Note that several words were chopped into two parts due to hyphenation in the original pdf files. Examples include “structure”, “distribution” and “guassian” under $Z_{601}$. It is interesting that the two parts are grouped under the same latent variables.

Table\textsuperscript{8} shows the topic hierarchy extracted from the part of the model shown in Figure\textsuperscript{8}. There are five top-level topics. Those topics have low branching factors and hence their grandchildren are displayed instead of their children. The grandchildren topics are grouped according to their respective parents.

The topics and the relationships among them are meaningful. For example, the topic 1 consists of terms related to mixture models. The subtopic 1.1 is about the EM
algorithm; the subtopic 1.2 is about Gaussian mixtures; and the subtopic 1.3 is about generative distributions. The subtopic 1.4 is a combination of variance and noise, which is split into two subtopics at the next lower level. The topic 3 is about image analysis. The first four subtopics are about different aspects of image analysis: feature extraction (3.1); sources of images (3.2); pixels in images (3.3); and objects in images (3.4). The subtopics 3.5 and 3.6 are also related but do not fit in well with those four subtopics. They are placed in another subgroup by HLTA. The topic 4 is about speech recognition. Its subtopics are about different aspects of speech recognition. Only the subtopic 4.4 does not seem to fit in the group well.

5.2. Results on the NYT dataset

Next we report the results on a large dataset, namely the NYT dataset. It consists of 300,000 articles published on New York Times from 1987 to 2007. A vocabulary
of 10,000 words was selected using average TF-IDF for the analysis. The dataset was randomly divided into 50 equal-sized minibatches, and each minibatch consists of 6,000 articles. The first minibatch was used in the model construction phase. During the parameter estimation phase, EM was run on each minibatch once to improve the parameters of the final model. The limit on the number of top-level latent variables was set at $\tau = 30$.

The analysis took 670 minutes on a desktop machine. The result is an HLTM with 5 levels of latent variables and 21 latent variables at the top level. Figure 9 shows a part of the model structure. We see that most words in the subtree rooted at $Z_{501}$ are about *economy and stock market*; most words in the subtree $Z_{502}$ are about *companies and various industries*; most words in the subtree rooted at $Z_{503}$ are about *movies and music*; and most words in the subtree rooted at $Z_{504}$ are about *cooking*.

Table 4: Part of topic hierarchy obtained by HLTA on the NYT dataset

| Level | Latent Variables |
|-------|------------------|
| 1.    | [0.20] economy stock economic market dollar |
| 1.1.  | [0.20] economy economic economist rising recession |
| 1.2.  | [0.20] currency minimum expand expansion wage |
| 1.2.1. | [0.20] currency expand expansion expanding euro |
| 1.2.2. | [0.23] labor union demand industries dependent |
| 1.2.3. | [0.21] minimum wage employment retirement |
| 1.3.  | [0.20] stock market investor analyst investment |
| 1.4.  | [0.20] price prices merger shareholder acquisition |
| 1.5.  | [0.20] dollar billion level maintain million lower |
| 1.6.  | [0.20] profit revenue growth troubles share revenues |
| 1.7.  | [0.20] percent average shortage soaring reduce |
| 2.    | [0.22] companies company firm industry incentive |
| 2.1.  | [0.23] firm consulting distribution partner |
| 2.2.  | [0.23] companies company industry |
| 2.3.  | [0.14] insurance coverage insurer pay premium |
| 2.4.  | [0.25] store stores consumer product retailer |
| 2.5.  | [0.21] proposal proposed health welfare |
| 2.6.  | [0.20] drug pharmaceutical prescription |
| 2.7.  | [0.07] federal reserve fed nasdaq composite |
| 2.8.  | [0.09] enron internal accounting collapsed |
| 2.9.  | [0.09] gas drilling exploration oil natural |
Table 4 shows a part of the topic hierarchy extracted from the part of the model shown in Figure 9. The topics and the relationships among them are meaningful. For example, the topic 1 is about economy and stock market. It splits into two groups of subtopics, one on economy and another on stock market. Each subtopic further splits into sub-subtopics. For example, the subtopic 1.2 under economy splits into three subtopics: currency expansion, labor union and minimum wages. The topic 2 is about company-firm-industry. Its subtopics include several types of companies such as insurance, retail stores/consumer products, natural gas/oil, drug, and so on.

5.3. Broadly vs Narrowly Defined Topics

In HLTA, each latent variable is introduced to model a pattern of probabilistic word co-occurrence. It also gives us a topic, which is a soft cluster of documents. The size of the topic is determined by considering not only the words in the pattern, but all the words in the vocabulary. As such, it conceptually includes two types of documents: (1) documents that contain, in a probabilistic sense, the pattern, and (2) documents that do not contain the pattern but are otherwise similar to those that do. Because of the inclusion of the second type of documents, the topic is said to be broadly defined. All the topics reported above are broadly defined.

The size of a widely defined topic might appear unrealistically large at the first glance. For example, one topic detected from the NYT dataset consists of the words affair, widely, scandal, viewed, intern, monica, lewinsky, and its size is 0.18. Although this seems too large, it is actually reasonable. It is a fact that the fraction of documents that contain the seven words in the topic should obviously be much smaller than 18%. However, those documents contain many other words, such as bill and clinton, about American politics. Other documents that contain many of those other words are also included in the topic, and hence it is not surprising for the topic to cover 18% of the documents. As a matter of fact, there are several other topics about American politics are of the same size. One of them is: corruption campaign political democratic presidency.

In some applications, it might be desirable to identify narrowly defined topics — topics made up of only the documents containing particular patterns. Such topics can
be obtained by viewing the subtree rooted at each latent variable as a separate model and optimizing its parameters by running EM on the subtree. The size of a narrowly defined topic is typically much smaller than that of the widely defined version. For example, the sizes of the narrowly defined version of the two topics from the previous paragraph are 0.013 and 0.120 respectively.

6. Comparisons with the LDA-Based Methods

HLTA differs fundamentally from the LDA-based methods. In this section, we first explain the differences and then compare the performances of the two approaches empirically.

6.1. Conceptual Differences

There are four key conceptual differences between HLTA and the LDA approach. The first difference is that, as mentioned in Section 1, HLTA is based on word variables while the LDA approach is based on token variables. The use of word variables allows HLTA to learn structures such as those shown in Figure 8 and Figure 9 and those structures give rise to meaningful topics and topic hierarchies. This is the most important advantage that HLTA has over the LDA approach. In the meantime, HLTA also has two drawbacks that need to be addressed in future research due to the use of word variables. It does not consider word counts and does not allow a word to appear in multiple branches of a hierarchy.

The second difference concerns the definition of topic. In the LDA approach, a topic is a distribution over a vocabulary. It is characterized using a few words with the highest probabilities under the topic. However, high probability words from one topic may also occur with high probabilities in other topics, and hence might not be the best words to characterize the topic. In HLTA, a topic is a soft cluster of document. It is characterized using words that not only occur with high probabilities in the cluster but also occur low probabilities outside the cluster.

The third difference lies in the relationship between topics and documents. In the LDA approach, a document is a mixture of topics, and the probabilities of the topics
within a document sum to 1. Because of this, the LDA models are sometimes called *mixed-membership models*. In HLTA, a document is a member of a topic, and a document might belong to multiple topics with probability 1. In this sense, HLTMs can be said to be *multi-membership models*.

The fourth difference is about the semantics of the resultant hierarchies. In the context of document analysis, a common concept of hierarchy is a tree where each node represents a cluster of documents, and the cluster of documents at a node is the union of the document clusters at its children. Neither HLTA nor the LDA approach produces such hierarchies. The output of an LDA-based method for HTD is a tree where each node represents an LDA topic, i.e., a distribution over the vocabulary. The topics at higher levels appear more often than those at lower levels, but they are not necessarily related thematically. In contrast, the output of HLTA is a tree where each node represents a latent variable. Latent variables at higher levels of the hierarchy capture long-range word co-occurrence patterns and hence give thematically more general topics, while latent variables at lower levels of the hierarchy capture short-range word co-occurrence patterns and hence give thematically more specific topics.

In addition, the LDA approach is limited in the number of hierarchy levels it can handle, which is typically set at three in practice. In HLTA, the number of levels is not limited to three and the number of latent variables on each level is determined fully automatically. The model produced by HLTA has no more than $4n$ parameters, where $n$ is the size of the vocabulary. The model produced by the LDA approach has approximately $k(n + N)$ parameters, where $k$ is the number of topics and $N$ is the number of documents. This is much larger than $4n$.

6.2. *Empirical Comparisons*

We now present empirical results to compare HLTA with nCRP [10] and nHDP [12]. Also included in the comparisons is CorEx [34], a recently proposed algorithm that constructs hierarchical latent trees by optimizing an information theoretical objective function.
6.2.1. Datasets

In addition to the NIPS and NYT datasets, the 20 Newsgroup (referred to as Newsgroup) dataset was also used in our experiments. It consists of 19,940 newsgroup documents after preprocessing. Three versions of the NIPS dataset and two versions of the Newsgroup dataset were created by choosing vocabularies with different sizes using average TF-IDF. So, the experiments were performed on six datasets. Information about the datasets is summarized in Table 5. The data are represented as binary vectors for HLTA and CorEX, and as bags-of-words for nCRP and nHDP.

| Vocabulary Size | NIPS-1k | NIPS-5k | NIPS-10k | NEWS-1k | NEWS-5k | NYT  |
|-----------------|---------|---------|----------|---------|---------|------|
| Sample Size     | 1955    | 1955    | 1955     | 19940   | 19940   | 300,000 |

The NYT dataset is much larger than the other datasets. Among the four methods, only HLTA and nHDP can handle it through the adoption of stochastic EM and stochastic gradient descent. We will first report the results on the other datasets, and then present the results on the NYT dataset at the end.

6.2.2. Running times

For HLTA, the limit on the number of top-level latent variables was set at \( \tau = 20 \). On the NIPS and Newsgroup datasets, it produced hierarchies with between 4 to 6 levels. For nCRP and nHDP, the height of hierarchy was set at 3, as is usually done in the literature. The number of nodes at each level was set in such way that nCRP and nHDP would yield roughly the same total number of topics as HLTA. CorEx is configured similarly. Both nCRP and nHDP were allowed to run to 1,000 iterations. All experiments were conducted on the same desktop computer. Each experiment was

---
1http://qwone.com/~jason/20Newsgroups/
repeated 3 times so that variance can be estimated.

The running times are shown in Table 6. We see that in most cases HLTA are faster than nHDP, although the differences decrease with the sample size as nHDP works in a stochastic way in these datasets while HLTA doesn’t. It is much more efficient than nCRP and CorEx.

Table 6: Running times: The sign “-” indicates non-termination after 72 hours.

| Time(min) | Nips-1k  | Nips-5k  | Nips-10k | News-1k  | News-5k  |
|-----------|----------|----------|----------|----------|----------|
| HLTA      | 5.6 ± 0.1| 86.1 ± 3.7| 318.9 ± 12.5| 66.1 ± 2.9| 432.6 ± 39.3|
| nCRP      | 852.7 ± 149.8| 3,939.6 ± 301.0| —        | —        | —        |
| nHDP      | 379.2 ± 13.7| 416.3 ± 48.9| 412.7 ± 15.8| 250.2 ± 81.4| 332.1 ± 16.3|
| CorEx     | 53.1 ± 0.2| 371.8 ± 23.2| 1,190.8 ± 9.2| 778.8 ± 34.2| 4,287.0 ± 52.4|

6.2.3. Model Quality

For topic models, the standard way to assess the model quality is to measure the log-likelihood of a held-out test set. In our experiments, each dataset was randomly partitioned into a training set with 80% of the data, and a test set with 20% of the data. Models were learned from the training set and per-document held-out loglikelihood statistics for the models were calculated on the test set. The statistics are shown in Table 7.

It should be noted that the likelihood values for HLTA and CorEx were calculated on binary data while those for nCRP and nHDP were calculated on bag-of-words data. For comparability, we also computed the held-out loglikelihood for nHDP on the binary version of the datasets. The results are denoted as nHDP-b. The results on the count data (bag-of-words) are denoted as nHDP-c.

We see that the likelihood values for HLTA are significantly higher than those for CorEx. They are also higher than nHDP-b, and drastically higher than those for nCRP and nHDP. In addition, the variance is significantly smaller for HLTA than other methods in most cases.
Table 7: Per-document loglikelihood. Best scores are marked in bold.

|         | Nips-1k   | Nips-5k   | Nips-10k  | News-1k  | News-5k  |
|---------|-----------|-----------|-----------|----------|----------|
| HLTA    | -393±0    | -1,121±1  | -1,428±1  | -114±0   | -242±0   |
| nCRP    | -2,951±35 | -5,626±117| —         | —        | —        |
| nHDP-c  | -4,107±6  | -7,986±11 | -9,147±56 | -329±1   | -686±3   |
| nHDP-b  | -1,344±3  | -3,520±7  | -4,245±25 | -242±1   | -471±2   |
| CorEx   | -445±2    | -1,243±2  | -1,610±4  | -149±1   | -323±4   |

6.2.4. Quality of Topics

It has been argued that, in general, better model fit does not necessarily imply better topic quality [35]. It might therefore be more meaningful to compare alternative methods in terms of topic quality. One metric for topic quality is the topic coherence score proposed by [36]. Suppose a topic \( t \) is characterized using the top \( M \) words. The coherence score of \( t \) is given by:

\[
C(t, W(t)) = \sum_{m=2}^{M} \sum_{l=1}^{m-1} \log \frac{D(w_m^{(t)}, w_l^{(t)}) + 1}{D(w_l^{(t)})},
\]

where \( D(w_i) \) is the number of documents containing word \( w_i \) and \( D(w_i, w_j) \) is the number of documents containing both \( w_i \) and \( w_j \). With \( M \) fixed, a higher coherence score indicates a better quality topic. In our experiments, we set \( M = 4 \).

Table 8 shows the average topic coherence scores of the topics produced by the four algorithms. We see that the quality of topics produced by HLTA is significantly higher than the quality of topics produced nHDP and other algorithms in all cases, and the variance is significantly smaller for HLTA than other methods in most cases.

6.2.5. Quality of Topic Hierarchies

There is no metric for measuring the quality of topic hierarchies to the best of our knowledge, and it is difficult to come up with one. Hence, we resort to manual comparison.

The entire topic hierarchies produced by HLTA and nHDP on the NIPS and NYT datasets can be found at the URL mentioned at the beginning of this section. Here
Table 8: Average topic coherence scores. Best scores are marked in bold.

|       | Nips-1k   | Nips-5k   | Nips-10k  | News-1k   | News-5k   |
|-------|-----------|-----------|-----------|-----------|-----------|
| HLTA  | -5.95±0.04| -7.74±0.07| -8.15±0.05| -12.00±0.09| -12.67±0.15|
| nCRP  | -7.46±0.31| -9.03±0.16| —         | —         | —         |
| nHDP  | -7.66±0.23| -9.70±0.19| -10.89±0.38| -13.51±0.08| -13.93±0.21|
| CorEx | -7.20±0.23| -9.76±0.48| -11.96±0.52| -13.49±1.48| -14.71±0.45|

we compare parts of the topic hierarchies obtained by the two algorithms on the NIPS dataset. For each top-level topic in Table 3 we identified the topic obtained by nHDP that is closest in meaning, which is shown in Table 9.

We refer to topics from HLTA in Table 3 using the letter ‘T’ followed by topic numbers, and those from nHDP in Table 9 using ‘\(\hat{T}\)’. For HLTA, the topic T1 is about mixture models, and its subtopics are concerned with various aspects of mixture models. For nHDP, the topic \(\hat{T}1\) and its children \(\hat{T}1.1\), \(\hat{T}1.2\) and \(\hat{T}1.5\) are about probability/mixture models. However, \(\hat{T}1.3\) and \(\hat{T}1.4\) do not fit in the group well. In addition, it is not clear what \(\hat{T}1.2\) and \(\hat{T}1.4\) mean.

The HLTA topic T3 is about image analysis and its subtopics give a clear spectrum of different aspects of image analysis. The nHDP topic \(\hat{T}3\) is also about image analysis. However, its subtopics do not give a clear spectrum of different aspects of image analysis. In addition, \(\hat{T}3.2\) does not fit in the group well, and the meanings of \(\hat{T}3.4\) and \(\hat{T}3.5\) are not clear. Furthermore, the subtopic \(\hat{T}6.2\) is about image analysis, but is not placed under T3.

The HLTA topic T4 is about speech recognition and its subtopics cover different aspects of speech recognition. In contrast, the nHDP topic \(\hat{T}4\) and its subtopics do not present a clear semantic hierarchy. Some of them are not meaningful. There are two other nHDP topics that are related to speech recognition, namely \(\hat{T}1.5\) and \(\hat{T}3.2\), and they are placed elsewhere.

Overall, the topics and topic hierarchy obtained by HLTA are more meaningful than those by nHDP.
Table 9: Parts of the topic hierarchies obtained by nHDP on Nips-10k.

1. gaussian likelihood mixture density Bayesian
   1.1. gaussian density likelihood Bayesian
   1.2. frey hidden posterior chaining log
   1.3. classifier classifiers confidence
   1.4. smola adaboost onoda mika svms
   1.5. speech context hme hmm experts
2. units hidden unit layer weight
   2.1. units hidden net unit weight
   2.2. units unit layer node rule
   2.3. humphreys neocortex knierim sagi
   2.4. control architecture signal memory
   2.5. wp faces correspondences avoidance
3. image recognition images feature features
   3.1. image recognition feature images object
   3.2. smola adaboost onoda utterance
   3.3. object matching shape image features
   3.4. nearest basis examples rbf classifier
   3.5. tangent distance simard distances
4. rules language rule sequence context
   4.1. recognition speech mlp word trained
   4.2. rules rule stack machine examples
   4.3. voicing syllable fault faults units
   4.4. rules hint table hidden structure
   4.5. syllable stress nucleus heavy bit
5. action policy reinforcement optimal actions
   5.1. novelty control robot reinforcement
   5.2. asynchronous tsitsiklis bertsekas
   5.3. states variable table search decision
   5.4. action actions policy agent states
   5.5. optimal action states tasks mdp
6. control trajectory motor robot controller
   6.1. control units motion robot examples
   6.2. recognition regression image object
   6.3. trajectory control speech forward
   6.4. arm field motor internal sensory
   6.5. controller rule rules position hand

6.2.6. Comparisons on the NYT Dataset

On the NYT dataset, we only compare HLTA with nHDP. Table [10] reports the running times and coherence scores of the topics obtained. HLTA took around 11 hours, which is slightly longer than the time that nHDP took (10.5 hours). However, HLTA enjoys significantly higher topic coherence score.
The topic hierarchy produced by HLTA is also more meaningful. Table 11 shows the part of the hierarchy by nHDP that corresponds to the part of the hierarchy by HLTA shown in Table 4. In the HLTA hierarchy, the topics are nicely divided into three groups, economy, stock market, and companies. In Table 11, there is no such clear division. The topics are all mixed up. The hierarchy does not match the semantic relationships among the topics.

Table 10: Performances on the New York Times data.

| Time (min) | Average topic coherence |
|------------|-------------------------|
| HLTA       | 670                     | -12.86                   |
| hHDP       | 637                     | -13.35                   |

Table 11: A part of topic hierarchy by nHDP.

1. company business million companies money
   1.1. economy economic percent government
   1.2. percent stock market analyst quarter
       1.2.1. stock fund market investor investment
       1.2.2. economy rate rates fed economist
       1.2.3. company quarter million sales analyst
       1.2.4. travel ticket airline flight traveler
       1.2.5. car ford sales vehicles chrysler
   1.3. computer technology system software
   1.4. company deal million billion stock
   1.5. worker job union employees contract
   1.6. project million plan official area

7. Related Work

There are some other methods for learning latent tree models. We refer the readers to [37] for a detailed survey. Most of the methods are designed for density estimation [38], latent structure discovery [39], and multi-dimensional clustering [14]. None of these methods are designed for topic detection. These methods do not build
hierarchical structures and do not provide a way to extract the topics from models when they are applied on text data.

HLTA also resembles hierarchical clustering (HC). However, there are fundamental differences. First, HLTA learns probabilistic graphical models which allow inference among variables, while HC builds dendrograms which are not probabilistic models and do not allow inference among nodes. Second, an HLTA can be seen as a clustering tool which clusters the data points and variables simultaneously, while HC can only be used to cluster either data points or variables.

8. Conclusions and Future Directions

We propose a novel method called HLTA for hierarchical topic detection. The idea is to model patterns of word co-occurrence and co-occurrence of those patterns using a hierarchical latent tree model. Each latent variable in HLTM represents a soft partition of documents. The document clusters in each partition are interpreted as topics. Each topic is characterized using the words that occur with high probability in documents belonging to the topic and occur with low probability in documents not belonging to the topic. Progressive EM is used to accelerate model learning. Empirical results show that HLTA outperforms nHDP, the state-of-the-art LDA-based method for hierarchical topic detection, in terms of both quality of topics and topic hierarchy, with comparable speed on large-scale data.

HLTA treats words as binary variables and each word is allowed to appear in only one branch of a hierarchy. For future work, it would be interesting to extend HLTA so that it can handle count data and a word is allowed to appear in multiple branches of the hierarchy. Another direction is to further scale up HLTA through parallelization.

9. Acknowledgments

We thank John Paisley for sharing the nHDP implements with us, and we thank Farhan Khawar and Jun Zhu for valuable discussions. Research on this article was supported by Hong Kong Research Grants Council under grant 16202515 and the Hong Kong Institute of Education under project RG90/2014-2015R.
References

[1] D. M. Blei, A. Y. Ng, M. I. Jordan, Latent dirichlet allocation, the Journal of machine Learning research 3 (2003) 993–1022.

[2] J. Lafferty, D. M. Blei, Correlated topic models, in: Advances in Neural Information Processing Systems, Citeseer, 2006, pp. 147–155.

[3] D. M. Blei, J. D. Lafferty, Dynamic topic models, in: Proceedings of the 23rd international conference on Machine learning. ACM, 2006, pp. 113–120.

[4] X. Wang, A. McCallum, Topics over time: a non-markov continuous-time model of topical trends, in: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2006, pp. 424–433.

[5] A. Ahmed, E. P. Xing, Dynamic non-parametric mixture models and the recurrent chinese restaurant process, Carnegie Mellon University, School of Computer Science, Machine Learning Department, 2007.

[6] D. Andrzejewski, X. Zhu, M. Craven, Incorporating domain knowledge into topic modeling via dirichlet forest priors, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 25–32.

[7] J. Jagarlamudi, H. Daumé III, R. Udupa, Incorporating lexical priors into topic models, in: Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, 2012, pp. 204–213.

[8] Y. W. Teh, M. I. Jordan, M. J. Beal, D. M. Blei, Hierarchical dirichlet processes, Journal of the american statistical association 101 (476).

[9] D. M. Blei, T. Griffiths, M. Jordan, J. Tenenbaum, Hierarchical topic models and the nested chinese restaurant process, Advances in neural information processing systems 16 (2004) 106–114.
[10] D. M. Blei, T. Griffiths, M. Jordan, The nested Chinese restaurant process and Bayesian nonparametric inference of topic hierarchies, Journal of the ACM 57 (2) (2010) 7:1–7:30.

[11] W. Li, A. McCallum, Pachinko allocation: Dag-structured mixture models of topic correlations, in: Proceedings of the 23rd international conference on Machine learning, ACM, 2006, pp. 577–584.

[12] J. Paisley, C. Wang, D. M. Blei, M. Jordan, et al., Nested hierarchical dirichlet processes, Pattern Analysis and Machine Intelligence, IEEE Transactions on 37 (2) (2015) 256–270.

[13] N. L. Zhang, Hierarchical latent class models for cluster analysis, Journal of Machine Learning Research 5 (6) (2004) 697–723.

[14] T. Chen, N. L. Zhang, T. Liu, K. M. Poon, Y. Wang, Model-based multidimensional clustering of categorical data, Artificial Intelligence 176 (2012) 2246–2269.

[15] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann Publishers, San Mateo, California, 1988.

[16] D. J. Bartholomew, M. Knott, Latent Variable Models and Factor Analysis, 2nd Edition, Arnold, 1999.

[17] G. J. McLachlan, D. Peel, Finite Mixture Models, Wiley, New York, 2000.

[18] J. A. Hagenaars, A. L. McCutcheon (Eds.), Applied Latent Class Analysis, Cambridge, New York, 2002.

[19] T. Liu, N. L. Zhang, P. Chen, A. H. Liu, K. M. Poon, Y. Wang, Greedy learning of latent tree models for multidimensional clustering, Machine Learning 98 (1-2) (2013) 301–330.

[20] T. M. Cover, J. A. Thomas, Elements of Information Theory, 2nd Edition, Wiley, 2006.
[21] C. K. Chow, C. N. Liu, Approximating discrete probability distributions with dependence trees., IEEE Transactions on Information Theory 14 (3) (1968) 462–467.

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

[23] G. Schwarz, Estimating the dimension of a model, The Annals of Statistics 6 (1978) 461–464.

[24] A. E. Raftery, Bayesian model selection in social research, Sociological Methodology 25 (1995) 111–163.

[25] R. E. Kass, A. E. Raftery, Bayes factor, Journal of American Statistical Association 90 (430) (1995) 773–795.

[26] C. K. Chow, C. N. Liu, Approximating discrete probability distributions with dependence trees, IEEE Transactions on Information Theory 14 (3) (1968) 462–467.

[27] K. P. Murphy, Machine Learning: A Probabilistic Perspective, The MIT Press, 2012.

[28] K. Pearson, Contributions to the mathematical theory of evolution, Philosophical Transactions of the Royal Society of London. A (1894) 71–110.

[29] J. T. Chang, Full reconstruction of markov models on evolutionary trees: identifiability and consistency, Mathematical biosciences 137 (1) (1996) 51–73.

[30] E. Mossel, S. Roch, Learning nonsingular phylogenies and hidden Markov models, The Annals of Applied Probability 16 (2) (2006) 583–614.

[31] A. Anandkumar, D. Hsu, S. M. Kakade, A method of moments for mixture models and hidden Markov models, in: Proceedings of the 25th Annual Conference on Learning Theory, 2012.
[32] N. L. Zhang, X. Wang, P. Chen, A study of recently discovered equalities about latent tree models using inverse edges, in: Probabilistic Graphical Models, Vol. 8754 of Lecture Notes in Computer Science, Springer, 2014, pp. 567–580.

[33] O. Bousquet, L. Bottou, The tradeoffs of large scale learning, in: Advances in neural information processing systems, 2008, pp. 161–168.

[34] G. Ver Steeg, A. Galstyan, Discovering structure in high-dimensional data through correlation explanation, in: Advances in Neural Information Processing Systems 27, 2014, pp. 577–585.

[35] J. Chang, S. Gerrish, C. Wang, J. L. Boyd-graber, D. M. Blei, Reading tea leaves: How humans interpret topic models, in: Advances in neural information processing systems, 2009, pp. 288–296.

[36] D. Mimno, H. M. Wallach, E. Talley, M. Leenders, A. McCallum, Optimizing semantic coherence in topic models, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2011, pp. 262–272.

[37] R. Mourad, C. Sinoquet, N. L. Zhang, T. Liu, P. Leray, et al., A survey on latent tree models and applications., J. Artif. Intell. Res.(JAIR) 47 (2013) 157–203.

[38] N. J. Choi, V. Y. F. Tan, A. Anandkumar, A. S. Willsky, Learning latent tree graphical models, Journal of Machine Learning Research 12 (2011) 1771–1812.

[39] T. Chen, N. L. Zhang, Y. Wang, Efficient model evaluation in the search-based approach to latent structure discovery, in: 4th European Workshop on Probabilistic Graphical Models, 2008, pp. 57–64.