Estimating Structural Relevance of XML Elements Through Language Model

Cyril Laitang
IRIT
118 route de Narbonne
31062, Toulouse
laitang@irit.fr

Karen Pinel-Sauvagnat
IRIT
118 route de Narbonne
31062, Toulouse
sauvagnat@irit.fr

Mohand Boughanem
IRIT
118 route de Narbonne
31062, Toulouse
bougha@irit.fr

ABSTRACT

Language modeling approaches have been extensively used as an effective way of measuring ad-hoc document content relevance. However, in structured information retrieval (SIR) there is to our knowledge no approach which aims at assessing structural relevance using language models. In this paper we present a language model based on document-query structure likelihood. As the effectiveness of language modeling relies on the associated smoothing technique we experimented two of these techniques. Experiments are conducted on the INEX 2010 Datacentric test set and show the interest of our method compared to official participants to the task.

Categories and Subject Descriptors

[**Structuring Unstructured Datas**]

General Terms

XML retrieval, Language model

1. INTRODUCTION

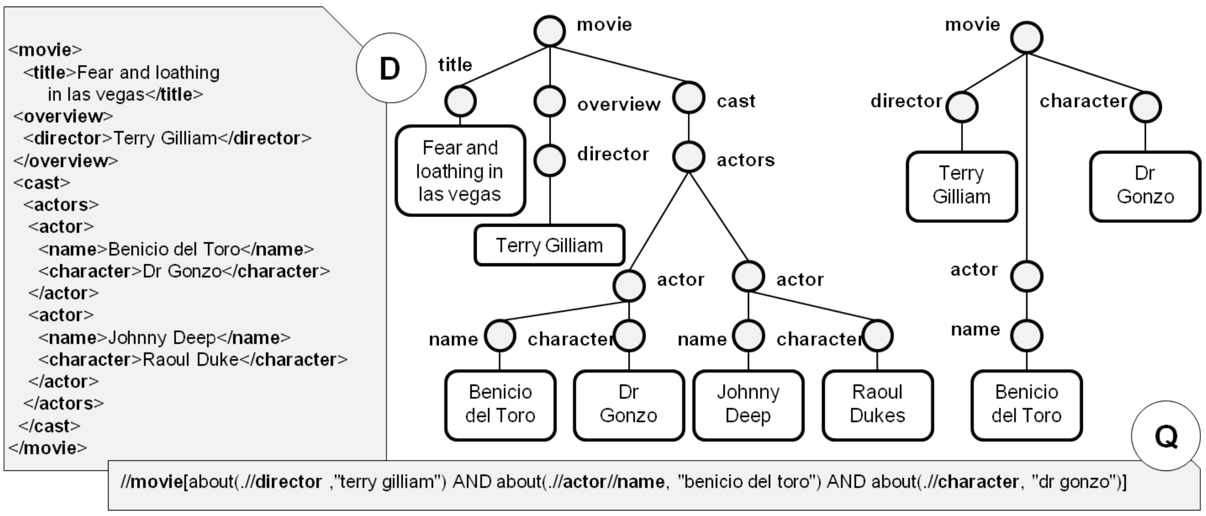
XML documents are semi-structured documents which organize text through semantically meaningful elements labeled with tags. Structural information of XML documents is exploited by Information Retrieval Systems (IRS) to return to users the most exhaustive and specific documents parts (i.e. XML elements, also called nodes) answering to their needs. These needs can be expressed through Content and Structure queries (CAS) which contain both keywords and structural information on the location of the needed text content.
Figure 1 illustrates the structural organization of an XML document and a CAS query.

Most of the retrieval models used for structuerd retrieval are adaptation of traditional retrieval models including the vector space model [3], the probabilistic model [1] or language models [7] [16] [22]. Although the latter have been shown to be effective to evaluate content relevance, to the best of our knowledge they have not been used to assess the document-query structure likelihood. This is what we propose in this paper.

The rest of the paper is organized as follows: section 2 presents language models approaches applied to Structured Information Retrieval. Section 3 presents our language model for structure. Finally, in section 4, some experiments are conducted on a strongly structured test set provided by the INEX evaluation campaign.

2. USING LANGUAGE MODELS IN STRUCTURED INFORMATION RETRIEVAL

Since language models (LM) have shown good performances in Information Retrieval (IR), these approaches have been naturally extended to Structured Information Retrieval (SIR) to model content. As XML elements usually contain fewer terms than documents processed by traditional IRS, most of the proposed approaches use the document context to assess the relevance of an element.
Ogilvie et al. [15] [16] proposed to model each element by a language model estimated using evidence from the content of the element node itself, from its children and its parent nodes. This approach has been then extended in [17] with a generative model to estimate the parameter of the linear interpolation using both positive and negative example extracted from relevant documents. Debasis et al. [4] also extended Ogilvie et al. approach [16] by scoring XML elements though a combined probability of generating the given query from its content and the top level element.
Zhao et al. [22] proposed a generative model where structure is used as an hidden parameter during the element and document model generation process. Huang [6] computed the likelihood of an element based on the length of its path in its document, a shorter path being considered as more relevant than a longer one.
In Hiemstra et al. [7] one can find some of the insight behind the mixture models which aim at using a combination between the likelihood of generating a query at an element level and its combination between this likelihood over the element ascendant and descendant.
Li et al. [18] exploited cross entropy scores between the query model and the document model. More specifically authors split the content conditions constraints and combined them to evaluate the final element score.



```
<movie>
  <title>Fear and loathing
     in las vegas</title>
 <overview>
  <director>Terry Gilliam</director>
 </overview>
 <cast>
  <actors>
   <actor>
    <name>Benicio del Toro</name>
    <character>Dr Gonzo</character>
   </actor>
   <actor>
    <name>Johnny Deep</name>
    <character>Raoul Duke</character>
   </actor>
  </actors>
 </cast>
</movie>
```

```
//movie[about(.//director ,"terry gilliam") AND about(.//actor//name, "benicio del toro") AND about(.//character, "dr gonzo")]
```

Figure 1: Tree representation of an XML document and a query in which *we want a "movie" directed by "Terry Gilliam"' with the actor "Benicio del Toro" and with a character named "Dr Gonzo"*.

Most of the aforementioned approaches show significant scores improvement over traditional baselines such as BM25. However, so far, the documents and queries structures have only been used to process the score propagation or as an element length normalization process.

Approaches can be classified as done in [19] [12] into aggregation or contextualization approaches. Ogilvie et al. [16] [16] and Debasis et al. [4] are indeed aggregation approaches while Hiemstra et al. work [7] can be classified as a contextualization technique.

Structural constraints of queries, when they are taken into account, are considered as a tag-equivalence problem [4] or are processed using path heuristics based on the distance between nodes [6].

Thus, the likelihood is always computed based solely on the content. The structure is used only as a support to build the element model or to boost the final relevance score based on the environment. It should be considered as a hidden parameter added to the content based LM element-query likelihood.
The question is therefore: can language models be extended to assess structure relevance?

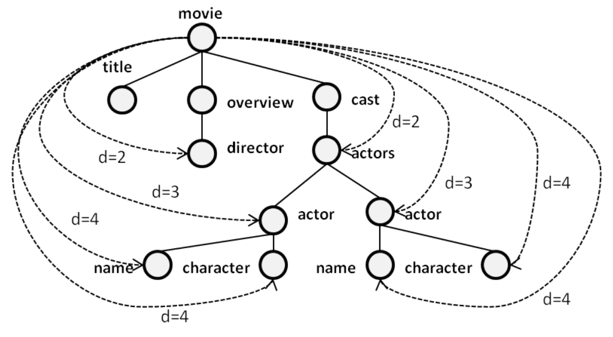
3. A LANGUAGE MODEL FOR DOCUMENT-QUERY STRUCTURE MATCHING

Language models in Information Retrieval aim at ranking documents based on the likelihood of a query being generated by a document model. Most of them are estimated from document content, often terms. We propose a similar approach except that instead of considering terms we consider edges of XML tree structures. We assume that document trees are generated from a given language model. Our aim is then to rank trees T based on their probability to generate a query Q composed of structure conditions.

3.1 Tree Representation

We consider an XML tree $T = \{e_{i,j}\}$ as a set of weighted edges $e_{i,j}$, where i and j are tags of the tree. Similarly a query $Q = \{e_{i,j}\}$ is considered as a set of weighted edges representative of the structural constraints.

Several paths in the tree may be represented by the same edge $e_{i,j}$ since they own the same tags. These paths are of the form $u//v$ (XPath-like form), where u and v are two nodes in N^T the set of nodes in tree T, and where $tag(u) = i$ and $tag(v) = j$, with tag(x) a function returning the label of node x.
Each of those paths is seen as one occurrence of $e_{i,j}$, and is denoted by $e_{i,j}^{u \to v}$.



T={ $e_{movie,title}$, $e_{movie,overview}$, $e_{movie,director}$, $e_{movie,cast}$, $e_{movie,actors}$, $e_{movie,actor}$, $e_{movie,name}$, $e_{movie, character}$, $e_{overview,director}$, $e_{cast,actors}$, $e_{cast,actor}$, $e_{cast,name}$, $e_{cast, character}$, $e_{actors,actor}$, $e_{actors,name}$, $e_{actors, character}$, $e_{actor,name}$, $e_{actor, character}$ }

Figure 2: Tree vocabulary

Figures 2 and 3 respectively illustrate an example of an XML tree T and a query Q. T is composed of 8 parent-child edges and 10 additional edges representing ancestor-descendants relations. We notice that edge $e_{movie,actor}$ has two occurrences which correspond to the following Xpaths: $/movie//actor[1]$ and $/movie//actor[2]$.

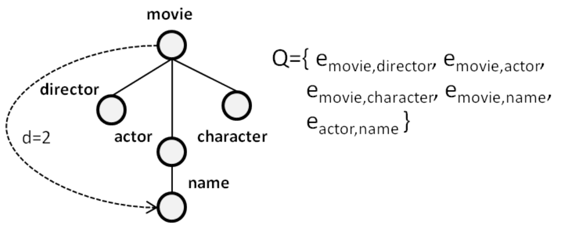

movie
director
actor
character
name
d=2
Q={ $e_{movie,director}$, $e_{movie,actor}$, $e_{movie,character}$, $e_{movie,name}$, $e_{actor,name}$ }

Figure 3: Query vocabulary

In practice, to form the tree and query vocabulary, the same process will be repeated until the whole ancestor-descendants edges are extracted.

Each edge is associated a weight which corresponds to the sum of the weights of its corresponding occurrences in T.

$$w(e_{i,j}, T) = \sum_{\substack{u,v \in N^T / tag(u)=i \\ and\ tag(v)=j}} w(e_{i,j}^{u \to v}) \tag{1}$$

with

$$w(e_{i,j}^{u \to v}) = exp(1 - d(u,v)) \tag{2}$$

d is the distance separing u and v in the tree. This formula, with $w(e_{i,j}^{u \to v}) \in [0,1]$, aims at giving more importance to edges formed by closer nodes, as we believe them to be more relevant.

Based on our previous example in figure 2, the complete weighted edges are shown in table 1. For instance, $w(e_{movie,actor}, T) = exp(-2) + exp(-2)$.

The overall weight of a tree structure is then evaluated as the sum of the tree's weighted edges:

$$w(T) = \sum_{e_{i,j} \in T} w(e_{i,j}, T) \tag{3}$$

3.2 Query Likelihood

We assume, by analogy with LM based on evidence of keywords, that the XML tree structure is generated by a language model M_T. The query is assumed to be a sample of edges generated from a tree language model. Trees are then ranked based on the likelihood of generating the query using the corresponding tree model.

In order to evaluate the structural proximity $RSV(Q, M_T)$ between a query Q and a tree model M_T, we compute the probability $P(Q \mid M_T)$ of the tree to generate the query. The underlying idea is to evaluate the probability of each edge independently as unigrams. We thus have:

$$P(Q \mid M_T) \quad = \quad \prod_{e_{i,j} \in Q} P(e_{i,j} \mid M_T)^{w(e_{ij}, Q)} \tag{4}$$

As our language model approach ranks the XML trees based on the probability of occurrence of a query edges $e_{i,j} \in T$ and as missing edges will assigned a zero probability, a missing query edge could alone discard the document. In order to overcome this data sparseness issue it has been common practice to use smoothing techniques [21] which mainly use a reference language model to assign non-zero probability to events unseen in the training data [8].

We experimented two different smoothing techniques namely *Jelinek-Mercer* [9] and *Dirichlet* [13].

3.3 Jelinek-Mercer

This model involves a linear interpolation of the maximum likelihood of the tree model with the collection, using a coefficient α between $[0,1]$ to control the influence of each. We smooth our tree model M_T based on all $T \in C$.

$$\begin{aligned} P(e_{ij} \mid M_T) &= \alpha \times P(e_{i,j} \mid M_T) \\ &\quad +(1-\alpha) \times P(e_{i,j} \mid M_C) \\ &= \alpha \times P(e_{i,j} \mid T) \\ &\quad +(1-\alpha) \times P(e_{i,j} \mid C) \end{aligned} \tag{5}$$

Alternatively, when T is a tree representing a document part (i.e. when T is not a document tree), we can smooth based on the language model of the document D containing T. The intuition is that all the subtrees in a document should inherit from their parent document model.

$$\begin{aligned} p(e_{ij} \mid M_T) &= \beta \times P(e_{i,j} \mid M_T) + (1-\beta) \times [\\ &\quad \alpha \times P(e_{i,j} \mid M_D) + (1-\alpha) \times P(e_{i,j} \mid M_C)] \\ &= \beta \times P(e_{i,j} \mid T) + (1-\beta) \times [\\ &\quad \alpha \times P(e_{i,j} \mid D) + (1-\alpha) \times P(e_{i,j} \mid C)] \end{aligned} \tag{6}$$

The probability function of an edge in the document tree is computed as :

$$p(e_{i,j} \mid T) = \frac{w(e_{i,j}, T)}{w(T)} \tag{7}$$

With $w(e_{i,j}, T)$ evaluated according to equation 1 and $w(T)$ evaluated according to equation 3.

The probability of an edge $e_{i,j}$ in the collection is computed as :

$$p(e_{i,j} \mid C) = \frac{w(e_{i,j}, C)}{w(C)} \tag{8}$$

With $w(e_{i,j}, C) = \sum_{\substack{u,v \in N^C / tag(u)=i \\ and\ tag(v)=j}} w(e_{i,j}^{u \to v})$, i.e. the weight of edge $e_{i,j}$ in collection C is the sum of the weights of all its occurrences $e_{i,j}^{u \to v}$ in the collection. $w(e_{i,j}^{u \to v})$ is evaluated according to equation 2. $w(C)$ is the overall collection weight, i.e. the sum of the weights of all edges in the collection.

Similarly, the probability of an edge $e_{i,j}$ in the document tree is :

$$p(e_{i,j} \mid D) = \frac{w(e_{i,j}, D)}{w(D)} \tag{9}$$

3.4 Dirichlet priors

Dirichlet prior smoothing is a linear interpolated smoothing based on document's length. The probability of a query structure of being generated in the document tree will then be computed as the product of the probabilities from e in Q

Tags	title	overview	director	cast	actors	actor	name	character
movie	$exp(0)$	$exp(0)$	$exp(-1)$	$exp(0)$	$exp(-1)$	2*$exp(-2)$	2*$exp(-3)$	2*$exp(-3)$
overview	0	0	$exp(0)$	0	0	0	0	0
cast	0	0	0	0	$exp(0)$	2*$exp(-1)$	2*$exp(-2)$	2*$exp(-2)$
actors	0	0	0	0	0	2*$exp(0)$	2*$exp(-1)$	2*$exp(-1)$
actor	0	0	0	0	0	0	2*$exp(0)$	2*$exp(0)$

Table 1: Tree vocabulary: weight of the edges

in the document tree smoothed by the weight of these edges in the collection with μ the smoothing parameter.

$$\begin{aligned} p(e_{i,j} \mid M_T) &= \frac{w(e_{i,j},T) + \mu P(e_{i,j} \mid M_C)}{w(T)+\mu} \\ &= \frac{w(e_{i,j},T) + \mu P(e_{i,j} \mid C)}{w(T)+\mu} \end{aligned} \tag{10}$$

With $w(e_{i,j},T)$ from equation 2, $P(e_{i,j} \mid C)$ from equation 8 and w(T) from equation 3.

Similarly to equation 6, when T is a document subtree we can alternatively evaluate the probability smoothed by the document :

$$\begin{aligned} p(e_{i,j} \mid M_T) &= \frac{w(e_{i,j},T) + \mu \frac{(w(e_{i,j},D) + \lambda P(e_{i,j} \mid M_C)}{w(D)+\lambda}}{w(T)+\mu} \\ &= \frac{w(e_{i,j},T) + \mu \frac{(w(e_{i,j},D) + \lambda P(e_{i,j} \mid C)}{w(D)+\lambda}}{w(T)+\mu} \end{aligned} \tag{11}$$

4. EXPERIMENTS AND RESULTS

In order to evaluate our model, we conducted experiments on the Datacentric 2010 collection of the *Initiative for the Evaluation of XML Retrieval* (INEX) campaign which is the reference evaluation campaign for SIR models. We choose this test set for two main reasons :

- The document collection contains strongly structured documents with semantically relevant elements labels. Following [14] it maked it data-oriented which means that structure is relevant on its own.
- The 2010 associated measures aims at investigating techniques for finding information using queries considering content and structure[1].

4.1 INEX 2010 Datacentric track

The track uses the IMDB data collection generated from IMDB web site. In total, the data collection contains 4,418,081 XML files, including 1,594,513 movies, 1,872,471 actors, 129,137 directors who did not act in any movie, 178,117 producers who did not direct nor act in any movie, and 643,843 other people involved in movies who did not produce nor direct nor act in any movie.
28 content and structure queries are used in the track and the models relevance were evaluated though two measures [20] detailed in [10] :

[1]We did not used Datacentric 2011 ad-hoc track as relevance assessments were done over the whole documents and not elements.

Run name	Equation	Smoothing parameter
Jelinek T	5	$\alpha = 0.4$
Dirichlet T	10	$\mu = 2000$
Jelinek TD	6	$\alpha = 0.3, \beta = 0.3$
Dirichlet TD	11	$\mu = 2000, \lambda = 2000$

Table 2: Smoothing parameters used for experimentation

- *MAiP* (Mean average interpolated precision) which is computed through 101 standard recall points (eg : 0.00, 0.01, etc..)
- *MAgp T2I* which assess the exhaustivity of the returned results. Element score is the score at a tolerance to irrelevance (T2I) points, 300 in our case with no overlap.

4.2 Query evaluation

To evaluate our model on the DataCentric track of INEX, it is necessary to take into account content relevance of XML trees (as queries are composed of content and structure conditions). We thus evaluate the relevance RSV of a tree T regarding a query as follows:

$$RSV(Q,T) = \lambda \times c(T) + (1-\lambda) \times s(T) \tag{12}$$

Where :

- $c(T)$ is the content score of T evaluated by propagating content relevance of leaf nodes to the root node of T. The leaf nodes score is computed using a tf-idf formula (see [11] for more details).
- $s(T)$ is evaluated with $p(Q \mid M_T)$ using Jelinek-Mercer smoothing (equation 5 or 6) or Dirichlet prior (equation 10 or 11).

Equation 12 is only evaluated for some particular trees of the collection:

- As trees in the collection may contain many branches and thus many edges, and as this would increase unnecessarily the vocabulary of our language model, we decided to prune them. For this purpose, we concatenate what we call interesting paths. An interesting path starts from a relevant element according to content conditions of the query and sharing a label from the query and ends with an element sharing a label with the query root (see [11] for more details). Following our previous example, figure 4 illustrates the pruning of a tree trough the selection of interesting paths.
- Moreover, RSV(Q, T) is not evaluated for all pruned $T \in C$. We only keep trees potentially relevant for the query, i.e. trees for which c(T)>0.

4.3 Results

After various experimentations we fixed the smoothing parameters values as shown in table 2.

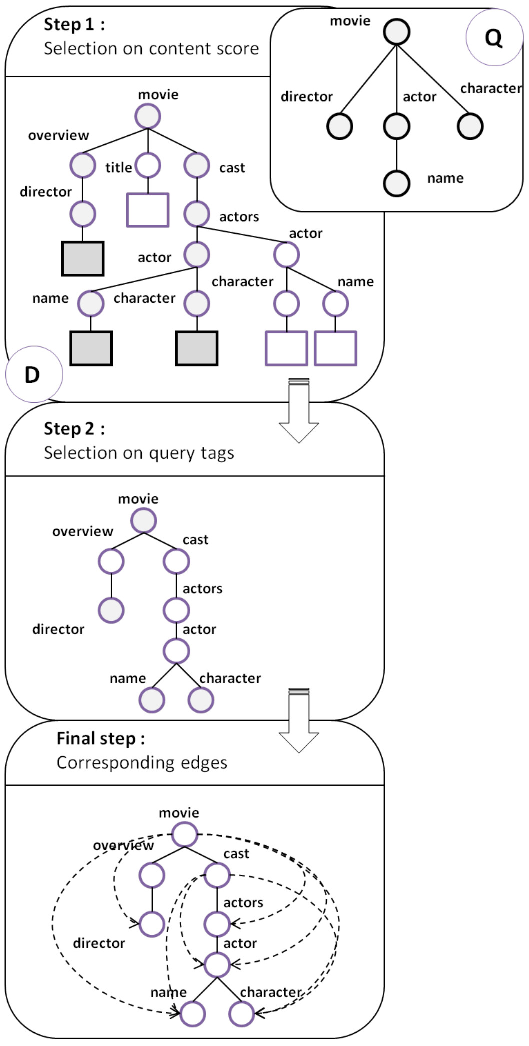


Figure 4: Example of a tree pruning: Step 1: relevant leaf nodes according to content conditions are selected; Step 2: corresponding paths starting from a node sharing a tag with the query and ending with a node also sharing a tag with the query are extracted; Step 3: interesting paths are concatenated.

4.3.1 Results for various combinations of λ in equation 12

We first studied the evolution of our performance over the MAiP measure of INEX 2010 for various settings of the λ parameter of equation 12. Results are shown in Figure 5. Structure improves scores on all runs for $\lambda > 0.1$. Our best results are obtained for λ between 0.5 and 0.8, which means that the overall relevance of T depends on both content and structure. However, a small amount of structure tends to improve the scores overall. Another observation is that using the context of the document while smoothing doesn't seem to be beneficial (runs JelinekT and DirichletT outperform runs JelinekTD and DirichletTD). Finally, the Jelinek smoothing outperforms the Dirichlet one.

4.3.2 Performances compared to INEX participants

In order to compare our approach with INEX official participants to the Datacentric task 2010, we fixed $\lambda = 0.5$ in equation 12 as it gives the best results overall. Table 3 presents results for the MAiP metric. We did not use MAgP as this metric over-ranks

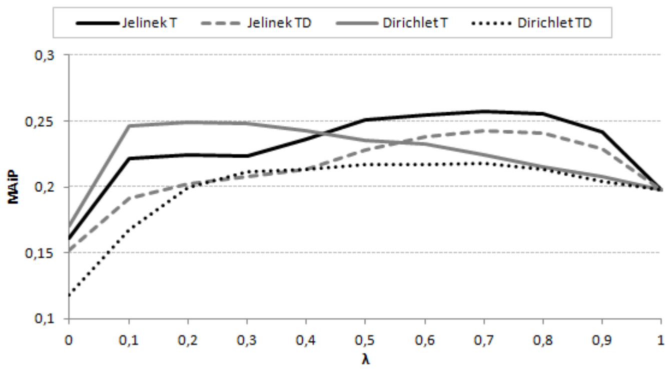


Figure 5: MAiP variations for various settings of λ parameter in equation 12

Runs	MAiP	% Improvement
Jelinek T	0.2541	**+29%**
Dirichel T	0.2353	+20%
Jelinek TD	0.2281	+16%
Dirichlet TD	0.2157	+10%
ufam2010Run2	0.1965	base
UPFL15TMI	0.1809	-8%

Table 3: Results compared to official participants best scores.

systems returning whole documents instead of elements.

The best runs considering the MAiP measure are the UFAM from [2] which re-generates the query structure based on the keywords and the *UPF* from Pompeu Fabra [5] which uses a content-based language model.
It appears that all our runs obtain better results compared to the best official participants ones. This tends to prove the interest of our approach.

5. CONCLUSION AND FUTURE WORK

In this paper we presented a semi-structured information retrieval model whose main originality is to use a language model to evaluate document-query structure likelihood. Results on the INEX 2010 Datacentric test set show the interest of the method.

In future work we plan to adapt and study more smoothing techniques. We also want to experiment different edge weighting method. In the same time we would like to evaluate the content through a language model approach. Finally we plan to integrate this new content evaluation directly into the structure matching process.

6. REFERENCES

[1] G. Amati and C. J. Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, 20:357–389, October 2002.

[2] F. Da C. Hummel, A. Da Silva, M Moro, and A. Laender. Automatically generating structured queries in XML keyword search. In *Proceedings of INEX 2010*, pages 194–205, 2011.

[3] Michael Fuller, Eric Mackie, Ron Sacks-Davis, and Ross Wilkinson. Coherent answers for a large structured document collection. In *Proceedings of SIGIR 1993*, pages 204–213, 1993.

[4] D. Ganguly, J. Leveling, G. Jones, S. Palchowdhury, S. Pal, and M. Mitra. DCU and ISI@INEX 2010: Adhoc and data-centric tracks. In *Proceedings of INEX 2010*, volume 6932, pages 182–193. 2011.

[5] R. Georgina. Upf at inex 2010: Towards query-type based focused retrieval. In *Proceedings of INEX 2010*, pages 206–218, 2011.

[6] Fang H. Using language models and topic models for xml retrieval. In *Proceedings of INEX 2007*, pages 94–102, 2008.

[7] D. Hiemstra. Statistical language models for intelligent xml retrieval. In *Group Communications and Charges. Technology and Business Models*, volume 2818, pages 107–118. 2003.

[8] D. Hiemstra and R. A. Baeza-Yates. Structured text retrieval models. In *Encyclopedia of Database Systems*, pages 2868–2871. 2009.

[9] F. Jelinek and Robert L. Mercer. Interpolated estimation of markov source parameters from sparse data. In *In Proceedings of the Workshop on Pattern Recognition in Practice*, pages 381–397, 1980.

[10] J. Kamps, J. Pehcevski, G. Kazai, M. Lalmas, and S. Robertson. Inex 2007 evaluation measures. In *Proceedings of INEX 2007*, pages 24–33, 2008.

[11] C. Laitang, K. Pinel-Sauvagnat, and M. Boughanem. Dtd-based costs for tree-edit distance in structured information retrieval. In *Proceedings of ECIR 2013*, 2013.

[12] Mounia Lalmas. *XML Retrieval*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2009.

[13] D. MacKay and L. Bauman Peto. A hierarchical dirichlet language model. *Natural Language Engineering*, 1:1–19, 1994.

[14] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. 2008.

[15] P. Ogilvie and J. Callan. Language models and structured document retrieval. In *Proceedings of INEX Workshop*, pages 33–40, 2002.

[16] P. Ogilvie and J. Callan. Hierarchical language models for xml component retrieval. In *Advances in XML Information Retrieval, INEX proceedings*, volume 3493, pages 269–285. 2005.

[17] P. Ogilvie and J. Callan. Parameter estimation for a simple hierarchical generative model for xml retrieval. In *Advances in XML Information Retrieval and Evaluation, INEX proceedings*, volume 3977, pages 211–224. 2006.

[18] L. Rongmei and T. P. van der Weide. Extended language models for XML element retrieval. In *Proceedings of INEX 2010*, pages 89–97, 2011.

[19] A. Trotman. Processing structural constraints. In *Encyclopedia of Database Systems*, pages 2191–2195. 2009.

[20] A. Trotman and Q. Wang. Overview of the inex 2010 data centric track. In *Proceedings of INEX 2010*, pages 171–181, 2011.

[21] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214.

[22] L. Zhao and J. Callan. Effective and efficient structured retrieval. In *Proceedings of CIKM 2009*, pages 1573–1576, 2009.