Artificial Intelligence 250 (2017) 105–124

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Latent tree models for hierarchical topic detection

Peixian Chen [a], Nevin L. Zhang [a,*], Tengfei Liu [b], Leonard K.M. Poon [c], Zhourong Chen [a], Farhan Khawar [a]

[a] Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong
[b] Ant Financial Services Group, Shanghai, China
[c] Department of Mathematics and Information Technology, The Education University of Hong Kong, Hong Kong

ARTICLE INFO

Article history:
Received 2 May 2016
Received in revised form 18 June 2017
Accepted 26 June 2017
Available online 29 June 2017

Keywords:
Probabilistic graphical models
Text analysis
Hierarchical latent tree analysis
Hierarchical topic detection

ABSTRACT

We present 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 *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 binary latent variables that represent word co-occurrence patterns or co-occurrences of such patterns. 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. In comparison with LDA-based methods, a key advantage of the new method is that it represents co-occurrence patterns explicitly using model structures. Extensive empirical results show that the new method significantly outperforms the LDA-based methods in term of model quality and meaningfulness of topics and topic hierarchies.

1. Introduction

The objective of *hierarchical topic detection (HTD)* is, given a corpus of documents, to 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 mentioned 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.

* Corresponding author.
E-mail address: lzhang@cse.ust.hk (N.L. Zhang).

http://dx.doi.org/10.1016/j.artint.2017.06.004
0004-3702/

Several HTD methods have been proposed previously, including the nested Chinese restaurant process (nCRP) [1,2], the hierarchical Pachinko allocation model (hPAM) [3,4], and the nested hierarchical Dirichlet process (nHDP) [5]. Those methods are extensions of *latent Dirichlet allocation (LDA)* [6]. Hence we refer to them collectively as *LDA-based methods*.

In this paper, we present a novel HTD method called *hierarchical latent tree analysis (HLTA)*. Like the LDA-based methods, HLTA is a probabilistic method and it involves latent variables. However, there are fundamental differences. The first difference lies in the types of variables used in the models. In the LDA-based methods, observed variables are token variables (usually denoted as $W_{d,n}$), and latent variables are constructs in a hypothetical document generation process, including a list of topics (usually denoted as β), a topic distribution vector for each document (usually denoted as θ_d), and a topic assignment for each token in each document (usually denoted as $Z_{d,n}$). In contrast, each observed variable in HLTA stands for a word. It is a binary variable and represents the presence/absence of the word in a document. The latent variables in HLTA are considered as unobserved attributes of the documents. If we compare whether words occur in particular documents to whether students do well in various subjects, then the latent variables correspond to latent traits such as analytical skill, literacy skill, and general intelligence.

In the LDA-based methods, each token variable stands for a location in a document, and its possible values are the words in a vocabulary. Here one cannot talk about conditional independence between words because the probabilities of all words must sum to 1. On the other hand, the output of HLTA is a tree-structured graphical model, where the word variables are at the leaves and the latent variables are at the internal nodes. Two word variables are conditionally independent given any latent variable on the path between them. Words that frequently co-occur in documents tend to be located in the same "region" of the tree. This fact is conducive to the discovery of meaningful topics and topic hierarchies. A drawback of using binary word variables is that word counts are discarded.

The second difference lies in the definition and characterization of topics. Topics in the LDA-based methods are probabilistic distributions over a vocabulary. When presented to users, a topic is characterized using a few words with the highest probabilities. In contrast, topics in HLTA are clusters of documents. More specifically, all latent variables in HTLA are assumed to be binary. Just as the concept "analytical skill" partitions a student population into two soft clusters, with one cluster consisting of people with high analytic skill and the other consisting of people with low analytic skill, a latent variable in HLTA partitions a document collection into two soft clusters of documents. The document clusters are interpreted as topics. For presentation to users, a topic is characterized using the words that not only occur with high probabilities in topic but also occur with low probabilities outside the topic. The consideration of occurrence probabilities outside the topic is important because a word that occurs with high probability in the topic might also occur with high probability outside the topic. When that happens, it is not a good choice for the characterization of the topic.

HLTA also differs from the LDA-based methods in several other ways. Those differences are more technical in nature and will be explained Section 4.

The rest of the paper is organized as follows. We discuss related work in Section 2 and review the basics of latent tree models in Section 3. In Section 4, we introduce hierarchical latent tree models (HLTMs) and explain how they can be used for hierarchical topic detection. The HLTA algorithm for learning HLTMs is described in Sections 5–7. In Section 8, we present the results HTLA obtains on a real-world dataset and discuss some practical issues. In Section 9, we empirically compare HLTA with the LDA-based methods. Finally, we end the paper in Section 10 with some concluding remarks and discussions of future work.

2. Related work

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) [6]. LDA assumes that documents are generated as follows: First, a list $\{\beta_1, \ldots, \beta_K\}$ of topics is drawn from a Dirichlet distribution. Then, for each document d, a topic distribution θ_d is drawn from another Dirichlet distribution. Each word $W_{d,n}$ in the document is generated by first picking a topic $Z_{d,n}$ according to the topic distribution θ_d, and then selecting a word according to the word distribution $\beta_{Z_{d,n}}$ of the topic. Given a document collection, the generation process is reverted via statistical inference (sampling or variational inference) to determine the topics and topic compositions of the documents.

LDA has been extended in various ways for additional modeling capabilities. Topic correlations are considered in [7,3]; topic evolution is modeled in [8–10]; topic structures are built in [11,3,1,4]; side information is exploited in [12,13]; supervised topic models are proposed in [14,15]; and so on. In the following, we discuss in more details three of the extensions that are more closely related to this paper than others.

The hierarchical Pachinko allocation model (hPAM) [3,4] is proposed as a method for modeling correlations among topics. It introduces multiple levels of supertopics on top of the basic topics. Each supertopic is a distribution over the topics at the next level below. Hence hPAM can also be viewed as an HTD method, and the hierarchical structure needs to be predetermined. To pick a topic for a token, it first draws a top-level topic from a multinomial distribution (which in turn is drawn from a Dirichlet distribution), and then draws a topic for the next level below from the multinomial distribution associated with the top-level topic, and so on. The rest of the generation process is the same as in LDA.

The nested Chinese Restaurant Process (nCRP) [2] and the nested Hierarchical Dirichlet Process (nHDP) [5] are proposed as HTD methods. They assume that there is a true topic tree behind data. A prior distribution is placed over all possible trees using nCRP and nHDP respectively. An assumption is made as to how documents are generated from the true topic

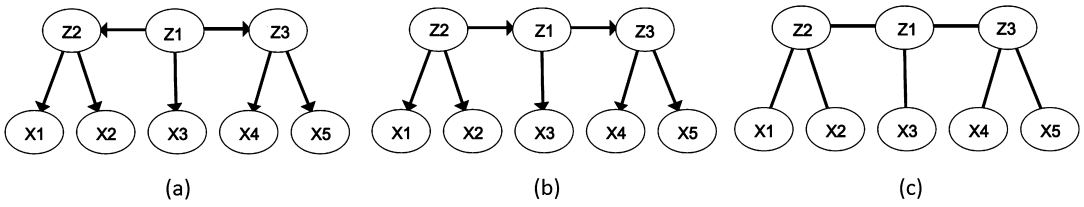


Fig. 1. The undirected latent tree model in (c) represents an equivalent class of directed latent tree models, which includes (a) and (b) as members.

tree, which, together with data, gives a likelihood function over all possible trees. In nCRP, the topics in a document are assumed to be from one path down the tree, while in nHDP, the topics in a document can be from multiple paths, i.e., a subtree within the entire topic tree. The true topic tree is estimated by combining the prior and the likelihood in posterior inference. During inference, one in theory deals with a tree with infinitely many levels and each node having infinitely many children. In practice, the tree is truncated so that it has a predetermined number of levels. In nHDP, each node also has a predetermined number of children, and nCRP uses a hyperparameter to control the number. As such, the two methods in effect require the user to provide the structure of an hierarchy as input.

As mentioned in the introduction, HLTA models document collections using hierarchical latent tree models (HLTMs) and the latent variables in the models are regarded as unobserved attributes of the documents. The concept of latent tree models was introduced in [16,17], where they were referred to as hierarchical latent class models. The term "latent tree models" first appeared in [18,19]. Latent tree models (LTMs) generalize two classes of models from the previous literature. The first class is latent class models [20,21], which are used for categorical data clustering in social sciences and medicine. The second class is probabilistic phylogenetic trees [22], which are a tool for determining the evolution history of a set of species.

The reader is referred to [23] for a survey of previous works on latent tree models. The works were conducted in three settings. In the first setting, data are assumed to be generated from an unknown LTM, and the task is to recover the generative model [24]. Here one tries to discover relationships between the latent structure and observed marginals that hold in LTMs, and then use those relationships to reconstruct the true latent structure from data. One can also prove theoretical results on consistency and sample complexity.

In the second setting, no assumption is made about how data are generated and the task is to fit an LTM to data [25]. In this setting it does not make sense to talk about theoretical guarantees on consistency and sample complexity. Instead, algorithms are evaluated empirically using held-out likelihood. It has been shown that, on real-world datasets, better models can be obtained using methods developed in this setting than using those developed in the first setting [26]. The reason is that, although the assumption of the first setting is reasonable for data from domains such as phylogeny, it is not reasonable for other types of data such as text data and survey data.

The third setting is similar to the second setting, except that model fit is no longer the only concern. In addition, one needs to consider how useful the resulting model is to users, and might want to, for example, obtain a hierarchy of latent variables. Liu et al. [27] are the first to use latent tree models for hierarchical topic detection. They propose an algorithm, namely HLTA, for learning HLTMs from text data and give a method for extracting topic hierarchies from the models. A method for scaling up the algorithm is proposed by Chen et al. [28]. This paper is based on [27,28]. There are substantial extensions: The novelty of HLTA w.r.t. the LDA-based methods is now systematically discussed; the theory and algorithm are described in more details and two practical issues are discussed; a new parameter estimation method is used for large datasets; and the empirical evaluations are more extensive.

Another method to learn a hierarchy of latent variables from data is proposed by Ver Steeg and Galstyan [29]. The method is named *correlation explanation (CorEx)*. Unlike HLTA, CorEx is a model-free method and it hence does not intend to provide a representation for the joint distribution of the observed variables.

HLTA produces a hierarchy with word variables at the bottom and multiple levels of latent variables on top. It is related to hierarchical variable clustering. However, there are fundamental differences. One difference is that HLTA defines a distribution over documents while variable clustering does not. In addition, HLTA also partitions document collections in multiple ways. There is a vast literature on document clustering [30]. In particular, co-clustering [31] can identify document clusters where each cluster is associated with a potentially different set of words. However, document clustering and topic detection are generally considered two different fields with little overlap. This paper bridges the two fields by developing a full-fledged HTD method that partitions documents in multiple ways.

3. Latent tree models

A *latent tree model (LTM)* is a tree-structured Bayesian network [32], where the leaf nodes represent observed variables and the internal nodes represent latent variables. An example is shown in Fig. 1(a). In this paper, all variables are assumed to be binary. The model parameters include a marginal distribution for the root Z_1, and a conditional distribution for each of the other nodes given its parent. The product of the distributions defines a joint distribution over all the variables.

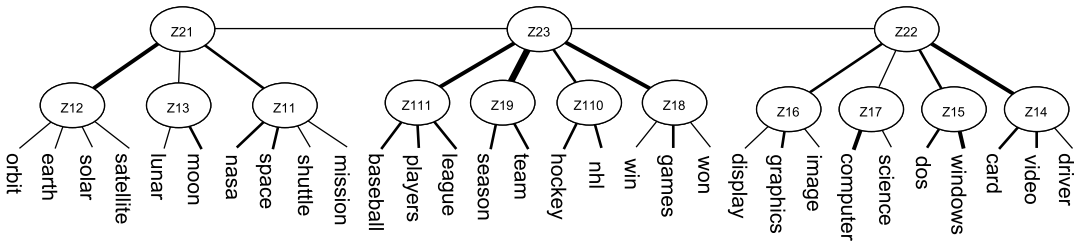


Fig. 2. Hierarchical latent tree model obtained from a toy text dataset. The latent variables right above the word variables represent word co-occurrence patterns and those at higher levels represent co-occurrence of patterns at the level below.

In general, an LTM has n observed variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ and m latent variables $\mathbf{Z} = \{Z_1, \ldots, Z_m\}$. Denote the parent of a variable Y as $pa(Y)$ and let $pa(Y)$ be the empty set when Y is the root. Then the LTM defines a joint distribution over all observed and latent variables as follows:

$$P(X_1, \ldots, X_n, Z_1, \ldots, Z_m) = \prod_{Y \in \mathbf{X} \cup \mathbf{Z}} P(Y \mid pa(Y)) \tag{1}$$

By changing the root from Z_1 to Z_2 in Fig. 1(a), we get another model shown in (b). The two models are *equivalent* in the sense that they represent the same set of distributions over the observed variables $X_1, \ldots, X_5$ [17]. It is not possible to distinguish between equivalent models based on data. This implies that the root of an LTM, and hence orientations of edges, are unidentifiable. It therefore makes more sense to talk about undirected LTMs, which is what we do in this paper. One example is shown in Fig. 1(c). It represents an equivalent class of directed models. A member of the class can be obtained by picking a latent node as the root and directing the edges away from the root. For example, (a) and (b) are obtained from (c) by choosing Z_1 and Z_2 to be the root respectively. In implementation, an undirected model is represented using an arbitrary directed model in the equivalence class it represents.

In the literature, there are variations of LTMs where some internal nodes are observed [24] and/or the variables are continuous [33–35]. In this paper, we focus on basic LTMs as defined in the previous two paragraphs.

We use $|Z|$ to denote the number of possible states of a variable Z. An LTM is *regular* if, for any latent node Z, we have that

$$|Z| \leq \frac{\prod_{i=1}^{k} |Z_i|}{\max_{i=1}^{k} |Z_i|}, \tag{2}$$

where $Z_1, \ldots, Z_k$ are the neighbors of Z, and that the inequality holds strictly when $k = 2$ [17]. When all variables are binary, the condition reduces to that each latent node must have at least three neighbors.

For any irregular LTM, there is a regular model that has fewer parameters and represents that same set of distributions over the observed variables [17]. Consequently, we focus only on regular models.

4. Hierarchical latent tree models and topic detection

We will later present an algorithm, called HLTA, for learning from text data models such as the one shown in Fig. 2. There is a layer of observed variables at the bottom and multiple layers of latent variables on top. The model is hence called a *hierarchical latent tree model (HLTM)*. In this section, we discuss how to interpret HLTMs and how to extract topics and topic hierarchies from them.

4.1. HLTMs for text data

We use the toy model in Fig. 2 as a running example. It is learned from a subset of the 20 Newsgroups data.[1] The variables at the bottom level, level 0, are observed binary variables that represent the presence/absence of words in a document. The latent variables at level 1 are introduced during data analysis to model word co-occurrence patterns. For example, $Z11$ captures the probabilistic co-occurrence of the words `nasa`, `space`, `shuttle` and `mission`; $Z12$ captures the probabilistic co-occurrence of the words `orbit`, `earth`, `solar` and `satellite`; $Z13$ captures the probabilistic co-occurrence of the words `lunar` and `moon`. Latent variables at level 2 are introduced during data analysis to model the co-occurrence of the patterns at level 1. For example, $Z21$ represents the probabilistic co-occurrence of the patterns $Z11$, $Z12$ and $Z13$.

Because the latent variables are introduced layer by layer, and each latent variable is introduced to explain the correlations among a group of variables at the level below, we regard, for the purpose of model interpretation, the edges between two layers as directed and they are directed downwards. (The edges between top-level latent variables are not directed.)

[1] http://qwone.com/~jason/20Newsgroups/.

Table 1
Document partition given by latent variable $Z21$.

$Z21$	$s0$ (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

This allows us to talk about the subtree rooted at a latent node. For example, the subtree rooted at Z_{21} consists of the observed variables orbit, earth, ..., mission.

4.2. Topics from HLTMs

There are totally 14 latent variables in the toy example. Each latent variable has two states and hence partitions the document collection into two soft clusters. To figure out what the partition and the two clusters are about, we need to consider the relationship between the latent variable and the observed variables in its subtree. Take $Z21$ as an example. Denote the two document clusters it gives as $Z21 = s0$ and $Z21 = s1$. The occurrence probabilities in the two clusters of the words in the $Z21$ subtree is given in Table 1, along with the sizes of the two clusters. We see that the cluster $Z21 = s1$ 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 meaningful and is interpreted as a *topic*. One might label the topic "NASA". The other cluster $Z21 = s0$ consists of 95% of the documents. In this cluster, the words occur with low probabilities. We interpret it as a *background topic*.

There are three subtle issues concerning Table 1. The first issue is how the word variables are ordered. To answer the question, we need the *mutual information* (MI) $I(X; Y)$ [36] between the two discrete variables X and Y, which is defined as follows:

$$I(X; Y) = \sum_{X,Y} P(X, Y) \log \frac{P(X, Y)}{P(X)P(Y)}. \tag{3}$$

In Table 1, the word variables are ordered according to their mutual information with $Z21$. The words placed at the top of the table have the highest MI with $Z21$. They are the best ones to characterize the difference between the two clusters because their occurrence probabilities in the two clusters differ the most. They occur with high probabilities in the clusters $Z21 = s1$ and with low probabilities in $Z21 = s0$. If one is to choose only the top, say 5, words to characterize the topic $Z21 = s1$, then the best words to pick are space, nasa, orbit, earth and shuttle.

The second issue is how the background topic is determined. The answer is that, among the two document clusters given by $Z21$, the one where the words occur with lower probabilities is regarded as the background topic. In general, we consider the sum of the probabilities of the top 3 words. The cluster where the sum is lower is designated to be the background topic and labeled $s0$, and the other one is considered a genuine topic and labeled $s1$.

Finally, the creation of Table 1 requires the joint distribution of $Z21$ with each of the words variable in its subtrees (e.g., $P(\text{space}, Z21)$). The distributions can be computed using Belief Propagation [32]. The computation takes linear time because the model is tree-structured.

4.3. Topic hierarchies from HLTMs

If the background topics are ignored, each latent variable gives us exactly one topic. As such, the model in Fig. 2 gives us 14 topics, which are shown in Table 2. 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. For example, the topic given by $Z22$ (windows, card, graphics, video, dos) consists of a mixture of words about several aspects of computers. We can say that the topic is about computers. The subtopics are each concerned with only one aspect of computers: $Z14$ (card, video, driver), $Z15$ (dos, windows), and $Z16$ (graphics, display, image).

4.4. More on novelty

In the introduction, we have discussed the differences between HLTA and the LDA-based methods with regard to the types of observed and latent variables used and the definition and characterization of topics. There are three other important differences. The third difference lies in the relationship between topics and documents. In the LDA-based methods, a document is a mixture of topics, and the probabilities of the topics within a document sum to 1. Because of this, the LDA

Table 2
Topic hierarchy given by the model in Fig. 2.

```
[0.05] space nasa orbit earth shuttle
  [0.06] orbit earth solar satellite
  [0.05] space nasa shuttle mission
  [0.03] moon lunar

[0.14] team games players season hockey
  [0.14] team season
  [0.11] players baseball league
  [0.09] games win won
  [0.08] hockey nhl

[0.24] windows card graphics video dos
  [0.12] card video driver
  [0.15] windows dos
  [0.10] graphics display image
  [0.09] computer science
```

models are sometimes called *mixed-membership models*. In HLTA, a topic is a soft cluster of documents, 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 between HLTA and the LDA-based methods is about the semantics of the hierarchies they produce. The LDA-based methods nCRP and nHDP produce a tree of topics, each of which being 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. Similarly, hPAM yields a collection of topics that are organized into a directed acyclic graph. The topics at the lowest level are distributions over the words, and topics at higher levels are distributions over the topics at the level below and hence are called super-topics. On the other hand, the output of HLTA is a tree of latent variables. Latent variables at high levels of the tree capture long-range word co-occurrence patterns and hence give thematically more general topics, while latent variables at low levels of the tree capture short-range word co-occurrence patterns and hence give thematically more specific topics.

Note that, in the context of document analysis, a common concept of hierarchy is a rooted 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-based methods yield such hierarchies.

Finally, LDA-based methods require the user to provide the structure of a hierarchy, including the number of latent levels and the number of nodes at each level. The number of latent levels is usually set at 3 out of efficiency considerations. The contents of the nodes (distributions over vocabulary) are learned from data. In contrast, HLTA learns both model structures and model parameters from data. The number of latent levels is not limited to 3.

5. Model structure construction

We present the HLTA algorithm in this and the next two sections. 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 4, and we will hence focus on how to learn the HLTM. In this section we will describe the procedures for constructing the model structure. In Section 6 we will discuss parameter estimation issues, and in Section 7 we will discuss techniques employed to accelerate the algorithm.

5.1. Top-level control

The top-level control of HLTA is given in Algorithm 1 and the subroutines are given in Algorithms 2–6. In this subsection, we illustrate the top-level control using the toy dataset mentioned in Section 4, which involves 30 word variables.

There are 5 steps. The first step (line 3) yields the model shown in Fig. 3(a). It is said to be a *flat LTM* because each latent variable is connected to at least one observed variable. In hierarchical models such as the one shown in Fig. 2, on the other hand, only the latent variables at the lowest latent layer are connected to observed variables, and other latent variables are not. The learning of a flat model is the key step of HLTA. We will discuss it in details later.

We refer to the latent variables in the flat model from the first step as level 1 latent variables. The objective of the second step (line 9) is to turn the level 1 latent variables into observed variables through data completion. To do so, the subroutine HardAssignment carries out inference to compute the posterior distribution of each latent variable for each document. The document is assigned to the state with the highest posterior probability, resulting in a dataset $\mathcal{D}_1$ over the level 1 latent variables.

In the third step, line 3 is executed again to learn a flat LTM for the level 1 latent variables, resulting the model shown in Fig. 3(b).

In the fourth step (line 7), the flat model for the level 1 latent variables is stacked on top of the flat model for the observed variables, resulting in the hierarchical model in Fig. 2. While doing so, the subroutine StackModels cuts off the links among the level 1 latent variables. The parameter values for the new model are copied from two source models.

Algorithm 1 HLTA($\mathcal{D}$, τ, μ, δ, κ).

Inputs: $\mathcal{D}$ – a collection of documents, τ – upper bound on the number of top-level topics, μ – upper bound on island size, δ – threshold used in UD-test, κ – number of EM steps on final model.
Outputs: An HLTM and a topic hierarchy.

1: $\mathcal{D}_1 \leftarrow \mathcal{D}$, $m \leftarrow null$.
2: **repeat**
3: $m_1 \leftarrow$ LearnFlatModel($\mathcal{D}_1$, δ, μ);
4: **if** $m = null$ **then**
5: $m \leftarrow m_1$;
6: **else**
7: $m \leftarrow$ StackModels(m_1, m);
8: **end if**
9: $\mathcal{D}_1 \leftarrow$ HardAssignment(m, $\mathcal{D}$);
10: **until** number of top-level nodes in $m \leq \tau$.
11: Run EM on m for κ steps.
12: **return** m and topic hierarchy extracted from m.

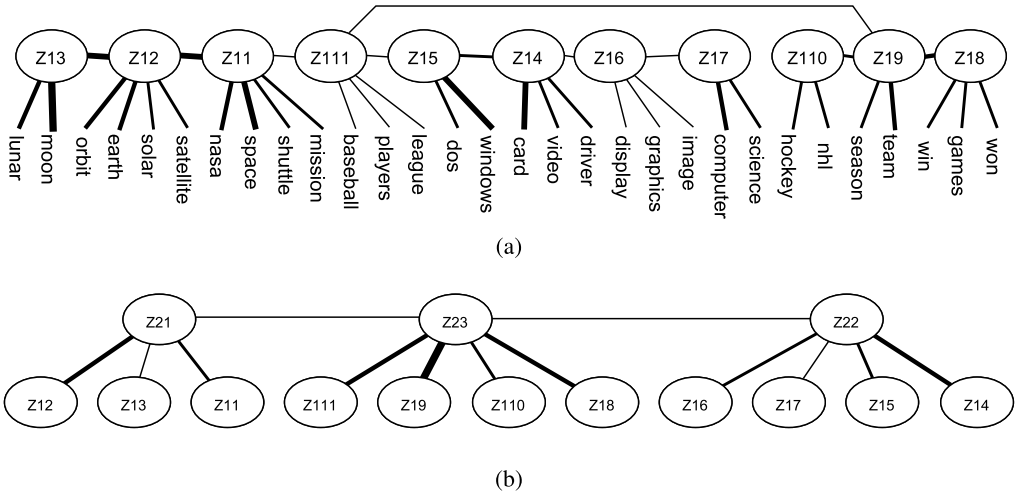


Fig. 3. Illustration of the top-level control of HLTA: (a) A flat model over the word variables is first learned; (b) The latent variables in (a) are converted into observed variables through data completion and another flat model is learned for them; Finally, the flat model in (b) is stacked on top of the flat model in (c) to obtain the hierarchical model in Fig. 2.

In general, the first four steps are repeated until the number of top-level latent variables falls below a user-specified upper bound τ (lines 2 to 10). In our running example, we set $\tau = 5$. The number of nodes at the top level in our current model is 3, which is below the threshold τ. Hence the loop is exited.

In the fifth step (line 11), the EM algorithm [37] is run on the final hierarchical model for κ steps to improve its parameters, where κ is another user specified input parameter.

The five steps can be grouped into two phases conceptually. The *model construction phase* consists of the first four steps. The objective is to build a hierarchical model structure. The *parameter estimation phase* consists of the fifth step. The objective is to optimize the parameters of the hierarchical structure from the first phase.

5.2. Learning flat models

The objective of the flat model learning step is to find, among all flat models, the one that has the highest BIC score. The BIC score [38] of a model m given a dataset $\mathcal{D}$ is defined as follows:

$$\mathcal{BIC}(m \mid \mathcal{D}) = \log P(\mathcal{D} \mid m, \theta^*) - \frac{d}{2} \log |\mathcal{D}|, \tag{4}$$

where θ^* is the maximum likelihood estimate of the model parameters, d is the number of free model parameters, and $|\mathcal{D}|$ is the sample size. Maximizing the BIC score intuitively means to find a model that fits the data well and that is not overly complex.

One way to solve the problem is through search. The state-of-the-art in this direction is an algorithm named EAST [25]. It has been shown in [26] to find better models than alternative algorithms such as BIN [39] and CLRG [24]. However, it does not scale up. It is capable of handling data with only dozens of observed variables and is hence not suitable for text analysis.

Algorithm 2 LearnFlatModel($\mathcal{D}$, δ, μ).

1: $\mathcal{L} \leftarrow$ BuildIslands($\mathcal{D}$, δ, μ);
2: $m \leftarrow$ BridgeIslands($\mathcal{L}$, $\mathcal{D}_1$);
3: **return** m.

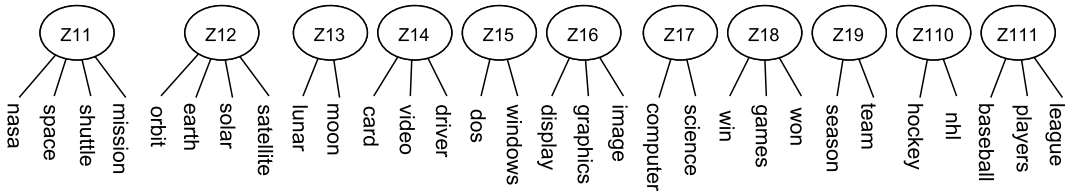


Fig. 4. The subroutine BuildIslands partitions word variables into uni-dimensional clusters and introduce a latent variable to each cluster to form an island (an LCM).

In the following, we present an algorithm that, when combined with the parameter estimation technique to be described in the next section, is efficient enough to deal with large text data. The pseudo code is given in Algorithm 2. It calls two subroutines. The first subroutine is BuildIslands. It partitions all 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. It then introduces a latent variable for each cluster to model the co-occurrence of the words inside it. Thus we obtain an LTM with a single latent variable for each word variable cluster, and it is called a *latent class model (LCM)*. In our running example, the results are shown in Fig. 4. We metaphorically refer to the LCMs as *islands*.

The second subroutine is BridgeIslands. It links up the islands by first estimating the mutual information between every pair of latent variables, and then finding the maximum spanning tree [40]. The result is the model in Fig. 3(a).

We now set out to describe the two subroutines in details.

5.2.1. 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 of variables is uni-dimensional.

To perform UD-test on a set $\mathcal{S}$ of observed variables, we first learn two latent tree models m_1 and m_2 for $\mathcal{S}$ 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. Fig. 5(b) shows what the two models might look like when $\mathcal{S}$ consists of four word variables `nasa`, `space`, `shuttle` and `mission`. We conclude that $\mathcal{S}$ is uni-dimensional if the following inequality holds:

$$\mathcal{BIC}(m_2 \mid \mathcal{D}) - \mathcal{BIC}(m_1 \mid \mathcal{D}) < \delta, \tag{5}$$

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

Note that the UD-test is related to the Bayes factor for comparing the two models [41]:

$$K = \frac{P(\mathcal{D}|m_2)}{P(\mathcal{D}|m_1)}. \tag{6}$$

The strength of evidence in favor of m_2 depends on the value of K. The following guidelines are suggested in [41]: If the quantity $2 \log K$ is from 0 to 2, the evidence is negligible; If it is between 2 and 6, there is positive evidence in favor of m_2; If it is between 6 to 10, there is strong evidence in favor of m_2; And if it is larger than 10, then there is very strong evidence in favor of m_2. Here, "log" stands for natural logarithm.

It is well known that the BIC score $\mathcal{BIC}(m|\mathcal{D})$ is a large sample approximation of the marginal loglikelihood $\log P(\mathcal{D}|m)$ [38]. Consequently, the difference $\mathcal{BIC}(m_2|\mathcal{D}) - \mathcal{BIC}(m_1|\mathcal{D})$ is a large sample approximation of the logarithm of the Bayes factor $\log K$. According to the cut-off values for the Bayes factor, we conclude that there is *positive, strong, and very strong* evidence favoring m_2 when the difference is larger than 1, 3 and 5 respectively. In our experiments, we always set $\delta = 3$.

5.2.2. Building islands

The subroutine BuildIslands (Algorithm 3) builds islands one by one. It builds the first island by calling another subroutine OneIsland (Algorithm 4). 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.

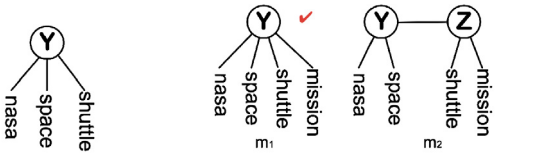
The subroutine OneIsland (Algorithm 4) requires a measurement of how closely correlated each pair of variables are. In this paper, mutual information is used for the purpose. The MI $I(X; Y)$ between the two variables X and Y is given by (3). We will also need the MI between a variable X and a set of variables $\mathcal{S}$. We estimate it as follows:

Algorithm 3 BUILDISLANDS($\mathcal{D}$, δ, μ).

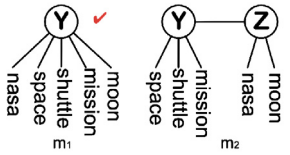
1: $\mathcal{V} \leftarrow$ variables in $\mathcal{D}$, $\mathcal{L} \leftarrow \emptyset$.
2: **while** $|\mathcal{V}| > 0$ **do**
3: $m \leftarrow$ ONEISLAND($\mathcal{D}$, $\mathcal{V}$, δ, μ);
4: $\mathcal{L} \leftarrow \mathcal{L} \cup \{m\}$;
5: $\mathcal{V} \leftarrow$ variables in $\mathcal{D}$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

Algorithm 4 ONEISLAND($\mathcal{D}$, $\mathcal{V}$, δ, μ).

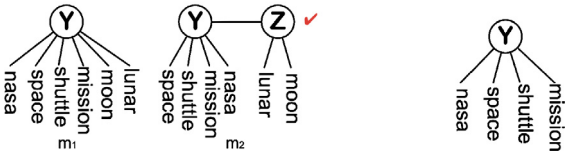
1: **if** $|\mathcal{V}| \leq 3$, $m \leftarrow$ LEARNLCM($\mathcal{D}$, $\mathcal{V}$), **return** m.
2: $\mathcal{S} \leftarrow$ two variables in $\mathcal{V}$ with highest MI;
3: $X \leftarrow \arg\max_{A \in \mathcal{V} \setminus \mathcal{S}} \hat{I}(A, \mathcal{S})$;
4: $\mathcal{S} \leftarrow \mathcal{S} \cup X$;
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: $X \leftarrow \arg\max_{A \in \mathcal{V}_1} \hat{I}(A, \mathcal{S})$;
10: $W \leftarrow \arg\max_{A \in \mathcal{S}} \hat{I}(A, X)$;
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.
17: **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**



(a) Initial island (b) UD-test passes after adding `mission`



(c) UD-test passes after adding `moon`



(d) UD-test fails after adding `lunar` (e) Final island

Fig. 5. Illustration of uni-dimensionality test and the ONEISLAND subroutine: For each pairs of models m_1 and m_2, m_1 is picked unless the BIC score of m_2 is significantly higher than that of m_1.

$$\hat{I}(X, \mathcal{S}) = \max_{A \in \mathcal{S}} I(X, A). \tag{7}$$

The subroutine ONEISLAND maintains a working set $\mathcal{S}$ of observed variables. Initially, $\mathcal{S}$ 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 $\mathcal{S}$ as the third variable (lines 3 and 4). Then other variables are added to $\mathcal{S}$ one by one. At each step, we pick the variable X that has the highest MI with the current set $\mathcal{S}$ (line 9), and perform UD-test on the set $\mathcal{S} \cup \{X\}$ (lines 12, 14, 15). If the UD-test passes, X is added to $\mathcal{S}$ (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 μ (line 18). In our experiments, we always set $\mu = 15$.

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 $\mathcal{S}$ 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 $\mathcal{S} \setminus \{W\}$ and the second latent variable connected to W and X. Both latent variables are binary and the model parameters are optimized by a fast subroutine PEM-LTM-2L that will be described in the next section.

We illustrate the ONEISLAND subroutine using an example in Fig. 5. 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 (Fig. 5(a)). 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 Fig. 5(b) 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 Fig. 5(a). 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 (Fig. 5(c)). 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 (Fig. 5(d)). 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` (Fig. 5(e)). 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.

5.2.3. 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 [42]. 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 at the end of Section 6.3.

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 (3) from the following joint distribution:

$$P(Y, Y' \mid \mathcal{D}, m, m') = C \sum_{d \in \mathcal{D}} P(Y \mid m, d) P(Y' \mid m', d) \tag{8}$$

where $P(Y \mid m, d)$ is the posterior distribution of Y in m given data case d , $P(Y' \mid m', d)$ is that of Y' in m', and C is the normalization constant.

In our running example, applying BRIDGEISLANDS to the islands in Fig. 4 results in the flat model shown in Fig. 3(a).

6. Parameter estimation during model construction

In the model construction phase, a large number of intermediate models are generated. Whether HLTA can scale up depends on whether the parameters of those 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. In the next section, we will discuss how to estimate the parameters of the final model efficiently when the sample size is very large.

6.1. The EM algorithm

We start by briefly reviewing the EM algorithm. Let $\mathbf{X}$ and $\mathbf{H}$ be respectively the sets of observed and latent variables in an LTM m, and let $\mathbf{V} = \mathbf{X} \cup \mathbf{H}$. Assume one latent variable is picked as the root and all edges are directed away from the root. For any V in $\mathbf{V}$ that is not the root, the parent $\mathrm{pa}(V)$ of V is a latent variable and can take values '0' or '1'. For technical convenience, let $\mathrm{pa}(V)$ 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

$$\theta_{ijk} = P(V_i = k | \mathrm{pa}(V_i) = j), \tag{9}$$

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

Given a dataset $\mathcal{D}$, the loglikelihood function of θ is given by

$$l(\theta \mid \mathcal{D}) = \sum_{d \in \mathcal{D}} \sum_{\mathbf{H}} \log P(d, \mathbf{H} \mid \theta). \tag{10}$$

The *maximum likelihood estimate (MLE)* of θ is the value that maximizes the loglikelihood function.

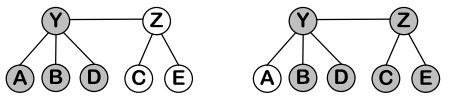


Fig. 6. Progressive EM: EM is first run in the submodel shaded in (a) to estimate the distributions $P(Y)$, $P(A|Y)$, $P(B|Y)$ and $P(D|Y)$, and then, EM is run in the submodel shaded in (b), with $P(Y)$, $P(B|Y)$ and $P(D|Y)$ fixed, to estimate the distributions $P(Z|Y)$, $P(C|Z)$ and $P(E|Z)$.

Due to the presence of latent variables, there is no closed form solution for the MLE. An iterative method called the *Expectation–Maximization (EM)* [37] 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. For latent tree models, the two steps are as follows:

- The E-step:

$$n_{ijk}^{(t)} = \sum_{d \in \mathcal{D}} P(V_i = k, pa(V_i) = j | d, m, \theta^{(t)}). \tag{11}$$

- The M-step:

$$\theta_{ijk}^{(t+1)} = \frac{n_{ijk}^{(t)}}{\sum_k n_{ijk}^{(t)}}. \tag{12}$$

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 [43].

EM terminates when the improvement 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.

6.2. Progressive EM

Being an iterative algorithm, EM can be trapped in local maxima. It is also time-consuming and does not scale up well. Progressive EM is proposed as a fast alternative to EM for the model construction phase. 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 Fig. 6. Assume Y is selected to be the root. To estimate all the parameters of the model, we first run EM in the part of the model shaded in Fig. 6(a) to estimate $P(Y)$, $P(A|Y)$, $P(B|Y)$ and $P(D|Y)$, and then run EM in the part of the model shaded in Fig. 6(b), with $P(Y)$, $P(B|Y)$ and $P(D|Y)$ fixed, to estimate $P(Z|Y)$, $P(C|Z)$ and $P(E|Z)$.

6.3. 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 4). 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 $\mathcal{S}$ 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 $\mathcal{S}$ that has maximum MI with $\mathcal{S}$, and the variable W inside $\mathcal{S}$ 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 5), which is an application of progressive EM. Let us explain Pem-Lcm using the intermediate models shown in Fig. 5. Let m be the model shown on the left of Fig. 5(c) and $\mathcal{S} = \{\texttt{nasa}, \texttt{space}, \texttt{shuttle}, \texttt{mission}, \texttt{moon}\}$. 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 Fig. 5(d). The only distribution to be estimated is $P(lunar|Y)$, as other distributions have already been estimated. Pem-Lcm estimates the distribution by running EM on a part of the model m_1 in Fig. 7 (left), where the variables involved are marked with rectangles. The variables `nasa` and `space` are included in the submodel, instead of other observed variables, because they were the seed variables picked at line 2 of Algorithm 4.

At line 14, 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 6),[2] which is also an application of progressive EM. In our running example, let `moon` be the variable W that has the highest MI with `lunar` among all variables in $\mathcal{S}$. Then the model m_2

[2] When one of the seed variables is W, use the other seed variable and the variable picked at line 3 of Algorithm 4.

Algorithm 5 PEM-LCM(m, $\mathcal{S}$, X, $\mathcal{D}$).

1: $Y \leftarrow$ the latent variable of m;
2: $\mathcal{S}_1 \leftarrow \{X\} \cup$ two seed variables in $\mathcal{S}$;
3: While keeping the other parameters fixed, run EM in the part of m that involves $\mathcal{S}_1 \cup Y$ to estimate $P(X|Y)$.
4: **return** m

Algorithm 6 PEM-LTM-2L($m, \mathcal{S} \setminus \{W\}, \{W, X\}, \mathcal{D}$).

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: $\mathcal{S}_1 \leftarrow \{W, X\} \cup$ two seed variables in $\mathcal{S}$;
4: While keeping the other parameters fixed, run EM in the part of m_2 that involves $\mathcal{S}_1 \cup Y \cup Z$ to only estimate $P(W|Z)$, $P(X|Z)$ and $P(Z|Y)$.
5: **return** m_2

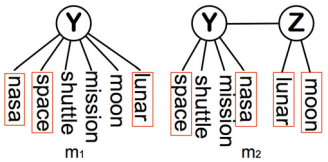


Fig. 7. Parameter estimation during island building: To determine whether the variable `lunar` should be added to the island m_1 in Fig. 5(c), two models are created. We need to estimate only $P(lunar|Y)$ for the model on the left, and $P(Z|Y)$, $P(moon|Z)$ and $P(lunar|Z)$ for the model on the right. The estimation is done by running EM in the parts of the models where the variables are marked with rectangles.

is as shown on the right hand side of Fig. 5(d). The distributions to be estimated are: $P(Z|Y)$, $P(moon|Z)$ and $P(lunar|Z)$. PEM-LTM-2L estimates the distributions by running EM on a part of the model m_2 in Fig. 7 (right), where the variables involved are marked with rectangles. The variables `nasa` and `space` are included in the submodel, instead of `shuttle` and `mission`, because they were the seed variables picked at line 2 of Algorithm 4.

There is also a parameter estimation problem inside the subroutine BRIDGEISLANDS. 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 Fig. 3 (a). To estimate $P(Z11|Z12)$, we form a sub-model by picking two children of $Z11$, for instance `nasa` and `space`, and two children of $Z12$, for instance `orbit` and `earth`. Then we estimate the distribution $P(Z11|Z12)$ by running EM in the submodel with all other parameters fixed.

6.4. Complexity analysis

Let n be the number of observed variables and N be the sample size. HLTA requires the computation of empirical MI between each pair of observed variables. This takes $O(n^2N)$ 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 variables 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 by c_1, on each submodel. This is the key reason why HLTA can scale up. The data projection takes $O(N)$ time for each submodel. Hence the total time for island building is $O(2n(N + c_1))$.

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 on the running time of this step is $n^2N + n(N + c_1)$. The total number of variables (observed and latent) in the resulting flat model is upper bounded by $2n$. Inference on the model takes no more than $2n$ propagation steps for each data case. Let c_2 be the time for each propagation step. Then the hard assignment step takes $O(4nc_2N)$ time. So, the total time for the first pass through the loop in HLTA is $O(2n^2N + 3n(N + c_1) + 4nc_2N) = O(2n^2N + 3nc_1 + 4nc_2N)$, where the term $3nN$ is ignored because it is dominated by the term $4nc_2N$.

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 $O(4n^2N + 6nc_1 + 8nc_2N)$.

The total number of variables (observed and latent) in the final model is upper bounded by $2n$. Hence, one EM iteration takes $O(4nc_2N)$ time and the final parameter optimization steps takes $O(4nc_2N\kappa)$ times.

The total running time of HLTA is $O(4n^2N + 6nc_1 + 8nc_2N) + O(4nc_2N\kappa)$. The two terms are the times for model construction phase and the parameter estimation phase respectively.

7. Dealing with large datasets

We employ two techniques to further accelerate HLTA so that it can handle large datasets with millions of documents. The first technique is downsampling and we use it to reduce the complexity of the model construction phase. Specifically,

we use a subset of N' randomly sampled data cases instead of the entire dataset and thereby reduce the complexity to $O(4n^2N' + 6nc_1 + 8nc_2N')$. When N is very large, we can set N' to be a small fraction of N and hence achieve substantial computational savings. In the meantime, we can still expect to obtain a good structure if N' is not too small. The reason is that model construction relies on salient regularities of data and those regularities should be preserved in the subset when N' is not too small.

The second technique is stepwise EM [44,45]. We use it to accelerate the convergence of the parameter estimation process in the second phase, where the task is to improve the values of the parameters $\theta = \{\theta_{ijk}\}$ (Equation (9)) obtained in the model construction phase. While standard EM, a.k.a. *batch EM*, updates the parameter once in each iteration, stepwise EM updates the parameters multiple times in each iteration.

Suppose the data set $\mathcal{D}$ is randomly divided into equal-sized minibatches $\mathcal{D}_1, \ldots, \mathcal{D}_B$. Stepwise EM updates the parameters after processing each minibatch. It maintains a collection of auxiliary variables n_{ijk}, which are initialized to 0 in our experiments. Suppose the parameters have been updated $u-1$ times before and the current values are $\theta = \{\theta_{ijk}\}$. Let $\mathcal{D}_b$ be the next minibatch to process. Stepwise EM carries out the updating as follows:

$$n'_{ijk} = \sum_{d \in \mathcal{D}_b} P(V_i = k, pa(V_i) = j | d, m, \theta), \tag{13}$$

$$n_{ijk} = (1 - \eta_u) n_{ijk} + \eta_u n'_{ijk}, \tag{14}$$

$$\theta_{ijk} = \frac{n_{ijk}}{\sum_k n_{ijk}}. \tag{15}$$

Note that equation (13) is similar to (11) except that the statistics are calculated on the minibatch $\mathcal{D}_b$ rather than the entire dataset $\mathcal{D}$. The parameter η_u is known as the *stepsize* and is given by $\eta_u = (u+2)^{-\alpha}$ and the parameter α is to be chosen the range $0.5 \leq \alpha \leq 1$ [46]. In all our experiments, we set $\alpha = 0.75$.

Stepwise EM is similar to stochastic gradient descent [47] in that it updates the parameters after processing each minibatch. It has been shown to yield estimates of the same or even better quality as batch EM and it converges much faster than the latter [46]. As such, we can run it for much fewer iterations than batch EM and thereby substantially reduce the running time.

8. Illustration of results and practical issues

HLTA is a novel method for hierarchical topic detection and, as discussed in the introduction, it is fundamentally different from the LDA-based methods. We will empirically compare HLTA with the LDA-based methods in the next section. In this section, we present the results HLTA obtains on a real-world dataset so that the reader can gain a clear understanding of what it has to offer. We also discuss two practical issues.

8.1. Results on the NYT dataset

HLTA is implemented in Java. The source code is available online,[3] along with the datasets used in this paper and the full details of the results obtained on them. HLTA has been tested on several datasets. One of them is the NYT dataset, which consists of 300,000 articles published on New York Times between 1987 and 2007.[4] A vocabulary of 10,000 words was selected using *average TF-IDF* [48] for the analysis. The average TF-IDF of a term t in a collection of documents $\mathcal{D}$ is defined as follows:

$$\texttt{tf-idf}(t, \mathcal{D}) = \frac{\sum_{d \in \mathcal{D}} \texttt{tf}(t, d) \cdot \texttt{idf}(t, \mathcal{D})}{|\mathcal{D}|}, \tag{16}$$

where $|\cdot|$ stands for the cardinality of a set, $\texttt{tf}(t, d)$ is the term frequency of t in document d, and $\texttt{idf}(t, \mathcal{D}) = \log(|\mathcal{D}|/|\{d \in \mathcal{D} : t \in d\}|)$ is the inverse document frequency of t in the document collection $\mathcal{D}$.

A subset of 10,000 randomly sampled documents was used in the model construction phase. Stepwise EM was used in the parameter estimation phase and the size of minibatches was set at 1,000. Other parameter settings are given in the next section. The analysis took around 420 minutes on a desktop machine.

The result is an HLTM with 5 levels of latent variables and 21 latent variables at the top level. Fig. 8 shows a part of the model structure. Four top-level latent variables are included in the figure. The level 4 and level 2 latent variables in the subtrees rooted at the five top-level latent variables are also included.

Each level 2 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. We see that most words in the subtree rooted at $Z501$ are about *economy and stock market*; most words in the subtree $Z502$ are about *companies and various industries*; most words in the subtree rooted at $Z503$ are about *movies and music*; and most words in the subtree rooted at $Z504$ are about *cooking*.

[3] http://www.cse.ust.hk/~lzhang/topic/index.htm.
[4] http://archive.ics.uci.edu/ml/datasets/Bag+of+Words.

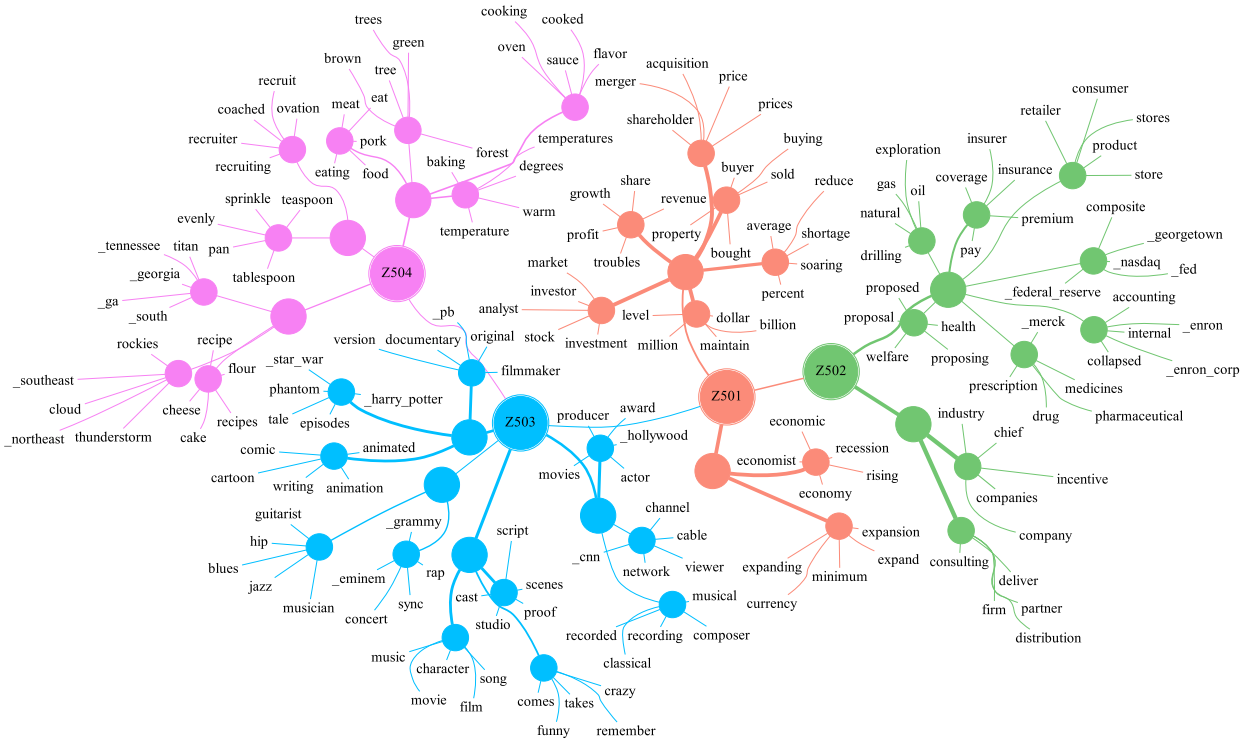


Fig. 8. Part of the hierarchical latent tree model obtained by HLTA on the NYT dataset: The nodes $Z501$ to $Z504$ are top-level latent variables. The subtrees rooted at different top-level latent variables are coded using different colors. Edge width represents mutual information. A bigger version of the figure is available at http://www.cse.ust.hk/~lzhang/topic/NYTimes/NYT-graph.pdf.

Table 3
A part of the topic hierarchy obtained by HLTA on the NYT dataset.

```
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 3 shows a part of the topic hierarchy extracted from the part of the model shown in Fig. 8. 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.

Table 4
Topics detected from AAAI/IJCAI papers (2000–15) that contain the word "network".

[0.05]	neural-network neural hidden-layer layer activation
[0.08]	bayesian-network probabilistic-inference variable-elimination
[0.04]	dynamic-bayesian-network dynamic-bayesian slice time-slice
[0.03]	domingos markov-logic-network richardson-domingos
[0.06]	dechter constraint-network freuder consistency-algorithm
[0.09]	social-network twitter social-media social tweet
[0.03]	complex-network community-structure community-detection
[0.01]	semantic-network conceptnet partof
[0.09]	wireless sensor-network remote radar radio beacon
[0.08]	traffic transportation road driver drive road-network

8.2. Two practical issues

Next we discuss two practical issues.

8.2.1. 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 other 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 broadly 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. Obviously, the fraction of documents that contain the seven words in the topic should be much smaller than 18%. However, those documents also 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 corpus. As a matter of fact, there are several other topics about American politics that are of similar sizes. 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 as follows: First, pick a list of words to characterize a topic using the method described in Section 4; then, form a latent class model using those words as observed variables; and finally, use the model to partition all documents into two clusters. The cluster where the words occur with relatively higher probabilities are designated as the narrow topic. 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 aforementioned topics are 0.008 and 0.169 respectively.

8.2.2. Use of collocations as observed variables

In HLTMs, each observed variable is connected to only one latent variable. If individual words are used as observed variables, then each word would appear in only one branch of the resulting topic hierarchy. This is not reasonable. Take the word "network" as an example. It can appear in different multi-word expressions such as "neural network", "Bayesian network", "constraint network", "social network", "sensor network", and so on. Clearly, those terms should appear in different branches in a good topic hierarchy.

Multi-word expressions such as "neural network" are known as collocations in the literature. In general, a *collocation* is sequence of consecutive words that has the characteristics of a syntactic and semantic unit [49]. It has been shown in [50] that replacing collocations with single tokens improves the performance of topic models. We borrow the idea to mitigate the problem mentioned above.

Specifically, we use the method described in [51]. The method first treats individual words as tokens and finds top n tokens with the highest average TF-IDF. Let PICK-TOP-TOKENS$(\mathcal{D}, n)$ be the subroutine which does that. The method then calculates the TF-IDF values of all 2-grams, and includes the top n 1-grams and 2-grams with highest TF-IDF values as tokens. After that, the selected 2-grams (e.g., "social network") are replaced with single tokens (e.g., "social-network") in all the documents and the subroutine PICK-TOP-TOKENS $(\mathcal{D}, n)$ is run again to pick a new set of n tokens. The process can be repeated if one wants to consider n-grams with $n > 2$ as tokens.

The method has been applied in an analysis of all the papers published at AAAI and IJCAI between 2000 and 2015. Table 4 shows some of the topics from the resulting topic hierarchy. They all contain the word "network" and are from different branches of the hierarchy.

Table 5
Information about the datasets used in empirical comparisons.

	NIPS-1k	NIPS-5k	NIPS-10k	News-1k	News-5k	NYT
Vocabulary size	1,000	5,000	10,000	1000	5000	10,000
Sample size	1,955	1,955	1,955	19,940	19,940	300,000

9. Empirical comparisons

We now present empirical results to compare HLTA with LDA-based methods for hierarchical topic detection, including the nested Chinese restaurant process (nCRP) [2], the nested hierarchical Dirichlet process (nHDP) [5] and the hierarchical Pachinko allocation model (hPAM) [4]. Also included in the comparisons is CorEx [29]. CorEx produces a hierarchy of latent variables, but not a probability model over all the variables. For comparability, we convert the results into a hierarchical latent tree model.[5] Implementations of all those methods were obtained online.[6]

9.1. Datasets

Three datasets were used in our experiments. The first one is the NYT dataset mentioned before. The second one is the 20 Newsgroups dataset first mentioned in Section 4. It consists of 19,940 newsgroup posts. The third one is the NIPS dataset, which consists of 1,955 articles published at the NIPS conference between 1988 and 1999.[7] Symbols, stop words and the words that occur less than 3 times were removed for all the datasets.

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. For the NYT dataset, only one version was chosen. So, the experiments were performed on six datasets. Information about them is given in Table 5. Each dataset has two versions: a binary version where word frequencies are discarded, and a bags-of-words version where word frequencies are kept. HLTA and CorEx were run only on the binary version because they can only process binary data. The methods nCRP, nHDP and hPAM were run on both versions and the results are denoted as *nCRP-bin, nHDP-bin, hPAM-bin, nCRP-bow, nHDP-bow* and *hPAM-bow* respectively. Note that the LDA-based algorithms can be run on binary datasets without modification because binary data can be represented as bags-of-words where no words appear more than once.

9.2. Settings

HLTA was run in two modes. In the first mode, denoted as *HLTA-batch*, the entire dataset was used in the model construction phase and batch EM was used in the parameter estimation phase. In the second mode, denoted as *HLTA-step*, a subset of N' randomly sampled data cases was used in the model construction phase and stepwise EM was used in the parameter estimation phase (see Section 7). In all experiments, N' was set at 10,000, the size of minibatch at 1,000, and the parameter α in stepwise EM at 0.75. HLTA-batch was run on all the datasets, while HLTA-step was run only on NYT and the Newsgroup datasets. HLTA-step was not run on the NIPS datasets because the sample size is too small. For HLTA-batch, the number κ of iterations for batch EM was set at 50. For HLTA-step, stepwise EM was terminated after 100 updates.

The other parameters of HLTA (see Algorithm 1) were set as follows in both modes: The threshold δ used in UD-tests was set at 3; the upper bound μ on island size was set at 15; and the upper bound τ on the number of top-level topics was set at 30 for NYT and 20 for all other datasets. When extracting topics from an HLTM (see Section 4), we ignored the level 1 latent variables because the topics they give are too fine-grained and often consist of different forms of the same word (e.g., "image, images").

The LDA-based methods nCRP, nHDP and hPAM do not learn model structures. A hierarchy needs to be supplied as input. In our experiment, the height of the 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, nHDP and hPAM would yield roughly the same total number of topics as HLTA. For example, HLTA produced 170 topics on the NIPS-1k dataset. To ensure that nHDP would produce a similar number of topics, we used 13 nodes for the top level and let each node have three children, leading to a total number of 169 topics. For hPAM, we used 100 subtopics, 70 supertopics and one root topic. In nCRP, the number of topics is controlled by a hyperparameter η. After some tuning, we found that setting η at 0.1, 0.05 and 0.025 respectively for the three levels result to approximately 170 topics. For CorEx, the number of nodes at each level was set to be the number of nodes at the corresponding level in HLTA.

[5] Let Y be a latent variable and $Z_1, \ldots, Z_k$ be its children. Let $Z_i^{(d)}$ be the value of Z_i in a data case d. It is obtained via hard assignment if Z_i is a latent variable. CorEx gives the distribution $P(Y|Z_1^{(d)}, \ldots, Z_k^{(d)})$ for data case d. Let $\mathbf{1}^{(d)}(Z_1, \ldots, Z_k)$ be a function that takes value 1 when $Z_i = Z_i^{(d)}$ for all i and 0 otherwise. Then, the expression $\sum_{d\in\mathcal{D}} P(Y|Z_1^{(d)}, \ldots, Z_k^{(d)})\mathbf{1}^{(d)}(Z_1, \ldots, Z_k)/|\mathcal{D}|$ defines a joint distribution over Y and Z_1–Z_k. From the joint distribution, we obtain $P(Z_i|Y)$ for each i, and also $P(Y)$ if Y is the root.

[6] nCRP: https://github.com/blei-lab/hlda; nHDP: http://www.columbia.edu/~jwp2128/code/nHDP.zip; hPAM: http://www.arbylon.net/projects/knowceans-lda-cgen/Hpam2pGibbsSampler.java; and CorEx: https://github.com/gregversteeg/CorEx.

[7] http://www.cs.nyu.edu/~roweis/data.html.

Table 6
Per-document held-out loglikelihood on binary data: Best scores are marked in bold. The sign "−" indicates non-termination after 72 hours, and "NR" stands for "not run'.

	NIPS-1k	NIPS-5k	NIPS-10k	News-1k	News-5k	NYT
HLTA-batch	**−393 ± 0**	**−1,121 ± 1**	**−1,428 ± 1**	**−114 ± 0**	**−242 ± 0**	**−754 ± 1**
HLTA-step	NR	NR	NR	**−114 ± 0**	−243 ± 0	−755 ± 1
nCRP-bin	−671 ± 16	−3,034 ± 135	–	–	–	–
hPAM-bin	−1,183 ± 3	–	–	−183 ± 2	–	–
nHDP-bin	−1,188 ± 1	−3,272 ± 3	−4,001 ± 11	−183 ± 1	−407 ± 2	−1,530 ± 5
CorEx	−445 ± 2	−1,243 ± 2	−1,610 ± 4	−149 ± 1	−323 ± 4	–

Table 7
Running times.

Time (min)	NIPS-1k	NIPS-5k	NIPS-10k	News-1k	News-5k	NYT
HLTA-batch	**5.6 ± 0.1**	**86.1 ± 3.7**	318 ± 12	66.1 ± 2.9	432 ± 39	787 ± 42
HLTA-step	NR	NR	NR	**9.6 ± 0.1**	**133 ± 5**	**421 ± 17**
nCRP-bin	782 ± 39	3,608 ± 163	–	–	–	–
nHDP-bin	152 ± 1	288 ± 16	**299 ± 9**	162 ± 13	263 ± 9	430 ± 42
hPAM-bin	261 ± 17	–	–	328 ± 9	–	–
nCRP-bow	853 ± 150	3,939 ± 301	–	–	–	–
nHDP-bow	379 ± 14	416 ± 49	413 ± 16	250 ± 18	332 ± 16	564 ± 59
hPAM-bow	850 ± 27	–	–	604 ± 19	–	–
CorEx	53 ± 0.2	371 ± 23	1,190 ± 9	779 ± 34	4,287 ± 52	–

All experiments were conducted on the same desktop computer. Each experiment was repeated 3 times so that variances can be estimated.

9.3. Model quality and running times

For topic models, a standard way to assess model quality is to measure log-likelihood on a held-out test set [2,5]. 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 loglikelihood was calculated on the held-out test set. The statistics are shown in Table 6. For comparability, only the results on binary data are included.

We see that the held-out likelihood values for HLTA are drastically higher than those for all the alternative methods. This implies that the models obtained by HLTA can predict unseen data much better than the other methods. In addition, the variances are significantly smaller for HLTA than the other methods in most cases.

Table 7 shows the running times. We see that, between the two versions of HLTA, HLTA-step is significantly more efficient than HLTA-batch on large datasets, while there is virtually no decrease in model quality. To compare HLTA with other methods, we need to keep in mind that all algorithms have parameters that control computational complexity. Thus, running time comparison is only meaningful when it is done with reference to model quality. It is clear from Tables 6 and 7 that HLTA achieved much better model quality than the alternative algorithms using comparable or less time.

9.4. Quality of topics

It has been argued that, in general, better model fit does not necessarily imply better topic quality [52]. It might therefore be more meaningful to compare alternative methods in terms of topic quality directly. We measure topic quality using two metrics. The first one is the *topic coherence score* proposed by [53]. Suppose a topic t is characterized using a list $W^{(t)} = \{w_1^{(t)}, w_2^{(t)}, \ldots, w_M^{(t)}\}$ of M words. The coherence score of t is given by:

$$\texttt{Coherence}(W^{(t)}) = \sum_{i=2}^{M} \sum_{j=1}^{i-1} \log \frac{D\left(w_i^{(t)}, w_j^{(t)}\right) + 1}{D\left(w_j^{(t)}\right)}, \tag{17}$$

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. It is clear that the score depends on the choices of M and it generally decreases with M. In our experiments, we set $M = 4$ because some of the topics produced by HLTA have only 4 words and hence the choice of a larger value for M would put other methods at a disadvantage. With M fixed, a higher coherence score indicates a better quality topic.

The second metric we use is the *topic compactness score* proposed by [54]. The compactness score of a topic t is given by:

$$\texttt{compactness}(W^{(t)}) = \frac{2}{M(M-1)} \sum_{i=2}^{M} \sum_{j=1}^{i-1} S(w_i^{(t)}, w_j^{(t)}), \tag{18}$$

Table 8
Average topic coherence scores.

	NIPS-1k	NIPS-5k	NIPS-10k	News-1k	News-5k	NYT
HLTA-batch	**−5.95 ± 0.04**	**−7.74 ± 0.07**	**−8.15 ± 0.05**	−12.00 ± 0.09	−12.67 ± 0.15	−12.09 ± 0.07
HLTA-step	NR	NR	NR	**−11.66 ± 0.19**	**−12.08 ± 0.11**	**−11.97 ± 0.05**
nCRP-bow	−7.46 ± 0.31	−9.03 ± 0.16	–	–	–	–
nHDP-bow	−7.66 ± 0.23	−9.70 ± 0.19	−10.89 ± 0.38	−13.51 ± 0.08	−13.93 ± 0.21	−12.90 ± 0.16
hPAM-bow	−6.86 ± 0.08	–	–	−11.74 ± 0.14	–	–
nCRP-bin	−7.01 ± 0.37	−9.83 ± 0.08	–	–	–	–
nHDP-bin	−8.95 ± 0.11	−11.59 ± 0.12	−12.34 ± 0.11	−13.45 ± 0.05	−14.17 ± 0.08	−14.55 ± 0.07
hPAM-bin	−6.83 ± 0.11	–	–	−12.63 ± 0.06	–	–
CorEx	−7.20 ± 0.23	−9.76 ± 0.48	−11.96 ± 0.52	−13.49 ± 1.48	−14.71 ± 0.45	–

Table 9
Average compactness scores.

	NIPS-1k	NIPS-5k	NIPS-10k	News-1k	News-5k	NYT
HLTA-batch	**0.253 ± 0.003**	**0.279 ± 0.008**	**0.265 ± 0.001**	0.239 ± 0.010	0.239 ± 0.006	0.337 ± 0.003
HLTA-step	NR	NR	NR	**0.250 ± 0.003**	**0.243 ± 0.002**	**0.338 ± 0.002**
nCRP-bow	0.163 ± 0.003	0.153 ± 0.001	–	–	–	–
nHDP-bow	0.164 ± 0.005	0.147 ± 0.006	0.138 ± 0.002	0.150 ± 0.003	0.148 ± 0.004	0.250 ± 0.003
hPAM-bow	0.215 ± 0.013	–	–	0.210 ± 0.006	–	–
nCRP-bin	0.176 ± 0.005	0.137 ± 0.005	–	–	–	–
nHDP-bin	0.119 ± 0.005	0.107 ± 0.003	0.102 ± 0.003	0.138 ± 0.003	0.134 ± 0.003	0.166 ± 0.001
hPAM-bin	0.145 ± 0.008	–	–	0.169 ± 0.010	–	–
CorEx	0.243 ± 0.018	0.162 ± 0.013	0.167 ± 0.003	0.185 ± 0.012	0.156 ± 0.009	–

where $S(w_i, w_j)$ is the similarity between the words w_i and w_j as determined by a *word2vec* model [55,56]. The *word2vec* model was trained on a part of the Google News dataset.[8] It contains about 100 billion word tokens and each word is mapped to a high dimensional vector. The similarity between two words is defined as the cosine similarity of the two corresponding vectors. When calculating $\texttt{compactness}(W^{(t)})$, words that do not occur in the *word2vec* model were simply skipped.

Note that the coherence score is calculated on the corpus being analyzed. In this sense, it is an *intrinsic* metric. The intuition is that words in a good topic should tend to co-occur in the documents. On the other hand, the compactness score is calculated on a general and very large corpus not related to the corpus being analyzed. Hence it is an *extrinsic* metric. The intuition here is that words in a good topic should be closely related semantically.

Tables 8 and 9 show the average topic coherence and topic compactness scores of the topics produced by various methods. For LDA-based methods, we reported their scores on both datasets of binary and bags-of-words versions. We see that the scores for the topics produced by HLTA are significantly higher than those obtained by other methods in all cases.

9.5. 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 comparisons.

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 the previous section. Table 10 shows the part of the hierarchy by nHDP that corresponds to the part of the hierarchy by HLTA shown in Table 3. In the HLTA hierarchy, the topics are nicely divided into three groups, *economy*, *stock market*, and *companies*. In Table 10, there is no such clear division. The topics are all mixed up. The hierarchy does not match the semantic relationships among the topics.

Overall, the topics and topic hierarchy obtained by HLTA are more meaningful than those by nHDP.

10. Conclusions and future directions

In this paper, we have presented 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 an HLTM represents a soft partition of documents and the document clusters in the partitions 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 parameter learning for intermediate models created during model construction, and stepwise EM is used to accelerate parameter learning for the final model.

[8] https://code.google.com/archive/p/word2vec/.

Table 10
A part of the topic hierarchy obtained by nHDP on the NYT dataset.

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

HLTA differs fundamentally from the LDA-based methods for hierarchical topic detection. While both approaches define a probability distribution over documents, they use different types of observed and latent variables. HLTA involves both structure learning and parameter learning, while the LDA-based methods involve only parameter learning. The notions of topics and topic hierarchies are also different. Empirical results show that HLTA outperforms the LDA-based methods in terms of overall model fit and quality of topics/topic hierarchies, while takes no more time than the latter.

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 that a word is allowed to appear in multiple branches of the hierarchy. Another direction is to further scale up HLTA via distributed computing and by other means.

Acknowledgements

We thank the anonymous reviewers for their valuable comments and suggestions, John Paisley for sharing the nHDP implements with us, and Jun Zhu for useful discussions. Research on this article was supported by Hong Kong Research Grants Council under grants 16202515 and 16212516, and The Education University of Hong Kong under project RG90/2014-2015R.

References

[1] D.M. Blei, T. Griffiths, M. Jordan, J. Tenenbaum, Hierarchical topic models and the nested Chinese restaurant process, in: NIPS, vol. 16, 2004, pp. 106–114.
[2] D.M. Blei, T. Griffiths, M. Jordan, The nested Chinese restaurant process and Bayesian nonparametric inference of topic hierarchies, J. ACM 57 (2) (2010) 7:1–7:30.
[3] W. Li, A. McCallum, Pachinko Allocation: DAG-structured mixture models of topic correlations, in: ICML, 2006, pp. 577–584.
[4] D. Mimno, W. Li, A. McCallum, Mixtures of hierarchical topics with pachinko allocation, in: ICML, 2007, pp. 633–640.
[5] J. Paisley, C. Wang, D.M. Blei, M. Jordan, et al., Nested hierarchical Dirichlet processes, IEEE Trans. Pattern Anal. Mach. Intell. 37 (2) (2015) 256–270.
[6] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent Dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022.
[7] J. Lafferty, D.M. Blei, Correlated topic models, in: NIPS, 2006, pp. 147–155.
[8] D.M. Blei, J.D. Lafferty, Dynamic topic models, in: ICML, 2006.
[9] X. Wang, A. McCallum, Topics over time: a non-Markov continuous-time model of topical trends, in: SIGKDD, 2006, pp. 424–433.
[10] A. Ahmed, E.P. Xing, Dynamic non-parametric mixture models and the recurrent Chinese restaurant process, in: ICDM, 2007.
[11] Y.W. Teh, M.I. Jordan, M.J. Beal, D.M. Blei, Hierarchical Dirichlet processes, J. Am. Stat. Assoc. 101 (2005) 476.
[12] D. Andrzejewski, X. Zhu, M. Craven, Incorporating domain knowledge into topic modeling via Dirichlet forest priors, in: ICML, 2009, pp. 25–32.
[13] J. Jagarlamudi, H. Daumé III, R. Udupa, Incorporating lexical priors into topic models, Association for Computational Linguistics, 2012, pp. 204–213.
[14] J.D. Mcauliffe, D.M. Blei, Supervised topic models, in: NIPS, 2008, pp. 121–128.
[15] A.J. Perotte, F. Wood, N. Elhadad, N. Bartlett, Hierarchically supervised latent Dirichlet allocation, in: NIPS, 2011, pp. 2609–2617.
[16] N.L. Zhang, Hierarchical latent class models for cluster analysis, in: Proceedings of the 18th National Conference on Artificial Intelligence, 2002.
[17] N.L. Zhang, Hierarchical latent class models for cluster analysis, J. Mach. Learn. Res. 5 (2004) 697–723.
[18] N.L. Zhang, S. Yuan, T. Chen, Y. Wang, Latent tree models and diagnosis in traditional Chinese medicine, Artif. Intell. Med. 42 (3) (2008) 229–245.
[19] Y. Wang, N.L. Zhang, T. Chen, Latent tree models and approximate inference in Bayesian networks, in: AAAI, 2008.
[20] P.F. Lazarsfeld, N.W. Henry, Latent Structure Analysis, Houghton Mifflin, Boston, 1968.
[21] M. Knott, D.J. Bartholomew, Latent Variable Models and Factor Analysis, vol. 7, Edward Arnold, 1999.
[22] R. Durbin, S.R. Eddy, A. Krogh, G. Mitchison, Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids, Cambridge University Press, 1998.
[23] 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. 47 (2013) 157–203.
[24] N.J. Choi, V.Y.F. Tan, A. Anandkumar, A.S. Willsky, Learning latent tree graphical models, J. Mach. Learn. Res. 12 (2011) 1771–1812.
[25] T. Chen, N.L. Zhang, T. Liu, K.M. Poon, Y. Wang, Model-based multidimensional clustering of categorical data, Artif. Intell. 176 (2012) 2246–2269.
[26] T. Liu, N.L. Zhang, P. Chen, A.H. Liu, K.M. Poon, Y. Wang, Greedy learning of latent tree models for multidimensional clustering, Mach. Learn. 98 (1–2) (2013) 301–330.
[27] T. Liu, N.L. Zhang, P. Chen, Hierarchical latent tree analysis for topic detection, in: ECML/PKDD, Springer, 2014, pp. 256–272.
[28] P. Chen, N.L. Zhang, L.K. Poon, Z. Chen, Progressive EM for latent tree models and hierarchical topic detection, in: AAAI, 2016.
[29] G. Ver Steeg, A. Galstyan, Discovering structure in high-dimensional data through correlation explanation, in: NIPS, vol. 27, 2014, pp. 577–585.
[30] M. Steinbach, G. Karypis, V. Kumar, et al., A comparison of document clustering techniques, in: KDD Workshop on Text Mining, Boston, vol. 400, 2000, pp. 525–526.
[31] I.S. Dhillon, Co-clustering documents and words using bipartite spectral graph partitioning, in: SIGKDD, ACM, 2001, pp. 269–274.

[32] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, 1988.
[33] L. Poon, N.L. Zhang, T. Chen, Y. Wang, Variable selection in model-based clustering: to do or to facilitate, in: ICML-10, 2010, pp. 887–894.
[34] S. Kirshner, Latent tree copulas, in: European Workshop on Probabilistic Graphical Models, 2012.
[35] H. Liu, A. Parikh, E. Xing, Nonparametric latent tree graphical models: inference, estimation, and structure learning, J. Mach. Learn. Res. 12 (2017) 663–707.
[36] T.M. Cover, J.A. Thomas, Elements of Information Theory, 2nd edition, Wiley, 2006.
[37] A. Dempster, N. Laird, D. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. B 39 (1) (1977) 1–38.
[38] G. Schwarz, Estimating the dimension of a model, Ann. Stat. 6 (1978) 461–464.
[39] S. Harmeling, C.K.I. Williams, Greedy learning of binary latent trees, IEEE Trans. Pattern Anal. Mach. Intell. 33 (6) (2011) 1087–1097.
[40] C.K. Chow, C.N. Liu, Approximating discrete probability distributions with dependence trees, IEEE Trans. Inf. Theory 14 (3) (1968) 462–467.
[41] A.E. Raftery, Bayesian model selection in social research, Sociol. Method. 25 (1995) 111–163.
[42] C.K. Chow, C.N. Liu, Approximating discrete probability distributions with dependence trees, IEEE Trans. Inf. Theory 14 (3) (1968) 462–467.
[43] K.P. Murphy, Machine Learning: A Probabilistic Perspective, The MIT Press, 2012.
[44] M.-A. Sato, S. Ishii, On-line EM algorithm for the normalized Gaussian network, Neural Comput. 12 (2) (2000) 407–432.
[45] O. Cappé, E. Moulines, On-line expectation-maximization algorithm for latent data models, J. R. Stat. Soc., Ser. B, Stat. Methodol. 71 (3) (2009) 593–613.
[46] P. Liang, D. Klein, Online EM for unsupervised models, in: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2009, pp. 611–619.
[47] O. Bousquet, L. Bottou, The tradeoffs of large scale learning, in: NIPS, 2008, pp. 161–168.
[48] G.G. Chowdhury, Introduction to Modern Information Retrieval, Facet Publishing, 2010.
[49] Y. Choueka, Looking for needles in a haystack or locating interesting collocational expressions in large textual databases, in: RIAO 88 (Recherche d'Information Assistée par Ordinateur) Conference, 1988, pp. 609–623.
[50] J.H. Lau, T. Baldwin, D. Newman, On collocations and topic models, ACM Trans. Speech Lang. Process. (TSLP) 10 (3) (2013) 10.
[51] L.K. Poon, N.L. Zhang, Topic browsing for research papers with hierarchical latent tree analysis, preprint, arXiv:1609.09188.
[52] J. Chang, S. Gerrish, C. Wang, J.L. Boyd-graber, D.M. Blei, Reading tea leaves: how humans interpret topic models, in: NIPS, 2009, pp. 288–296.
[53] 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, 2011, pp. 262–272.
[54] Z. Chen, N. Zhang, D.-Y. Yeung, P. Chen, Sparse Boltzmann machines with structure learning as applied to text analysis, in: AAAI, 2017.
[55] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: ICLR, 2013.
[56] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: NIPS, 2013, pp. 3111–3119.