Hierarchical Dirichlet Trees for Information Retrieval

Gholamreza Haffari
School of Computing Sciences
Simon Fraser University
ghaffar1@cs.sfu.ca

Yee Whye Teh
Gatsby Computational Neuroscience
University College London
ywteh@gatsby.ucl.ac.uk

Abstract

We propose a principled probabilisitc framework which uses trees over the vocabulary to capture similarities among terms in an information retrieval setting. This allows the retrieval of documents based not just on occurrences of specific query terms, but also on similarities between terms (an effect similar to query expansion). Additionally our principled generative model exhibits an effect similar to inverse document frequency. We give encouraging experimental evidence of the superiority of the hierarchical Dirichlet tree compared to standard baselines.

1 Introduction

Information retrieval (IR) is the task of retrieving, given a query, the documents relevant to the user from a large quantity of documents (Salton and McGill, 1983). IR has become very important in recent years, with the proliferation of large quantities of documents on the world wide web. Many IR systems are based on some relevance score function $R(j, q)$ which returns the relevance of document j to query q. Examples of such relevance score functions include term frequency-inverse document frequency (tf-idf) and Okapi BM25 (Robertson et al., 1992).

Besides the effect that documents containing more query terms should be more relevant (term frequency), the main effect that many relevance scores try to capture is that of inverse document frequency: the importance of a term is inversely related to the number of documents that it appears in, i.e. the popularity of the term. This is because popular terms, e.g. common and stop words, are often uninformative, while rare terms are often very informative. Another important effect is that related or co-occurring terms are often useful in determining the relevance of documents. Because most relevance scores do not capture this effect, IR systems resort to techniques like query expansion which includes synonyms and other morphological forms of the original query terms in order to improve retrieval results; e.g. (Riezler et al., 2007; Metzler and Croft, 2007).

In this paper we explore a probabilistic model for IR that simultaneously handles both effects in a principled manner. It builds upon the work of (Cowans, 2004) who proposed a hierarchical Dirichlet document model. In this model, each document is modeled using a multinomial distribution (making the bag-of-words assumption) whose parameters are given Dirichlet priors. The common mean of the Dirichlet priors is itself assumed random and given a Dirichlet hyperprior. (Cowans, 2004) showed that the shared mean parameter induces sharing of information across documents in the corpus, and leads to an inverse document frequency effect.

We generalize the model of (Cowans, 2004) by replacing the Dirichlet distributions with Dirichlet tree distributions (Minka, 2003), thus we call our model the *hierarchical Dirichlet tree*. Related terms are placed close by in the vocabulary tree, allowing the model to take this knowledge into account when determining document relevance. This makes it unnecessary to use ad-hoc query expansion methods, as related words such as synonyms will be taken into account by the retrieval rule. The structure of the tree is learned from data in an unsupervised fashion, using a variety of agglomerative clustering techniques.

We review the hierarchical Dirichlet document (HDD) model in section 2, and present our proposed hierarchical Dirichlet tree (HDT) document model in section 3. We describe three algorithms for constructing the vocabulary tree in section 4, and give encouraging experimental evidence of the superiority of the hierarchical Dirichlet tree compared to standard baselines in section 5. We conclude the paper in section 6.

2 Hierarchical Dirichlet Document Model

The probabilistic approach to IR assumes that each document in a collection can be modeled probabilistically. Given a query q, it is further assumed that relevant documents j are those with highest generative probability $p(q|j)$ for the query. Thus given q the relevance score is $R(j,q) = p(q|j)$ and the documents with highest relevance are returned.

Assume that each document is a bag of words, with document j modeled as a multinomial distribution over the words in j. Let V be the terms in the vocabulary, n_{jw} be the number of occurrences of term $w \in V$ in document j, and $\theta^{\text{flat}}_{jw}$ be the probability of w occurring in document j (the superscript "flat" denotes a flat Dirichlet as opposed to our proposed Dirichlet tree). (Cowans, 2004) assumes the following hierarchical Bayesian model for the document collection:

$$\begin{aligned} \boldsymbol{\theta}^{\text{flat}}_0 &= (\theta^{\text{flat}}_{0w})_{w\in V} \sim \text{Dirichlet}(\gamma\mathbf{u}) \\ \boldsymbol{\theta}^{\text{flat}}_j &= (\theta^{\text{flat}}_{jw})_{w\in V} \sim \text{Dirichlet}(\alpha\boldsymbol{\theta}^{\text{flat}}_0) \\ \mathbf{n}_j &= (n_{jw})_{w\in V} \sim \text{Multinomial}(\boldsymbol{\theta}^{\text{flat}}_j) \end{aligned} \tag{1}$$

In the above, bold face $\mathbf{a} = (a_w)_{w\in V}$ means that $\mathbf{a}$ is a vector with $|V|$ entries indexed by $w \in V$, and $\mathbf{u}$ is a uniform distribution over V. The generative process is as follows (Figure 1(a)). First a vector $\boldsymbol{\theta}^{\text{flat}}_0$ is drawn from a symmetric Dirichlet distribution with concentration parameter γ. Then we draw the parameters $\boldsymbol{\theta}^{\text{flat}}_j$ for each document j from a common Dirichlet distribution with mean $\boldsymbol{\theta}^{\text{flat}}_0$ and concentration parameter α. Finally, the term frequencies of the document are drawn from a multinomial distribution with parameters $\boldsymbol{\theta}^{\text{flat}}_j$.

The insight of (Cowans, 2004) is that because the common mean parameter $\boldsymbol{\theta}^{\text{flat}}_0$ is random, it induces dependencies across the document models in

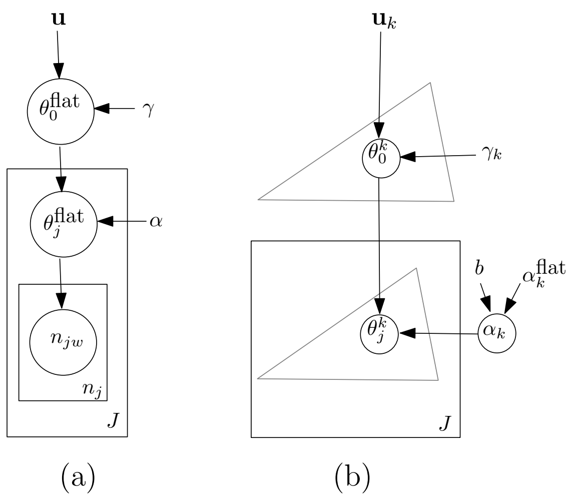

Figure 1: (a) The graphical model representation of the hierarchical Dirichlet document model. (b) The global tree and local trees in hierarchical Dirichlet tree document model. Triangles stand for trees with the same structure, but different parameters at each node. The generation of words in each document is not shown.

the collection, and this in turn is the mechanism for information sharing among documents. (Cowans, 2004) proposed a good estimate of $\boldsymbol{\theta}^{\text{flat}}_0$:

$$\theta^{\text{flat}}_{0w} = \frac{\gamma/|V| + n_{0w}}{\gamma + \sum_{w\in V} n_{0w}} \tag{2}$$

where n_{0w} is simply the number of documents containing term w, i.e. the document frequency. Integrating out the document parameters $\boldsymbol{\theta}^{\text{flat}}_j$, we see that the probability of query q being generated from document j is:

$$\begin{aligned} p(q|j) &= \prod_{x\in q} \frac{\alpha\theta^{\text{flat}}_{0x} + n_{jx}}{\alpha + \sum_{w\in V} n_{jw}} \\ &= \text{Const} \cdot \prod_{x\in q} \frac{\text{Const} + \frac{n_{jx}}{\gamma/|V| + n_{0x}}}{\alpha + \sum_{w\in V} n_{jw}} \end{aligned} \tag{3}$$

Where Const are terms not depending on j. We see that n_{jx} is term frequency, its denominator $\gamma/|V| + n_{0x}$ is an inverse document frequency factor, and $\alpha + \sum_{w\in V} n_{jw}$ normalizes for document length. The inverse document frequency factor is directly related to the shared mean parameter, in that popular terms x will have high $\theta^{\text{flat}}_{0x}$ value, causing all documents to assign higher probability to x, and down weighting the term frequency. This effect will be inherited by our model in the next section.

3 Hierarchical Dirichlet Trees

Apart from the constraint that the parameters should sum to one, the Dirichlet priors in the HDD model do not impose any dependency among the parameters of the resulting multinomial. In other words, the document models cannot capture the notion that related terms tend to co-occur together. For example, this model cannot incorporate the knowledge that if the word ‘computer’ is seen in a document, it is likely to observe the word ‘software’ in the same document. We relax the independence assumption of the Dirichlet distribution by using Dirichlet tree distributions (Minka, 2003), which can capture some dependencies among the resulting parameters. This allows relationships among terms to be modeled, and we will see that it improves retrieval performance.

3.1 Model

Let us assume that we have a tree over the vocabulary whose leaves correspond to vocabulary terms. Each internal node k of the tree has a multinomial distribution over its children $C(k)$. Words are drawn by starting at the root of the tree, recursively picking a child $l \in C(k)$ whenever we are in an internal node k, until we reach a leaf of the tree which corresponds to a vocabulary term (see Figure 2(b)). The Dirichlet tree distribution is the product of Dirichlet distributions placed over the child probabilities of each internal node, and serves as a (dependent) prior over the parameters of multinomial distributions over the vocabulary (the leaves).

Our model generalizes the HDD model by replacing the Dirichlet distributions in (1) by Dirichlet tree distributions. At each internal node k, define a hierarchical Dirichlet prior over the choice of the children:

$$\begin{aligned}\boldsymbol{\theta}_{0k} &= (\theta_{0l})_{l\in C(k)} \sim \mathrm{Dirichlet}(\gamma_k \mathbf{u}_k) \\ \boldsymbol{\theta}_{jk} &= (\theta_{jl})_{l\in C(k)} \sim \mathrm{Dirichlet}(\alpha_k \boldsymbol{\theta}_{0k})\end{aligned} \qquad (4)$$

where $\mathbf{u}_k$ is a uniform distribution over the children of node k, and each internal node has its own hyperparameters γ_k and α_k. θ_{jl} is the probability of choosing child l if we are at internal node k. If the tree is degenerate with just one internal node (the root) and all leaves are direct children of the root we recover the “flat” HDD model in the previous section. We call our model the *hierarchical Dirichlet tree* (HDT).

3.2 Inference and Learning

Given a term, the path from the root to the corresponding leaf is unique. Thus given the term frequencies $\mathbf{n}_j$ of document j as defined in (1), the number of times n_{jl} child $l \in C(k)$ was picked at node k is known and fixed. The probability of all words in document j, given the parameters, is then a product of multinomials probabilities over internal nodes k:

$$p(\mathbf{n}_j|\{\boldsymbol{\theta}_{jk}\}) = \prod_k \frac{n_{jk}!}{\prod_{l\in C(k)} n_{jl}!} \prod_{l\in C(k)} \theta_{jl}^{n_{jl}} \qquad (5)$$

The probability of the documents, integrating out the $\boldsymbol{\theta}_{jk}$’s, is:

$$p(\{\mathbf{n}_j\}|\{\boldsymbol{\theta}_{0k}\}) = \qquad (6)$$
$$\prod_j \prod_k \frac{n_{jk}!}{\prod_{l\in C(k)} n_{jl}!} \frac{\Gamma(\alpha_k)}{\Gamma(\alpha_k+n_{jk})} \prod_{l\in C(k)} \frac{\Gamma(\alpha_k\theta_{0l}+n_{jl})}{\Gamma(\alpha_k\theta_{0l})}$$

The probability of a query q under document j, i.e. the relevance score, follows from (3):

$$p(q|j) = \prod_{x\in q} \prod_{(kl)} \frac{\alpha_k\theta_{0l}+n_{jl}}{\alpha_k+n_{jk}} \qquad (7)$$

where the second product is over pairs (kl) where k is a parent of l on the path from the root to x.

The hierarchical Dirichlet tree model we proposed has a large number of parameters and hyperparameters (even after integrating out the $\boldsymbol{\theta}_{jk}$’s), since the vocabulary trees we will consider later typically have large numbers of internal nodes. This over flexibility might lead to overfitting or to parameter regimes that do not aid in the actual task of IR. To avoid both issues, we constrain the hierarchical Dirichlet tree to be centered over the flat hierarchical Dirichlet document model, and allow it to learn only the α_k hyperparameters, integrating out the $\boldsymbol{\theta}_{jk}$ parameters.

We set $\{\boldsymbol{\theta}_{0k}\}$, the hyperparameters of the global tree, so that it induces the same distribution over vocabulary terms as $\boldsymbol{\theta}_0^{\text{flat}}$:

$$\theta_{0l} = \theta_{0l}^{\text{flat}} \qquad \theta_{0k} = \sum_{l\in C(k)} \theta_{0l} \qquad (8)$$

The hyperparameters of the local trees α_k's are estimated using maximum a posteriori learning with likelihood given by (6), and a gamma prior with informative parameters. The density function of a $\mathrm{Gamma}(a, b)$ distribution is

$$g(x; a, b) = \frac{x^{a-1} b^a e^{-bx}}{\Gamma(a)}$$

where the mode happens at $x = \frac{a-1}{b}$. We set the mode of the prior such that the hierarchical Dirichlet tree reduces to the hierarchical Dirichlet document model at these values:

$$\alpha_l^{\text{flat}} = \alpha\theta_{0l}^{\text{flat}} \qquad \alpha_k^{\text{flat}} = \sum_{l \in C(k)} \alpha_l^{\text{flat}} \tag{9}$$

$$\alpha_k \sim \mathrm{Gamma}(b\alpha_k^{\text{flat}} + 1, b)$$

and $b > 0$ is an inverse scale hyperparameter to be tuned, with large values giving a sharp peak around α_k^{flat}. We tried a few values[1] of b and have found that the results we report in the next section are not sensitive to b. This prior is constructed such that if there is insufficient information in (6) the MAP value will simply default back to the hierarchical Dirichlet document model.

We used LBFGS[2] which is a gradient based optimization method to find the MAP values, where the gradient of the likelihood part of the objective function (6) is:

$$\frac{\partial \log p(\{\mathbf{n}_j\}|\{\boldsymbol{\theta}_{0j}\})}{\partial \alpha_k} = \sum_j \Psi(\alpha_k) - \Psi(\alpha_k + n_{jk}) + \sum_{l \in C(k)} \theta_{0l} \Big(\Psi(\alpha_k \theta_{0l} + n_{jl}) - \Psi(\alpha_k \theta_{0l}) \Big)$$

where $\Psi(x) := \partial \log \Gamma(x)/\partial x$ is the digamma function. Because each α_k can be optimized separately, the optimization is very fast (approximately 15-30 minutes in the experiments to follow on a Linux machine with 1.8 GH CPU speed).

4 Vocabulary Tree Structure Learning

The structure of the vocabulary tree plays an important role in the quality of the HDT document model, since it encapsulates the similarities among words captured by the model. In this paper we explored using trees learned in an unsupervised fashion from the training corpus.

Algorithm 1 Greedy Agglomerative Clustering

```
1: Place $m$ words into $m$ singleton clusters
2: **repeat**
3:     Merge the two clusters with highest similarity, resulting in one less cluster
4:     If there still are unincluded words, pick one and place it in a singleton cluster, resulting in one more cluster
5: **until** all words have been included and there is only one cluster left
```

The three methods are all agglomerative clustering algorithms (Duda et al., 2000) with different similarity functions. Initially each vocabulary word is placed in its own cluster; each iteration of the algorithm finds the pair of clusters with highest similarity and merges them, continuing until only one cluster is left. The sequence of merges determines a binary tree with vocabulary words as its leaves.

Using a heap data structure, this basic agglomerative clustering algorithm requires $\mathcal{O}(n^2 \log(n) + sn^2)$ computations where n is the size of the vocabulary and s is the amount of computation needed to compute the similarity between two clusters. Typically the vocabulary size n is large; to speed up the algorithm, we use a greedy version described in Algorithm 1 which restricts the number of cluster candidates to at most $m \ll n$. This greedy version is faster with complexity $\mathcal{O}(nm(\log m + s))$. In the experiments we used $m = 500$.

Distributional clustering (Dcluster) (Pereira et al., 1993) measures similarity among words in terms of the similarity among their local contexts. Each word is represented by the frequencies of various words in a window around each occurrence of the word. The similarity between two words is computed to be a symmetrized KL divergence between the distributions over neighboring words associated with the two words. For a cluster of words the neighboring words are the union of those associated with each word in the cluster. Dcluster has been used extensively in text classification (Baker and McCallum, 1998).

Probabilistic hierarchical clustering (Pcluster)

[1]Of the form 10^i for $i \in \{-2, -1, 0, 1\}$.

[2]We used a C++ re-implementation of Jorge Nocedal's LBFGS library (Nocedal, 1980) from the ALGLIB website: http://www.alglib.net.

(Friedman, 2003). Dcluster associates each word with its local context, as a result it captures both semantic and syntactic relationships among words. Pcluster captures more relevant semantic relationships by instead associating each word with the documents in which it appears. Specifically, each word is associated with a binary vector indexed by documents in the corpus, where a 1 means the word appears in the corresponding document. Pcluster models a cluster of words probabilistically, with the binary vectors being iid draws from a product of Bernoulli distributions. The similarity of two clusters c_1 and c_2 of words is $P(c_1 \cup c_2)/P(c_1)P(c_2)$, i.e. two clusters of words are similar if their union can be effectively modeled using one cluster, relative to modeling each separately. Conjugate beta priors are placed over the parameters of the Bernoulli distributions and integrated out so that the similarity scores are comparable.

Brown's algorithm (Bcluster) (Brown et al., 1990) was originally proposed to build class-based language models. In the 2-gram case, words are clustered such that the class of the previous word is most predictive of the class of the current word. Thus the similarity between two clusters of words is defined to be the resulting mutual information between adjacent classes corrresponding to a sequence of words.

4.1 Operations to Simplify Trees

Trees constructed using the agglomerative hierarchical clustering algorithms described in this section suffer from a few drawbacks. Firstly, because they are binary trees they have large numbers of internal nodes. Secondly, many internal nodes are simply not informative in that the two clusters of words below a node are indistinguishable. Thirdly, Pcluster and Dcluster tend to produce long chain-like branches which significantly slows down the computation of the relevance score.

To address these issues, we considered operations to simplify trees by contracting internal edges of the tree while preserving as much of the word relationship information as possible. Let L be the set of tree leaves and $\tau(a)$ be the distance from node or edge a to the leaves:

$$\tau(a) := \min_{l \in L} \#\{\text{edges between } a \text{ and } l\} \qquad (10)$$

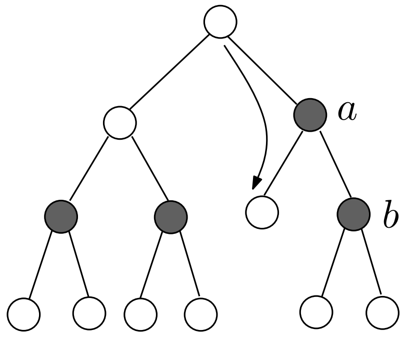

Figure 2: $\tau(\text{root}) = 2$, while $\tau(v) = 1$ for shaded vertices v. Contracting a and b results in both child of b being direct children of a while b is removed.

In the experiments we considered either contracting edges[3] close to the leaves $\tau(a) = 1$ (thus removing many of the long branches described above), or edges further up the tree $\tau(a) \geq 2$ (preserving the informative subtrees closer to the leaves while removing many internal nodes). See Figure 2.

(Miller et al., 2004) cut the BCluster tree at a certain depth k to simplify the tree, meaning every leaf descending from a particular internal node at level k is made an immediate child of that node. They use the tree to get extra features for a discriminative model to tackle the problem of sparsity—the features obtained from the new tree do not suffer from sparsity since each node has several words as its leaves. This technique did not work well for our application so we will not report results using it in our experiments.

5 Experiments

In this section we present experimental results on two IR datasets: Cranfield and Medline[4]. The Cranfield dataset consists of 1,400 documents and 225 queries; its vocabulary size after stemming and removing stop words is 4,227. The Medline dataset contains 1,033 documents and 30 queries with the vocabulary size of 8,800 after stemming and removing stop words. We compare HDT with the flat HDD model and Okapi BM25 (Robertson et al., 1992). Since one of our motivations has been to

[3]Contracting an edge means removing the edge and the adjacent child node and connecting the grandchildren to the parent.

[4]Both datasets can be downloaded from http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections.

Tree	Depth Statistics				Performance			
	Cranfield		Medline		Cranfield		Medline	
	avg / max	total	avg / max	total	avg-pr	top10-pr	avg-pr	top10-pr
BCluster	16.7 / 24	4226	16.4 / 22	8799	0.2675	0.3218	**0.2131**	0.6433
BC contract $\tau \geq 2$	6.2 / 16	3711	5.3 / 14	7473	**0.2685**	0.3147	0.2079	0.6533
BC contract $\tau = 1$	16.1 / 23	3702	15.8 / 22	7672	**0.2685**	0.3204	0.1975	0.6400
DCluster	41.2 / 194	4226	38.1 / 176	8799	0.2552	0.3120	0.1906	0.6300
DC contract $\tau \geq 2$	2.3 / 8	2469	3.3 / 9	5091	0.2555	0.3156	0.1906	0.6167
DC contract $\tau = 1$	40.9 / 194	3648	38.1 / 176	8799	0.2597	0.3129	0.1848	0.6300
PCluster	50.2 / 345	4226	37.1 / 561	8799	0.2613	0.3231	0.1681	0.6633
PC contract $\tau \geq 2$	35.2 / 318	3741	20.4 / 514	7280	0.2624	0.3213	0.1792	0.6767
PC contract $\tau = 1$	33.6 / 345	2246	34.1 / 561	4209	0.2588	**0.3240**	0.1880	0.6633
flat model	1 / 1	1	1 / 1	1	0.2506	0.3089	0.1381	0.6133
BM25	–	–	–	–	0.2566	0.3124	0.1804	0.6567
BM25QueryExp	–	–	–	–	0.2097	0.3191	0.2121	**0.7366**

Table 1: Average precision and Top-10 precision scores of HDT with different trees versus flat model and BM25. The statistics for each tree shows its average/maximum depth of its leaf nodes as well as the number of its total internal nodes. The **bold** numbers highlight the best results in the corresponding columns.

get away from query expansion, we also compare against Okapi BM25 with query expansion. The new terms to expand each query are chosen based on Robertson-Sparck Jones weights (Robertson and Sparck Jones, 1976) from the pseudo relevant documents. The comparison criteria are (i) top-10 precision, and (ii) average precision.

5.1 HDT vs Baselines

All the hierarchical clustering algorithms mentioned in section 4 are used to generate trees, each of which is further post-processed by tree simplification operators described in section 4.1. We consider (i) contracting nodes at higher levels of the hierarchy ($\tau \geq 2$), and (ii) contracting nodes right above the leaves ($\tau = 1$).

The statistics of the trees before and after post-processing are shown in Table 1. Roughly, the Dcluster and BCluster trees do not have long chains with leaves hanging directly off them, which is why their average depths are reduced significantly by the $\tau \geq 2$ simplification, but not by the $\tau = 1$ simplification. The converse is true for Pcluster: the trees have many chains with leaves hanging directly off them, which is why average depth is not reduced as much as the previous trees based on the $\tau \geq 2$ simplification. However the average depth is still reduced significantly compared to the original trees.

Table 1 presents the performance of HDT with different trees against the baselines in terms of the top-10 and average precision (we have bold faced the performance values which are the maximum of each column). HDT with every tree outperforms significantly the flat model in both datasets. More specifically, HDT with (original) BCluster and PCluster trees significantly outperforms the three baselines in terms of both performance measure for the Cranfield. Similar trends are observed on the Medline except here the baseline Okapi BM25 with query expansion is pretty strong[5], which is still outperformed by HDT with BCluster tree.

To further highlight the differences among the methods, we have shown the precision at particular recall points on Medline dataset in Figure 4 for HDT with PCluster tree vs the baselines. As the recall increases, the precision of the PCluster tree significantly outperforms the flat model and BM25. We attribute this to the ability of PCluster tree to give high scores to documents which have words relevant to a query word (an effect similar to query expansion).

5.2 Analysis

It is interesting to contrast the learned α_k's for each of the clustering methods. These α_k's impose cor-

[5]Note that we tuned the parameters of the baselines BM25 with/without query expansion with respect to their performance on the actual retrieval task, which in a sense makes them appear better than they should.

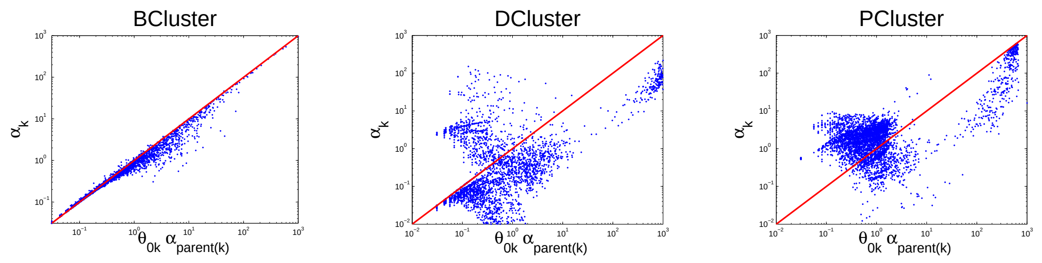

Figure 3: The plots showing the contribution of internal nodes in trees constructed by the three clustering algorithms for the Cranfield dataset. In each plot, a point represent an internal node showing a positive exponent in the node's contribution (i.e. positive correlation among its children) if the point is below $x = y$ line. From left to the right plots, the fraction of nodes below the line is 0.9044, 0.7977, and 0.3344 for a total of 4,226 internal nodes.

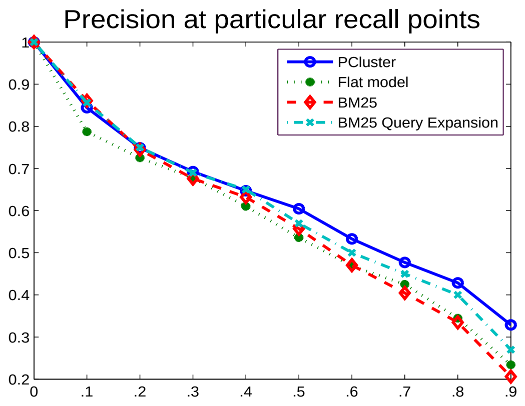

Figure 4: The precision of all methods at particular recall points for the Medline dataset.

relations on the probabilities of the children under k in an interesting fashion. In particular, if we compare α_k to $\theta_{0k}\alpha_{\text{parent}(k)}$, then a larger value of α_k implies that the probabilities of picking one of the children of k (from among all nodes) are positively correlated, while a smaller value of α_k implies negative correlation. Roughly speaking, this is because drawn values of θ_{jl} for $l \in C(k)$ are more likely to be closer to uniform (relative to the flat Dirichlet) thus if we had picked one child of k we will likely pick another child of k.

Figure 3 shows scatter plots of α_k values versus $\theta_{0k}\alpha_{\text{parent}(k)}$ for the internal nodes of the trees. Firstly, smaller values for both tend to be associated with lower levels of the trees, while large values are with higher levels of the trees. Thus we see that PCluster tend to have subtrees of vocabulary terms that are positively correlated with each other—i.e. they tend to co-occur in the same documents. The converse is true of DCluster and BCluster because they tend to put words with the same meaning together, thus to express a particular concept it is enough to select one of the words and not to choose the rest. Figure 5 show some fragments of the actual trees including the words they placed together and α_k parameters learned by HDT model for their internal nodes. Moreover, visual inspection of the trees shows that DCluster can easily misplace words in the tree, which explains its lower performance compared to the other tree construction methods.

Secondly, we observed that for higher nodes of the tree (corresponding generally to larger values of α_k and $\theta_{0k}\alpha_{\text{parent}(k)}$) PCluster α_k's are smaller, thus higher levels of the tree exhibit negative correlation. This is reasonable, since if the subtrees capture positively correlated words, then higher up the tree the different subtrees correspond to clusters of words that do not co-occur together, i.e. negatively correlated.

6 Conclusion and Future Work

We presented a hierarchical Dirichlet tree model for information retrieval which can inject (semantical or syntactical) word relationships as the domain knowledge into a probabilistic model for information retrieval. Using trees to capture word relationships, the model is highly efficient while making use of both prior information about words and their occurrence statistics in the corpus. Furthermore, we investigated the effect of different tree construction algorithms on the model performance.

On the Cranfield dataset, HDT achieves 26.85% for average-precision and 32.40% for top-10 preci-

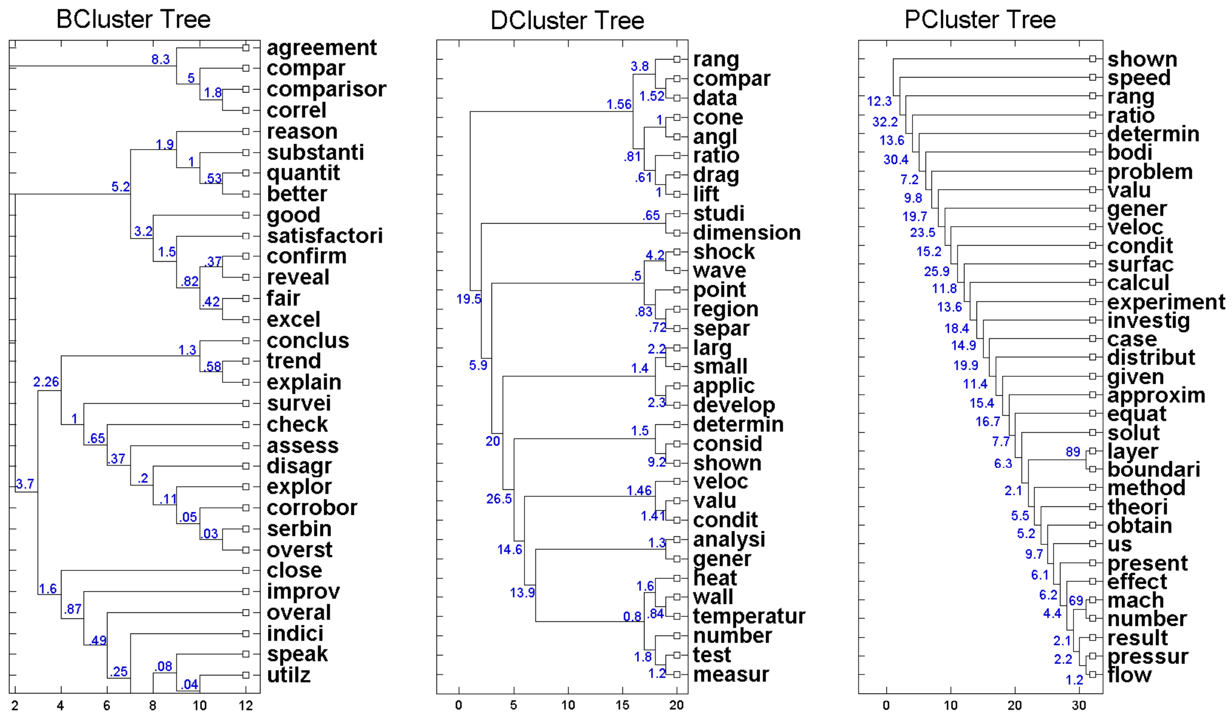

Figure 5: Small parts of the trees learned by clustering algorithms for the Cranfield dataset where the learned α_k for each internal node is written close to it.

sion, and outperforms all baselines including BM25 which gets 25.66% and 31.24% for these two measures. On the Medline dataset, HDT is competitive with BM25 with Query Expansion and outperforms all other baselines. These encouraging results show the benefits of HDT as a principled probabilistic model for information retrieval.

An interesting avenue of research is to construct the vocabulary tree based on WordNet, as a way to inject independent prior knowledge into the model. However WordNet has a low coverage problem, i.e. there are some words in the data which do not exist in it. One solution to this low coverage problem is to combine trees generated by the clustering algorithms mentioned in this paper and WordNet, which we leave as a future work.

References

L. Douglas Baker and Andrew Kachites McCallum. 1998. Distributional clustering of words for text classification. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 96–103.

P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. 1990. Class-based n-gram models of natural language. *Computational Linguistics*.

P. J. Cowans. 2004. Information retrieval using hierarchical dirichlet processes. In *Proceedings of the 27th Annual International Conference on Research and Development in Information Retrieval (SIGIR)*.

R. O. Duda, P. E. Hart, and D. G. Stork. 2000. *Pattern Classification*. Wiley-Interscience Publication.

N. Friedman. 2003. Pcluster: Probabilistic agglomerative clustering of gene expression profiles. Available from http://citeseer.ist.psu.edu/668029.html.

Donald Metzler and W. Bruce Croft. 2007. Latent concept expansion using markov random fields. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*.

S. Miller, J. Guinness, and A. Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proceedings of North American Chapter of the Association for Computational Linguistics - Human Language Technologies conference (NAACL HLT)*.

T. Minka. 2003. The dirichlet-tree distribution. Available from http://research.microsoft.com/minka/papers/dirichlet/minka-dirtree.pdf.

J. Nocedal. 1980. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of english words. In *31st Annual Meeting of the Association for Computational Linguistics*, pages 183–190.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*.

S. E. Robertson and K. Sparck Jones. 1976. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146.

S. E. Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, and M. Lau. 1992. Okapi at trec. In *Text REtrieval Conference*, pages 21–30.

G. Salton and M.J. McGill. 1983. *An Introduction to Modern Information Retrieval*. McGraw-Hill, New York.