Hierarchical Language Models for XML Component Retrieval

Paul Ogilvie and Jamie Callan

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
pto@lti.cs.cmu.edu, callan@lti.cs.cmu.edu

Abstract. Experiments using hierarchical language models for XML component retrieval are presented in this paper. The role of context is investigated through incorporation of the parent's model. We find that context can improve the effectiveness of finding relevant components slightly. Additionally, biasing the results toward long components through the use of component priors improves exhaustivity but harms specificity, so care must be taken to find an appropriate trade-off.

1 Introduction

Language modeling approaches have been applied successfully to retrieval of XML components in previous INEX evaluations [1][2][3][4][5]. In [4] and [5], the authors presented a hierarchical language model for retrieval of XML components. These works proposed that each document component be modeled by a language model estimated using evidence in the node and its children nodes. The work here extends the model to include the parent node's model in estimation, which allows for some context and is called shrinkage.

New experiments using this model are presented that examine the role of shrinkage introduced in this work and the use of the prior probabilities popularized by [3] for the evaluation of Content-Only queries. Our experiments show that shrinkage provides a modest boost in performance. Prior probabilities can have a strong effect in biasing results, particularly in improving exhaustivity (finding all relevant text) while at the same time harming specificity (finding the best component within the hierarchy). A prior based on the square of the length of text contained in a component and its children was found to be most effective.

Section 2 presents the model of documents and Section 3 describes how document components are ranked. Experimental methodology and results are presented in Sections 4 and 5. Related work is discussed in Section 6 and Section 7 concludes the paper.

2 Modeling Documents with Hierarchical Structure

Hierarchically structured documents may be represented as a tree, where nodes in the tree correspond to components of the document. From the content of the document, a generative distribution may be estimated for each node in the tree. The distribution at a node may be estimated using evidence from the text of the node, the children's distributions, or the parent's distribution, and so on.

Representing hierarchically structured documents in this manner is simple and flexible. In this approach we combine the evidence from the various components in the document using the document structure as guidance. This model uses linear interpolation to combine the evidence from the component, its children components, and its parent component. The model below is similar to previous work by the authors [4][5], but is extended to allow for the inclusion of a component's context within the document.

More formally, the hierarchical structure of the document is represented by a tree, each vertex $v \in \mathcal{V}$ in the tree corresponding to a document component. Directed edges in the tree are represented as a list of vertex pairs $(v_i, v_j) \in \mathcal{E}$ when v_i is the parent of v_j. *Parent*, *children* and *descendants* functions may be defined as:

$$parent\,(v_j) = v_i : (v_i, v_j) \in \mathcal{E}$$

$$children\,(v_i) = \{v_j : (v_i, v_j) \in \mathcal{E}\}$$

$$descendants\,(v_i) = \left\{ \begin{array}{l} v_j : v_j \in children\,(v_i) \text{ or} \\ \quad \exists v_k \in children\,(v_i) \\ \quad\quad \text{s.t. } v_j \in descendants\,(v_k) \end{array} \right\}$$

As stated above, the generative model for a component may be estimated using a linear interpolation of the model estimated directly from the component, its children's models, and its parent model. Estimation of the generative models for the components of a document is a three step process. First, a smoothed generative model is θ_{v_i} estimated from the observations of the component in the document v_i that does not include evidence of children components:

$$\begin{aligned} P\,(w\,|\theta_{v_i}) = &\left(1 - \lambda^u_{v_i}\right) P\,(w\,|MLE\,(v_i)) \\ &+\lambda^u_{v_i} P\left(w\,\middle|\theta_{type(v_i)}\right) \end{aligned} \tag{1}$$

This model estimates a distribution directly from observed text within the document component. The $\theta_{type(v_i)}$ model is a collection level background model for smoothing these estimates. The background model is sometimes referred to as a "universal" model, hence the u in λ^u. The *type* function may be used to specify document component specific models, as the language in titles may be different from other text, or it may simply return one language model for all components, which would provide larger amounts of text for the estimation of the corpus model.

The next step is to estimate the intermediate θ'_{v_i} model, from the bottom up to the top of the tree:

$$P\left(w \left| \theta'_{v_i} \right.\right) = \lambda^{c'}_{v_i} P\left(w \left| \theta_{v_i} \right.\right) + \sum_{v_j \in children(v_i)} \lambda^{c}_{v_j} P\left(w \left| \theta'_{v_j} \right.\right), \quad (2)$$

$$1 = \lambda^{c'}_{v_i} + \sum_{v_j \in children(v_i)} \lambda^{c}_{v_j}$$

This model incorporates the evidence from the children nodes. If the λ^c parameters are set proportional to the length of the text in the node, as in

$$\lambda^{c'}_{v_i} = \frac{|v_i|}{|v_i| + \sum_{v_k \in descendants(v_i)} |v_k|}$$
$$\lambda^{c}_{v_j} = \frac{|v_j| + \sum_{v_k \in descendants(v_j)} |v_k|}{|v_i| + \sum_{v_k \in descendants(v_i)} |v_k|} \quad (3)$$

where $|v_i|$ is the length in tokens of the text in node v_i not including tokens in its children, θ'_{v_i} is equivalent to a flat text model estimated from the text in node v_i interpolated with a background model. However, this choice of parameters is not required, and the weight placed on a child node may be dependent on the node's type. For example, the text in title nodes may be more representative than the body of a document, so a higher weight on the title model may improve retrieval performance.

After the estimation of θ'_{v_i}, the θ''_{v_i} models used for ranking the components are estimated from the root of the tree down:

$$P\left(w \left| \theta''_{v_i} \right.\right) = \left(1 - \lambda^{p}_{parent(v_i)}\right) P\left(w \left| \theta'_{v_i} \right.\right) + \lambda^{p}_{parent(v_i)} P\left(w \left| \theta''_{parent(v_i)} \right.\right) \quad (4)$$

Incorporating the parent model in this way allows the context of the component within the document to influence the language model. Incorporating a parent's language model in this way is referred to as shrinkage. In [6], McCallum and Nigam introduced shrinkage to information retrieval in the context of text classification. Classes were modeled hiearchically in this work. Class language models were estimated from the text in the documents assigned to the class, and all ancestor language models in the class hierarchy. The hierarchical model for classes used in [6] is very similar to the model of documents presented in this proposal. The difference in model estimation in this work is the application of shrinkage to document models, rather than class models.

The choice of linear interpolation parameters λ may depend upon the task and corpus. Ideally, the choice of these parameters would be set to maximize some measure of retrieval performance through automated learning.

A set of rankable items, document components that may be returned by the system, $\mathcal{R}$ must also be defined for the document. For the hierarchical model presented here, $\mathcal{R}$ may be any subset of $\mathcal{V}$.

2.1 Example

The estimation process described above may be clarified through describing the process for an example document. The example document is a well known children's poem encoded in XML:

```
<poem id='p1'>
    <title> Little Jack Horner </title>
    <body>
        Little Jack Horner
        Sat in the corner,
        Eating of Christmas pie;
        He put in his thumb
        And pulled out a plumb,
        And cried, <quote> What a
            good boy am I! </quote>
    </body>
</poem>
```

There are four components of this document, corresponding to the poem, title, body, and quote tags. Let us now assign labels to these components v_1 to the poem, v_2 to the title component, v_3 to the body component, and v_4 to the quote. The structure of the document may be drawn, as in Figure 1 or be described as a set of vertices and edges:

$$\begin{aligned}\mathcal{G} &= (\mathcal{V}, \mathcal{E}) \\ \mathcal{V} &= \{v_1, v_2, v_3, v_4\} \\ \mathcal{E} &= \{(v_1, v_2), (v_1, v_3), (v_3, v_4)\}\end{aligned}$$

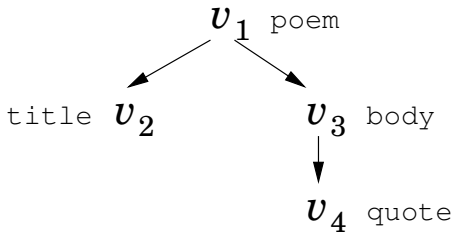

Fig. 1. The tree representing document structure for "Little Jack Horner".

Not all components of the document may be rankable items. A set of rankable items must be defined. In our example, perhaps only the poem and the body of the poem are considered rankable items: $\mathcal{R} = \{v_1, v_3\}$.

The estimation process is illustrated in Figure 2. First, smoothed θ_{v_i} models are estimated for each vertex using the text occurring in the document component corresponding to the vertex. Note that "What a good boy am I!" is not used for the estimation of θ_{v_3}, it is only used in the estimation for the model of v_3's child node v_4:

$$\begin{aligned} P\left(w\left|\theta_{v_i}\right.\right) = \left(1-\lambda_{v_i}^{u}\right) P\left(w\left|MLE\left(v_i\right)\right.\right) \\ +\lambda_{v_i}^{u} P\left(w\left|\theta_{type(v_i)}\right.\right) \end{aligned} \tag{5}$$

Next, the θ'_{v_i} models are estimated by combination of the θ_{v_i} model and the θ' models of v_i's children. For example, θ'_{v_3} is an interpolation of θ_{v_3} and θ'_{v_4}. Similarly, θ'_{v_1} is an interpolation of θ_{v_1}, θ'_{v_2}, and θ'_{v_3}:

$$\begin{aligned} P\left(w\left|\theta'_{v_1}\right.\right) &= \lambda_{v_1}^{c'} P\left(w\left|\theta_{v_1}\right.\right) + \lambda_{v_2}^{c} P\left(w\left|\theta'_{v_2}\right.\right) \\ &\quad +\lambda_{v_3}^{c} P\left(w\left|\theta'_{v_3}\right.\right) \\ P\left(w\left|\theta'_{v_2}\right.\right) &= P\left(w\left|\theta_{v_2}\right.\right) \\ P\left(w\left|\theta'_{v_3}\right.\right) &= \lambda_{v_3}^{c'} P\left(w\left|\theta_{v_3}\right.\right) + \lambda_{v_4}^{c} P\left(w\left|\theta'_{v_4}\right.\right) \\ P\left(w\left|\theta'_{v_4}\right.\right) &= P\left(w\left|\theta_{v_4}\right.\right) \end{aligned} \tag{6}$$

Finally, the θ''_{v_i} models used in ranking are estimated by interpolating the θ'_{v_i} model with the $\theta''_{parent(v_i)}$ model. In the example, θ''_{v_1} is simply taken as θ'_{v_1} as v_1 has no parent. The other vertices do have parents, and θ''_{v_3} is an interpolation of θ'_{v_3} and θ''_{v_1}:

$$\begin{aligned} P\left(w\left|\theta''_{v_1}\right.\right) &= P\left(w\left|\theta'_{v_1}\right.\right) \\ P\left(w\left|\theta''_{v_3}\right.\right) &= \left(1-\lambda_{v_1}^{p}\right) P\left(w\left|\theta'_{v_3}\right.\right) + \lambda_{v_1}^{p} P\left(w\left|\theta''_{v_1}\right.\right) \end{aligned} \tag{7}$$

We can expand the equations for these rankable items to use only θ models as follows:

$$P\left(w\left|\theta''_{v_1}\right.\right) = \lambda^{c'}_{v_1} P\left(w\left|\theta_{v_1}\right.\right) + \lambda^{c}_{v_2} P\left(w\left|\theta_{v_2}\right.\right) + \lambda^{c}_{v_3}\left(\lambda^{c'}_{v_3} P\left(w\left|\theta_{v_3}\right.\right) + \lambda^{c}_{v_4} P\left(w\left|\theta_{v_4}\right.\right)\right)$$

$$\begin{aligned} P\left(w\left|\theta''_{v_3}\right.\right) &= \left(1-\lambda^{p}_{v_1}\right)\left(\lambda^{c'}_{v_3} P\left(w\left|\theta_{v_3}\right.\right) + \lambda^{c}_{v_4} P\left(w\left|\theta_{v_4}\right.\right)\right) \\ &\quad + \lambda^{p}_{v_1} P\left(w\left|\theta''_{v_1}\right.\right) \\ &= \left(1-\lambda^{p}_{v_1}+\lambda^{p}_{v_1}\lambda^{c}_{v_3}\right) \\ &\quad \left(\lambda^{c'}_{v_3} P\left(w\left|\theta_{v_3}\right.\right) + \lambda^{c}_{v_4} P\left(w\left|\theta_{v_4}\right.\right)\right) \\ &\quad + \lambda^{p}_{v_1}\left(\lambda^{c'}_{v_1} P\left(w\left|\theta_{v_1}\right.\right) + \lambda^{c}_{v_2} P\left(w\left|\theta_{v_2}\right.\right)\right) \end{aligned} \tag{8}$$

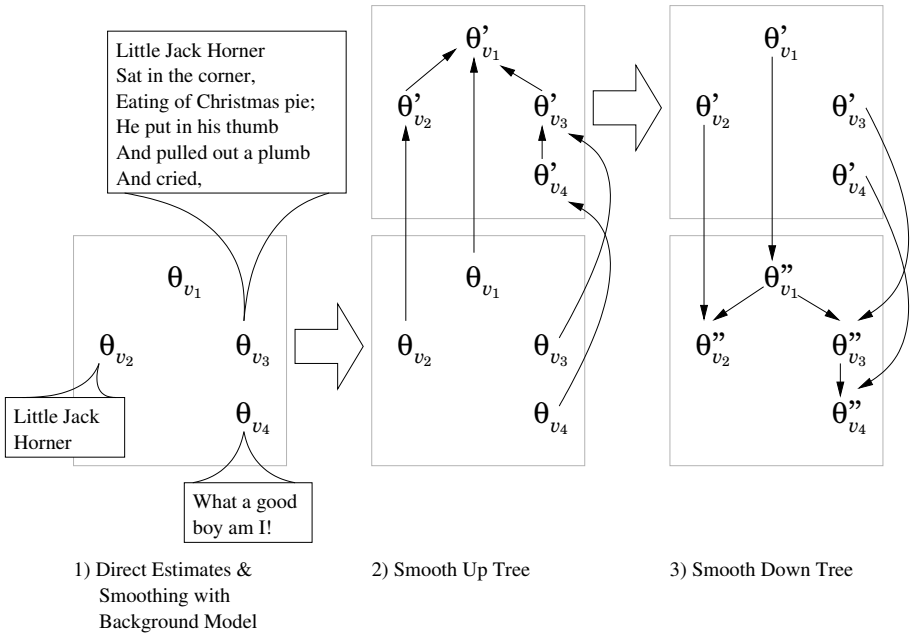

Fig. 2. The estimation process for "Little Jack Horner".

3 Ranking Items for Queries

This section describes how rankable items are ordered for queries. Ordering of items is based on the query-likelihood model, where items are ranked by the probability of generating the query. It is also desirable to provide support for structured queries as well, which will be discussed.

3.1 Queries

Rankable items across documents for flat text queries may simply be ordered by $P\left(Q\left|\theta''_{v_i}\right.\right)$, where

$$P\left(Q\left|\theta''_{v_i}\right.\right) = \prod_{w \in Q} P\left(w\left|\theta''_{v_i}\right.\right)^{tf(w,Q)} \tag{9}$$

This is the natural adaptation of query-likelihood [7][8] [9][10][11] to the model.

There are many cases where it is desirable to place constraints on where the query terms appear in the document structure of a representation. This can be done by constraining which θ_{v_i} distribution generates the terms. For example, consider the NEXI [12] query

```
//poem[about(.//title, Horner)]
```

which requests poem components where the title component is about 'Horner'. The NEXI query language was developed as a simple adaptation of XPath to information retrieval. All example queries in this proposal will be expressed in NEXI. For our example document with a single representation, instead of measuring $P\left(\text{'Horner'}\left|\theta''_{v_1}\right.\right)$, corresponding to the probability the poem component generated the query term 'Horner', $P\left(\text{'Horner'}\left|\theta''_{v_2}\right.\right)$ is used. $P\left(\text{'Horner'}\left|\theta''_{v_2}\right.\right)$ measures the probability that the title component generated 'Horner'.

There are cases where a structural constraint on the query may be applicable to multiple components. Consider the query:

```
//document[about(.//paragraph, plum)]
```

Most documents contain many paragraphs. Which distribution is chosen to generate 'plum'? Many reasonable options are applicable. One approach may create a new distribution estimated as a combination of all of the θ''_{v_i} distributions corresponding to paragraph components. Another approach may take the θ''_{v_i} distribution corresponding to paragraph nodes that maximizes the probabity of generating 'plum', which is the approach taken here.

Constraining the generating distribution in this manner is a strict interpretation of the constraints expressed in the query (as in previous SCAS tasks). The ranking items as described above requires that only poems be returned as results and that they contain titles. Note that through the use of smoothing, 'Horner' is not required to be present in the title. However, if the structural constraints are intended as hints to relevance (as in the VCAS task), then this approach can only return a subset of relevant items. Loose interpretation of structural constraints is something that remains a challenge and will be investigated as a part of future work.

3.2 Priors

Query independent information about relevant documents is not uncommon and may be useful for ranking purposes. For example, there may be a tendency for longer documents to be more likely to be relevant than short documents. This information may be leveraged through the use of priors, which are a belief independent of the query that the document may be relevant. They are incorporated to ranking within the generative framework using Bayes rule:

$$\begin{aligned} & P\left(v_i \text{ is Rel} \,|Q, g\left(v_i\right)=a\right) \\ & \propto P\left(Q \,|v_i \text{ is Rel}, g\left(v_i\right)=a\right) \\ & P\left(v_i \text{ is Rel} \,|g\left(v_i\right)=a\right) \\ & \approx P\left(Q \,|\theta''_{v_i}\right) P\left(v_i \text{ is Rel} \,|g\left(v_i\right)=a\right) \end{aligned} \tag{10}$$

where $g\left(v_i\right)$ is a function of the rankable item such as the length of v_i and $P\left(Q \,|\theta''_{v_i}\right)$ is assumed representative of $P\left(Q \,|v_i \text{ is Rel}, g\left(v_i\right)=a\right)$. Theoretically, the prior probability $P\left(v_i \text{ is Rel} \,|g\left(v_i\right)=a\right)$ can be estimated from training data. However, in practice the prior is not a true prior probability estimate as it is often used to correct a bias in the ranking function at the same time as incorporating the prior belief that v_i is relevant. This makes the choice of how $P\left(v_i \text{ is Rel} \,|g\left(v_i\right)=a\right)$ is estimated somewhat of a fine art, rather than a theoretically driven process.

4 Methodology

All experiments use a local adaptation of the Lemur [13] toolkit for XML retrieval. Two databases were built - one using the Krovetz stemmer [14], and one without stemming. A stopword list of around 400 words was used for both databases. Our prior INEX paper [5] describes most adaptations to index structures and basic retrieval algorithms used presently.

Since then, some query nodes for structured retrieval have been added. We presently only support *AND* clauses, *about* clauses, and path constraints. Numeric constraints are ignored by the retrieval system. *OR* clauses are converted to *AND* clauses, temporarily sidestepping the issue of how the *OR* probabilities are computed. *NOT* clauses are dropped from the query, as are terms with a '-' in front of them. Different clauses of the queries are given equal weight. Phrase constraints are dropped, but the words are kept. A '+' in front of a query term is ignored. Basically, all queries are converted to contain only *AND* clauses with path constraints and about clauses. For example, query 66 is converted from

```
//article[.//fm//yr < 2000]
   //sec[about(., 'search engines')]
```

to

```
//article//sec[about(., search engines)]
```

The graph structures used were taken directly from the XML structure of the document. All components were considered rankable items. The weight placed on the collection model λ^u is 0.2 and when using shrinkage, λ^p is set to 0.1. A single background model estimated from all text in the collection was used. Estimation of θ' models use $\lambda^{c'}$ and λ^c set according to Equation 3. These parameters were chosen by experimentation on the INEX 2003 topics.

The prior probabilities used in experiments are all based on the aggregated length of a component may take the following form:

- *linear* – $P\left(v_i \text{ is Rel} \,|\, length\left(v_i\right) = x\right) \propto x$
- *square* – $P\left(v_i \text{ is Rel} \,|\, length\left(v_i\right) = x\right) \propto x^2$
- *cubic* – $P\left(v_i \text{ is Rel} \,|\, length\left(v_i\right) = x\right) \propto x^3$

where

$$length\left(v_i\right) = |v_i| + \sum_{v_k \in descendants(v_i)} |v_k| \,. \tag{11}$$

5 Experiments

This section describes some experiments with variations of the system. The discussion in this section centers on the content-only topics. Figure 3 examines the effects of prior probabilities on the strict measure and the specificity oriented measure for content-only topics. The runs in this figure used the Krovetz stemmer and a shrinkage parameter $\lambda^p = 0.1$. The use of more extreme length priors generally resulted in noticeable improvements to the strict measure but at a sacrifice to the specificity oriented measure. Results for more configurations and measures are presented in Table 5. The trends in Figure 3 are confirmed in the table. The more extreme the prior, the higher the exhaustivity and the lower the specificity. Using a more extreme prior also reduced overlap in the result lists, as these runs had distinct biases toward long components. For the rest of the discussion in this section, a linear prior was chosen as a good trade-off between improved exhaustivity and harmed specificity.

Next, some experiments using different shrinkage parameters are explored. For these runs, only the system using the Krovetz stemmer and a linear prior was explored. Figure 4 demonstrates that small values of λ^p boost precision modestly for both strict and specificity oriented measures. A non-zero shrinkage parameter did seem to help, and using too large a parameter hurt precision at low recall. The bottom half of Table 5 contains more evaluation measures and parameter settings. Small settings for the shrinkage parameter can improve performance across all measures, but these results may not be significant.

This section concludes with a brief discussion of the content-and-structure runs. Our original submission had a bug which allowed a component to be returned in the result list multiple times (with different supporting components).

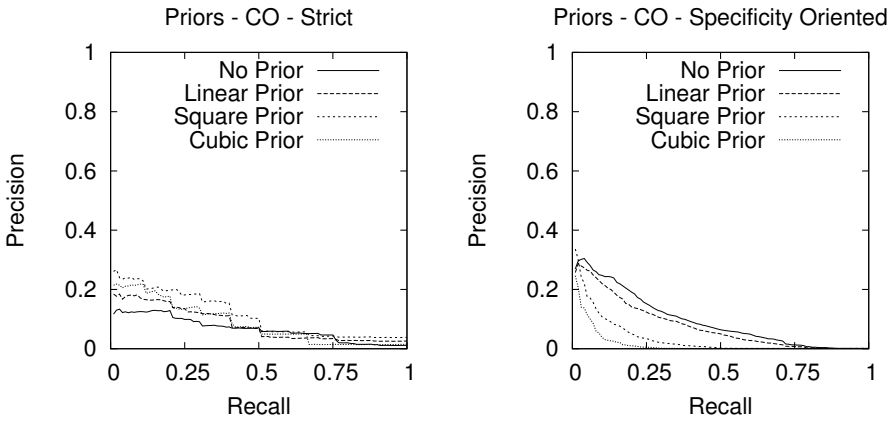

Fig. 3. More extreme length priors can greatly improve performance under strict evaluation, but at a sacrifice to specificity oriented evaluation.

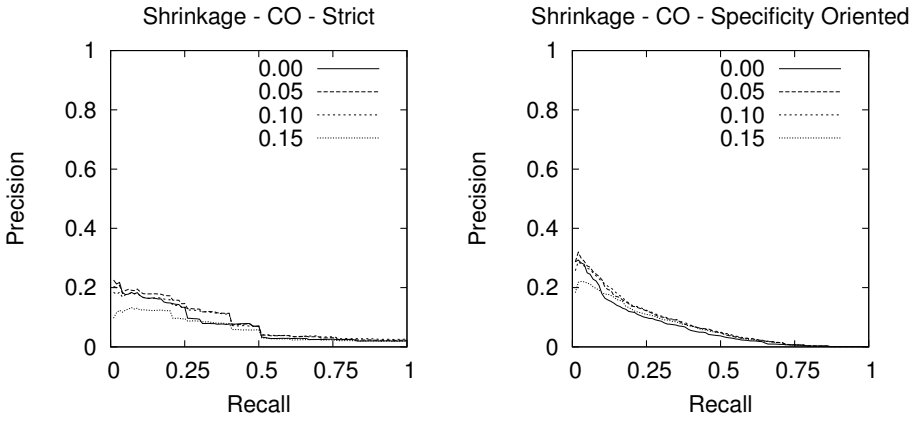

Fig. 4. Very small shrinkage parameter values boost precision moderately at mid-recall ranges for strict evaluation and across the range for specificity oriented evaluation.

Table 1. Run performance for Content-Only topics.

Official	λ^p	Stemmer	Prior	Strict	Generalized	SO	s3 e321	e3 s321
YES[1]	0.0	-	-	0.0640	0.0728	0.0655	0.0541	0.0806
NO	0.0	-	linear	0.0896	0.0770	0.0650	0.0564	0.1234
NO	0.0	-	square	0.1224	0.0448	0.0318	0.0236	0.1864
NO	0.0	-	cubic	0.0902	0.0226	0.0167	0.0130	0.1367
YES[2]	0.1	-	-	0.0675	0.0908	0.0897	0.0771	0.0769
NO	0.1	-	linear	0.0688	0.0829	0.0721	0.0640	0.1055
NO	0.1	-	square	0.1268	0.0480	0.0344	0.0262	0.1885
NO	0.1	-	cubic	0.0927	0.0230	0.0173	0.0139	0.1380
YES[3]	0.1	Krovetz	-	0.0667	0.0947	0.0941	0.0835	0.0776
NO	0.1	Krovetz	linear	0.0817	0.0882	0.0770	0.0663	0.1191
NO	0.1	Krovetz	square	0.1129	0.0445	0.0330	0.0252	0.1721
NO	0.1	Krovetz	cubic	0.0859	0.0200	0.0151	0.0117	0.1324
NO	0.000	Krovetz	linear	0.0745	0.0771	0.0651	0.0561	0.1173
NO	0.025	Krovetz	linear	0.0874	0.0867	0.0748	0.0632	0.1315
NO	0.050	Krovetz	linear	0.0849	0.0885	0.0765	0.0654	0.1315
NO	0.075	Krovetz	linear	0.0830	0.0889	0.0775	0.0667	0.1255
NO	0.100	Krovetz	linear	0.0817	0.0882	0.0770	0.0663	0.1191
NO	0.125	Krovetz	linear	0.0682	0.0840	0.0734	0.0632	0.1076
NO	0.150	Krovetz	linear	0.0591	0.0761	0.0670	0.0573	0.0915

[1]Lemur_CO_NoStem_Mix02 [2]Lemur_CO_NoStem_Mix02_Shrink01
[3]Lemur_CO_KStem_Mix02_Shrink01

Official	λ^p	Stemmer	Prior	Overlap	Aggregate
YES[1]	0.0	-	-	66.7	0.0651
NO	0.0	-	linear	72.6	0.0809
NO	0.0	-	square	47.1	0.0881
NO	0.0	-	cubic	41.7	0.0629
YES[2]	0.1	-	-	74.8	0.0774
NO	0.1	-	linear	73.4	0.0772
NO	0.1	-	square	46.3	0.0910
NO	0.1	-	cubic	40.5	0.0641
YES[3]	0.1	Krovetz	-	73.0	0.0807
NO	0.1	Krovetz	linear	72.6	0.0853
NO	0.1	Krovetz	square	46.2	0.0861
NO	0.1	Krovetz	cubic	40.4	0.0625
NO	0.000	Krovetz	linear	72.4	0.0764
NO	0.025	Krovetz	linear	72.1	0.0877
NO	0.050	Krovetz	linear	72.3	0.0881
NO	0.075	Krovetz	linear	72.5	0.0869
NO	0.100	Krovetz	linear	72.6	0.0853
NO	0.125	Krovetz	linear	72.8	0.0781
NO	0.150	Krovetz	linear	73.0	0.0685

[1]Lemur_CO_NoStem_Mix02
[2]Lemur_CO_NoStem_Mix02_Shrink01
[3]Lemur_CO_KStem_Mix02_Shrink01

This bug has been fixed, and a comparison of the official runs and the bug-fixes are in Table 5. The runs using query structure took a strict interpretation of constraints and as such, it is not surprising that they did poorly for the VCAS task. Our best performing run did not use any structure in the query. All non-structural constraints were removed from the query so that keywords present in the about clauses remained. This was then treated as a flat text query and run using the configuration for CO topics. The use of prior probabilities was investigated, and it was found that the trade-off between exhaustivity and specificity observed for CO held for VCAS as well. However, there does seem to be a preference for shorter components in the VCAS task, as the square prior hurt performance across the board, while the linear prior improved performance.

Table 2. Run performance for Content-and-Structure topics.

Official	Struct-ure	λ^p	Prior	Strict	General-ized	SO	s3 e321	e3 s321	Overlap	Aggregate
YES[1]	NO	0.1	-	0.0710	0.0746	0.0759	0.0834	0.0700	74.0	0.0759
NO	NO	0.1	Linear	0.0889	0.0847	0.0804	0.0819	0.0949	69.9	0.0864
NO	NO	0.1	Square	0.0585	0.0468	0.0377	0.0359	0.0836	48.3	0.0217
YES[2]	YES	0.0	-	0.0468	0.0180	0.0166	0.0253	0.0419	2.4	0.0325
NO (fix)	YES	0.0	-	0.0616	0.0276	0.0309	0.0409	0.0413	4.6	0.0459
YES[3]	YES	0.1	-	0.0466	0.0199	0.0177	0.0249	0.0457	2.4	0.0339
NO (fix)	YES	0.1	-	0.0621	0.0274	0.0302	0.0409	0.0418	4.8	0.0460

[1]Lemur_CAS_as_CO_NoStem_Mix02_Shrink01 [2]Lemur_CAS_NoStem_Mix02
[3]Lemur_CAS_NoStem_Mix02_Shrink01

6 Related Work

Much of the current work in XML component retrieval can be found in these proceedings and in [15][16], so only highly related works will be discussed here.

The Tijah system [1] also uses generative models for retrieval of XML components. They do not explicitly model the hierarchical relationship between components when estimating language models. Instead, they estimate a language model for each component using the text present in it and its children. This is equivalent to our model when $\lambda^p = 0$ and $\lambda^{c'}$ and λ^c are set according to Equation 3. They also incorporate prior probabilities using a log-normal and a linear component length prior. To provide context in the scoring of a component, they average the component score with the document score. For structured queries, constraints on structure are processed similarly. However, for *OR* clauses, the maximum of the scores is taken, while the minimum is taken for *AND* clauses. A system configuration for vague evaluation of structured queries is realized using query rewrites.

Kamps, Sigurbjörnsson, and de Rijke [3] [2] also work within the language modeling framework. Like [1], they do not explicitly model hierarchical relationship between document components when estimating the language model for a component. Rather than estimating the background model using a maximum likelihood estimator, they use an estimate based on element frequencies. They present experiments using a linear, square, and cubic component length prior, and also experiment with a cut-off filtering out short components. As with [1], their model is comparable to our model when $\lambda^p = 0$ and $\lambda^{c'}$ and λ^c are set according to Equation 3. For processing structured queries, [2] describes an approach that combines query rewrites with strict interpretation of the query.

7 Conclusions

This paper described experiments using hierarchical language models for modeling and ranking of XML document components. It extended previous work to incorporate context through the use of shrinkage, which helps modestly for flat text queries. A very small choice for the shrinkage parameter was found to be best for retrieval. This paper also presented experiments using length based priors. A prior probability proportional to the length of the component was found to be most effective across a number of measures.

For vague content and structure queries, where structure is intended only as a hint to the retrieval system, we found that ignoring structure in the query was better than taking a strict interpretation of the structural constraints. This is much like using a flat text query. As with the content only queries, a linear length prior was found to improve performance, but the vague content and structure queries may have a preference for shorter components than the content only queries on average.

Future experiments will examine use of the structural constraints in the content and structure queries as hints for relevance within the framework. More experimentation with how the shrinkage parameter is set will be performed, as well as different approaches to setting the interpolation parameters for the combination of evidence from child nodes.

8 Acknowledgments

This research was sponsored by National Science Foundation (NSF) grant no. CCR-0122581. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implicit, of the NSF or the US government.

References

1. List, J., Mihajlovic, V., Ramirez, G., Hiemstra, D.: The tijah xml-ir system at inex 2003. In: INEX 2003 Workshop Proceedings. (2003) 102–109

2. Sigurbjörnsson, B., Kamps, J., de Rijke, M.: Processing content-and-structure queries for xml retrieval. In: Proccedings of the First Twente Data Management Workshop. (2004) 31–38
3. Kamps, J., de Rijke, M., Sigurbjörnsson, B.: Length normalization in xml retrieval. In: Proceedings of the Twenty-Seventh Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. (2004) 80–87
4. Ogilvie, P., Callan, J.: Language models and structured document retrieval. In: Proceedings of the First Workshop of the INitiative for the Evaluation of XML Retrieval (INEX). (2003)
5. Ogilvie, P., Callan, J.P.: Using language models for flat text queries in xml retrieval. In: Proc. of the Second Annual Workshop of the Initiative for the Evaluation of XML retrieval (INEX), Dagstuhl, Germany (2003)
6. McCallum, A., Nigam, K.: Text classification by bootstrapping with keywords, em and shrinkage. In: Proceedings of the ACL 99 Workshop for Unsupervised Learning in Natural Language Processing. (1999) 52–58
7. Ponte, J., Croft, W.: A language modeling approach for information retrieval. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press (1998) 275–281
8. Hiemstra, D.: Using language models for information retrieval. PhD thesis, University of Twente (2001)
9. Zhai, C., Lafferty, J.: A study of smoothing methods for language models applied to information retrieval. ACM Transactions on Information Systems **2** (2004)
10. Song, F., Croft, W.: A general language model for information retrieval. In: Proceedings of the Eighth International Conference on Information and Knowledge Management. (1999)
11. Westerveld, T., Kraaij, W., Hiemstra, D.: Retrieving web pages using content, links, URLs, and anchors. In: The Tenth Text REtrieval Conf. (TREC-10), NIST SP 500-250. (2002) 663–672
12. Trotman, A., Sigurbjörnsson, B.: Narrow Extended XPath I. Technical report (2004) Available at http://inex.is.informatik.uni-duisburg.de:2004/.
13. Lemur: The Lemur Toolkit for Language Modeling and Information Retrieval. (http://lemurproject.org/)
14. Krovetz, R.: Viewing morphology as an inference process. In: Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM (1993) 191–202
15. Fuhr, N., Goevert, N., Kazai, G., Lalmas, M., eds.: Proceedings of the First Workshop of the INitiative for the Evaluation of XML Retrieval (INEX), ERCIM (2003)
16. Fuhr, N., Maalik, S., Lalmas, M., eds.: Proc. of the Second Annual Workshop of the Initiative for the Evaluation of XML retrieval (INEX), Dagstuhl, Germany (2003)