RUC @ INEX 2011 Data-Centric Track

Qiuyue Wang[1,2], Yantao Gan[1], Yu Sun[1]

[1] School of Information, Renmin University of China,
[2] Key Lab of Data Engineering and Knowledge Engineering, MOE,
Beijing 100872, P. R. China
qiuyuew@ruc.edu.cn, ganyantao19901018@163.com, yusun.aldrich@gmail.com

Abstract. We report our experiment results on the INEX 2011 Data-Centric Track. We participated in both the ad hoc and faceted search tasks. On the ad hoc search task, we employ language modeling approaches to do structured object retrieval, trying to capture both the structure in data and structure in query and unify the structured and unstructured information retrieval in a general framework. However, our initial experimental results using INEX test bed show that the unstructured retrieval model performs better than structured retrieval models. On the faceted search task, we propose a simple user-simulation model to evaluate the effectiveness of a faceted search system's recommending facet-values. We implemented the evaluation system and conducted the evaluations for the track. We also tested basic redundancy and relevance based approaches for recommending facet-values. The results show that our basic approaches of recommending facet-values perform quite well.

1 Introduction

More and more data on the Web are structured, e.g. data in Deep Web and Semantic Web, while most end users prefer to search any data with a simple keyword query interface. There has been increasing interest structured data retrieval in recent years. INEX 2011 Data-Centric Track is one of the efforts to investigate retrieval techniques on highly structured XML data, where rich structure carries important semantic and relationship information about pieces of data. Basically, structural information can be exploited to improve ad hoc retrieval performance, and also can be used to help users navigate or explore a large set of results as in faceted search systems. We participated in both the ad hoc and faceted search tasks.

On the ad hoc search task, we study how to exploit structural information in data, and how to infer and exploit structural information implicit in queries to improve retrieval performance. We base our studies on language modeling approaches, and propose structured language models to automatically infer structural information from unstructured queries, and match structure in data and in query probabilistically. Compared with other structured language modeling approaches in information retrieval, our approach can capture both structure in data and in query and unify structured and unstructured data retrieval in a general framework. However, our experimental results on INEX show that the structured models are inferior to the unstructured ones.

On the faceted search task, to avoid expensive and non-repeatable user studies, we propose a cost-based metric and implement an evaluation system based on user simulations to evaluate all the submitted runs for the task. We also tested the basic redundancy and relevance based approaches for recommending facet-values. Among all the submitted runs based on the same reference result list, the basic redundancy based approach performs best.

2 Ad Hoc Search Task

Language modeling approach has a solid statistical foundation, and can be easily adapted to model various kinds of complex and special retrieval problems, such as structured document retrieval. In particular, mixture models [1] and hierarchical language models [2][3][4] were proposed to be applied in XML retrieval. On the ad hoc search task in INEX 2011 data-centric track, we employ the language modeling approach to do structured object retrieval as the IMDB data collection can be viewed as a set of structured objects, i.e. movies and persons. With the rich structural information in data, we intend to investigate how to capture the structural information in data as well as that in query in language models to retrieve more accurate results for an ad hoc information need.

In this section, we discuss different ways of adapting language modeling approach to structured object retrieval, and evaluate them on the IMDB data collection.

2.1 Unstructured Data, Unstructured Query

The basic idea of language modeling approach in IR is to estimate a language model for each document (θ_D) and the query (θ_Q), and then rank the document in one of the two ways: by estimating the probability of generating the query string with the document language model, i.e. $P(Q|\theta_D)$, as in Equation 1, or by computing the Kullback-Leibler divergence of the query language model from the document language model, i.e. $D(\theta_Q \,/\!/\, \theta_D)$, as in Equation 2.

$$P(Q \mid \theta_D) = \sum_{w \in Q} P(w \mid \theta_D) \tag{1}$$

$$-D(\theta_Q \,\|\, \theta_D) = -\sum_{w \in V} P(w \mid \theta_Q) \log \frac{P(w \mid \theta_Q)}{P(w \mid \theta_D)} \tag{2}$$

On the surface, the KL-divergence model appears to be quite different from the query likelihood method. However, it turns out that the KL-divergence model covers the query likelihood method as a special case when we use the empirical distribution to estimate the query language model, i.e. maximum-likelihood estimate.

In IMDB data collection, each document is a structured object, i.e. movie or person. Our first retrieval strategy is to ignore the structural information in each object, estimate a language model for each object based on its free-text content, and rank them using query likelihood.

2.2 Structured Data, Unstructured Query

When taking into account the structural information in data, the common way is to view each object as consisting of multiple fields and represent its language model as a combination of different language models estimated from its different fields [5][6], as shown in Equation 3.

$$P(w|\theta_D) = \sum_{i=1}^{k} \lambda_i P(w|\theta_{D_i}) \quad (3)$$

Here we assume that object D consists of k fields. $P(w|\theta_{D_i})$ is the language model estimated from its ith field, which is normally smoothed by interpolating with the collection's ith field's language model.

The key problem in Equation 3 is to determine the combination weights, i.e. the value of λ_i. In the generative model, λ_i is set to be $P(\theta_{D_i}|\theta_D)$, the probability of object D generating the ith field, which could be uniform, i.e. $1/k$, or proportional to the frequency or length of the ith field in D as suggested in [3]. Alternatively, λ_i can be set to reflect the importance of the ith field to D, which could be learned or tuned to the task. If no training data exist, we suggest a new heuristic using the normalized average IDF values of all the terms in a field to determine the importance weight of this field. The intuition behind this is that the more discriminative the content of a field is, the more important this field is. For example, the "name" field of a person object is more important than its "birth_date" field. In [7], a concept of mapping probability is proposed to be used as λ_i to combine different fields' language models in Equation 3. The mapping probability of a given query term to a related field is the probability that this term maps to the field. Thus, the importance of a field depends on how likely query terms occur in this field.

Table 1 summarizes the various ways of determining λ_i, where N_i is the number of instances of the ith field in D and N is the number of instances of all fields in D. C_i denotes the set of distinct terms occurring in the ith field in the collection.

Table 1. Different concepts and calculations for determining λ_i.

Concept	Calculation
generating probability of the ith field: $P(\theta_{D_i}|\theta_D)$	AVG: $1/k$ FREQ: N_i/N LENGTH: $length(D_i)/length(D)$
importance or prior probability of the ith field: $P(\theta_i)$	AVG: $1/k$ IDF: $\frac{(\sum_{w\in C_i} idf_w)/|C_i|}{\sum_{j=1}^{k}[(\sum_{w\in C_j} idf_w)/|C_j]}$
posterior or mapping probability of the ith field given a query term w: $P(\theta_i|w)$	MAPPR: $\frac{P(w|\theta_i)P(\theta_i)}{\sum_{j=1}^{k}(P(w|\theta_j)P(\theta_j)}$

We implemented all the 5 strategies of calculating λ_i shown in Table 1, i.e. AVG, FREQ, LENGTH, IDF, MAPPR, and evaluate them on the IMDB data collection.

2.3 Structured Data, Structured Query

In the previous two sections, we treat a query as a bag of words with no structure. However, when search is against structured data, it is beneficial for the search system to discover the latent structural information in queries and exploit it to improve retrieval accuracy. For example, if a user wants to find a very recent romantic movie directed by Yimou Zhang, he/she may issue the keyword query "Yimou Zhang 2010 romance" to the system. If the retrieval system knows that the return element type should be *movie*, "Yimou Zhang" should be the *director* of the movie, "2010" and "romance" should appear in the *release-date* and *genre* fields respectively, the retrieval precision can be greatly improved and the correct movie "The Love of the Hawthorn Tree" could be returned.

Since asking users to provide such information for each query would be an excessive burden on them and most users are only willing to express the simplest forms of queries, a lot of research work has been done to automatically infer structured queries from the keyword queries submitted by users [8][9][10]. This process is typically divided into three steps: firstly, generating all possible structured queries from the input unstructured query by incorporating the knowledge of schemas, data statistics, heuristics, user/pseudo relevance feedback and etc.; secondly, ranking the structured queries according to their likelihood of matching user's intent; thirdly, selecting the top-k structured queries and evaluating them. However, if the inferred structured queries are not intended by the user, we cannot expect to get the right result. Another line of research work done in this direction is to infer some structural hints, not necessarily hard structural constraints as in structured queries, from the input keyword query, such as the mapping probability proposed in [7].

In this paper, we use structured language models to capture the structural hints in query. With a keyword query and data collection, we first infer a structured language model for the query as shown in Equation 4, and then we match query's structured language model with each object's structured language model by computing their KL divergence and rank the objects accordingly. With Equation 3 and 4, we can use Equation 2 to compute the KL divergence between two language models. But to address the structural matching, we use Equation 5 to compute the structural KL divergence, that is, to compute the overall KL divergence between the language models on all fields.

$$P(w|\theta_Q) = \sum_{i=1}^{k} \pi_i P(w|\theta_{Q_i}) \tag{4}$$

$$-D(\theta_Q \| \theta_D) = -\sum_{i=1}^{k} D(\theta_{Q_i} \| \theta_{D_i}) = -\sum_{i=1}^{k} \sum_{w \in V} P(w|\theta_{Q_i}) \log \frac{P(w|\theta_{Q_i})}{P(w|\theta_{D_i})} \tag{5}$$

Now, the key issue is how to estimate a structured language model for a keyword query. With no other information about queries, we use the pseudo relevance feedback approach similar to that used in [11] to estimate the structured language model for a query. Note that in [11] documents are unstructured while in our case documents are structured, so the details of our estimator are different from [11]'s. Since query and document models are modeling the same types of objects, we assume that the field weights in the query model, π_i, are the same as the field weights in the

document model, λ_i, and both are equal to $P(\theta_i)$, the field importance or prior probabilities. Given a set of feedback documents F, we assume that the content of each document field are sampled from a model mixing the query's and collection's corresponding field language models, which model the relevant and non-relevant information respectively as shown in Equation 6, where α_D is a document-specific mixing weight parameter.

$$P(w|\theta_{D_i}) = \alpha_D P(w|\theta_{Q_i}) + (1-\alpha_D)P(w|\theta_{C_i}) \tag{6}$$

All the parameters to be estimated can thus be represented as $\Lambda = \{\{P(w|\theta_{Q_i})\}_{i=1,\ldots,k}, \{\alpha_D\}_{D\in F}\}$. We use the Maximum A Posterior (MAP) estimator as shown in Equation 7 to estimate the parameters, where $P(F|\Lambda)$ is the likelihood of the feedback documents given in Equation 8 and $P(\Lambda)$ is a conjugate prior on all the field language models of the query as shown in Equation 9. In Equation 9, $P(w|\overline{\theta_{Q_i}})$ is the prior field language model of the query and parameter μ specifies our confidence about the prior.

$$\hat{\Lambda} = \arg\max_{\Lambda} P(F|\Lambda)P(\Lambda) \tag{7}$$

$$P(F|\Lambda) = \prod_{D\in F}\prod_{i=1}^{k}\prod_{w\in V} P(w|\theta_{D_i})^{c(w,D_i)} \tag{8}$$

$$P(\Lambda) \propto \prod_{i=1}^{k}\prod_{w\in V} P(w|\theta_{Q_i})^{\mu P(w|\bar{\theta}_{Q_i})} \tag{9}$$

The MAP estimate can be found using the EM algorithm. We introduce a hidden variable for the identity of each word in each field of a document, $z_{D_i,w}$, and $P(z_{D_i,w} = Q_i)$ is the probability that the word w in the ith field of document D is generated by the query's ith field's language model and $P(z_{D_i,w} = C_i)$ is the probability that the word w in the ith field of document D is generated by the collection's ith field's language model. We have $P(z_{D_i,w} = Q_i) + P(z_{D_i,w} = C_i) = 1$. The updating formulas are as follows:

E-step:

$$P^{(n+1)}(z_{D_i,w} = Q_i) = \frac{\alpha_D^{(n)} P^{(n)}(w|\theta_{Q_i})}{\alpha_D^{(n)} P^{(n)}(w|\theta_{Q_i}) + (1-\alpha_D^{(n)})P(w|\theta_{C_i})} \tag{10}$$

M-step:

$$\alpha_D^{(n+1)} = \frac{\sum_{w\in V}\sum_{i=1}^{k} c(w,D_i)P^{(n)}(z_{D_i,w} = Q_i)}{\sum_{w\in V}\sum_{i=1}^{k} c(w,D_i)} \tag{11}$$

$$P^{(n+1)}(w|\theta_{Q_i}) = \frac{\sum_{D\in F} c(w,D_i)P^{(n+1)}(z_{D_i,w} = Q_i) + \mu P(w|\bar{\theta}_{Q_i})}{\sum_{w\in V}\sum_{D\in F} c(w,D_i)P^{(n+1)}(z_{D_i,w} = Q_i) + \mu} \tag{12}$$

As the input query is unstructured, we set its prior field language model as $P(w|\bar{\theta}_{Q_i}) = \frac{c(w,Q)}{|Q|}$, and the collection language model is $P(w|\theta_{C_i}) = \frac{c(w,C_i)}{|C_i|}$. To avoid tuning the prior confidence value μ manually, we adopt the strategy in [12]. Thus, we start with a sufficiently large μ and discount it a little bit at each iteration of the EM algorithm until the stopping condition is reached. In our experiments, we set the initial prior confidence μ=5000 and the discounting factor δ =0.9. The EM iterations start with $\alpha_D^{(0)} = 0.1,\ P^{(0)}(w|\theta_{Q_i}) = \frac{c(w,Q)}{|Q|}$.

2.4 Experimental Results

We implemented the structured language modeling approaches discussed in the previous two sections inside the source code of Lemur 4.11. In this section, we will report our experiment results on the ad hoc search task of INEX 2011 data-centric track.

We indexed all the leaf fields in the IMDB XML data collection using the Indri. Each field is uniquely identified by its full XML path. But we take only the last tag name of the path as the field's name. Even though some different XML paths have identical last tag names, they are of same semantics in the IMDB collection, for example, "name" wherever it appears means a person's name and "title" means a movie's title no matter in a movie or a person object. In total, there are 49 distinct field names for the IMDB data collection, 31 fields for movie objects and 23 fields for person objects. We built the index using the Krovetz stemmer and a short stop word list that contains only eleven common words: {*a, an, and, as, by, in, of, or, that, the, to*}.

In the retrieval models discussed above, the document and document field language models, i.e. θ_D and θ_{D_i}, are estimated using maximum likelihood estimate, which are then smoothed using a Dirichlet prior with the collection or collection field language models. We set the Dirichlet prior for smoothing document language model in the unstructured language modeling approach to be 1000, which is optimal tuned on INEX 2010 data-centric track topics. In our submitted runs, we set the Dirichlet prior for smoothing document field language models in the structured language modeling approaches to be 1000 too. The results showed that the performance of structured language modeling approaches were much worse than that of unstructured approaches. This was an unfair comparison however since the optimal Dirichlet prior for smoothing document field language models may not be the same as that for document language model. Actually the lengths of document fields are typically short, much shorter than the average document length. Many fields contain only a couple of terms, e.g. director's name, release date of a movie. So we rerun our experiments setting the Dirichlet prior for document field language models to be much smaller than 1000, e.g. 10, 30, 50 and 100. The performance of structured language models is much improved. Since the lengths of different fields vary a lot, for example the plot or biography fields usually contain large pieces of text, it is better to set different

smoothing Dirichlet priors for different fields. How to appropriately smooth the document field language models in the structured language modeling approaches is an interesting question that we would like to explore in the future.

The results of different retrieval strategies are shown in Table 2. The first run is our baseline, which uses the standard KL-divergence method to rank the documents with no structure information taken into account as described in Section 2.1, with no relevance feedback (*fb=0*) and the Dirichlet prior for smoothing document language models is set 1000 (*dir=1000*). It is also the best official run (run ID is *p2-ruc11AS2*) that we submitted to the INEX 2011 data-centric track. Run 2-1 to 2-5 use the structured language models and KL divergence as discussed in Section 2.2 to rank structured documents, where *avg*, *idf*, *mapping_prob*, *length*, and *freq* are the five strategies of computing the field weights. For each strategy of computing field weights, we set the Dirichlet prior 10, 30, 50, and 100, and choose the best one in terms of *P@5* from the four results and put it in Table 2. For example, the best Dirichlet prior for *avg* strategy in terms of *P@5* is 10, so we put the result of "*avg, dir=10, fb=0*" in the table. But for *idf* strategy, the best Dirichlet prior is 100. Run 3-1 to 3-5 employ the pseudo relevance feedback to estimate a structured language model for each unstructured query and then rank the structured documents using structured KL divergence as discussed in Section 2.3. *fb=10* means that we take the top 10 documents returned by the retrieval method in Section 2.2 to estimate the structured query language model.

Table 2. Effectiveness of unstructured and structured language modeling approaches.

ID	parameters of runs	MAP	1/rank	P@5	P@10	P@20	P@30
1-1	unstructured, dir=1000, fb=0 (p2-ruc11AS2)	0.3829	0.6441	0.4474	0.4132	0.3842	0.3684
2-1	avg, dir=10, fb=0	0.307	**0.7383**	**0.5368**	**0.4868**	**0.4596**	**0.4368**
2-2	idf, dir=100, fb=0	0.2864	0.6738	0.4789	0.4263	0.3737	0.3421
2-3	mapping_prob, dir=10, fb=0	0.2295	0.6029	0.3895	0.3789	0.3461	0.3254
2-4	length, dir=100, fb=0	0.1996	0.5445	0.3737	0.3316	0.2763	0.2439
2-5	freq, dir=10, fb=0	0.1547	0.4698	0.3474	0.3	0.2697	0.2544
3-1	avg, dir=1000, fb=10	0.1993	0.5542	0.3895	0.3447	0.2974	0.2614
3-2	idf, dir=1000, fb=10	0.2025	0.5461	0.4	0.35	0.3013	0.2684
3-3	mapping_prob, dir=1000, fb=10	0.2027	0.5326	0.3842	0.3316	0.2882	0.2553
3-4	length, dir=1000, fb=10	0.1947	0.553	0.3842	0.3474	0.2882	0.2579
3-5	freq, dir=1000, fb=10	0.1876	0.5512	0.3789	0.3342	0.2829	0.2526
	Best in INEX 2011	**0.3969**	0.6999	-	0.4421	0.4171	0.3851

From Table 2, we can observe that structured language modeling approaches can greatly improve the early precision, but they also hurt recall and thus the average precision. The early precision of run 2-1 is much better than our best official run and also beats the best early precisions from all the submitted runs to INEX 2011 data-centric track. Structured relevance feedback approaches unexpectedly perform worse

than the unstructured approach and structured approaches without feedback. It demands further investigation.

3 Faceted Search Task

For the faceted search task, as it is the first year, our work focused on evaluating effectiveness of faceted search systems based on user simulation approach, and testing a simple redundancy based approach for recommending facet-values.

3.1 Faceted Search Evaluation

Faceted search systems are typically evaluated through user studies, which are expensive and not repeatable. For INEX 2011 data-centric track, we proposed to employ user simulation technology to evaluate faceted search systems. The measurement is the interaction cost defined as the number of results, facets or facet-values examined by the user before he/she encounters the first relevant result, which is similar to the Reciprocal Rank (RR) metric in traditional IR. Readers are referred to Section 5 in the track overview paper [13] for the detailed discussion about the user model and cost model assumed in our evaluation system.

We implemented the user simulation system in Java, which can evaluate both dynamic faceted search modules and static facet-value hierarchies. The only difference is that there is no EXPAND option for static facet-value hierarchies. The evaluation results of all the submitted runs to the faceted search task are given in Section 6.2 of [13]. We can see that the cost based metric and user simulation based evaluation system are feasible in comparing different facet-value recommending systems. We also tested the robustness of the evaluation metric in terms of different user models. In [13], we assume that the user model is the ***First-Match*** model, i.e. at each step user selects the first relevant facet-value among the list of recommended ones. In Table 3 and 4, we show the evaluation results for the ***Stochastic*** and ***Myopic*** user models respectively. Stochastic users randomly select a relevant facet-value from the recommended list, while myopic users select the facet-value that is contained in the least number of results. From the tables, we can see that the evaluation results for different user models are consistent. As the numbers of runs and topics are too small, we can hardly draw any conclusions, and we cannot test the robustness of the evaluation metric in terms of the number of topics and so on. In principle, such RR-like metrics are less robust than NDCG-like metrics.

Table 3. Evaluation results of all static runs in terms of NGs and ANG using ***Stochastic*** model.

run / NG	p4-UAms2011indri-c-cnt	p4-UAms2011indri-cNO-scr2	p4-UAms2011lucene-cNO-lth	p48-MPII-TOPX-2.0-facet-entropy (TopX)	p48-MPII-TOPX-2.0-facet-entropy (Lucene)	p4-2011IlpsFtScore	p4-2011IlpsNumdoc	p18-2011UPFfixG7DAnh	p18-2011UPFfixGDAh	p18-2011UPFfixGDAh2	p2-2011Simple1Run1

201	0.47	0	-	0	-	-	-	-	-	-	-
202	0	0	0	0	0	0	0	0	0.11	0	0
203	0	0.46	0	-	0	0.79	0.79	0	0	0	0.83
204	0.68	0.87	0.94	-	0	0	0.91	0	0	0.88	0.89
205	0	0	0.75	0	0.74	0.68	0.68	0.90	0.90	0.90	0.75
207	0	0.74	0	0	0	0	0	0	0	0	0.92
208	0	0	0	0	0	0	0	0	0	0	0
209	0	0	0	0	0	0	0	0	0	0	0
210	0.79	0.68	-	0	-	-	-	-	-	-	-
211	0	0	0.51	0	0	0	0	0.31	0.6	0.22	0.31
212	0.89	0.89	-	0	-	-	-	-	-	-	-
213	0.90	0.90	-	-	-	-	-	-	-	-	-
214	0	0	0.45	0	0	0	0	0	0.55	0.25	0
ANG	0.29	**0.35**	0.20	0	0.06	0.11	0.18	0.09	0.16	0.17	**0.28**

Table 4. Evaluation results of all static runs in terms of NGs and ANG using ***Myopic*** model.

run / NG	p4-UAms2011indri-c-cnt	p4-UAms2011indri-cNO-scr2	p4-UAms2011lucene-cNO-lth	p48-MPII-TOPX-2.0-facet-entropy (TopX)	p48-MPII-TOPX-2.0-facet-entropy (Lucene)	p4-2011llpsFtScore	p4-2011llpsNumdoc	p18-2011UPFixG7Danh	p18-2011UPFixGDAh	p18-2011UPFixGDAh2	p2-2011Simple1Run1
201	0.47	0	-	0	-	-	-	-	-	-	-
202	0	0	0	0	0	0	0	0	0.05	0	0
203	0	0.59	0	-	0	0.79	0.79	0	0	0	0.83
204	0.77	0.80	0.94	-	0	0	0.94	0	0	0.90	0.89
205	0	0	0.92	0	0.74	0.88	0.75	0.90	0.94	0.90	0.81
207	0	0.74	0	0	0	0	0	0	0	0	0.90
208	0	0	0	0	0	0	0	0	0	0	0
209	0	0	0	0	0	0	0	0	0	0	0
210	0.79	0.76	-	0	-	-	-	-	-	-	-
211	0.66	0.66	0.51	0	0	0	0	0.31	0.6	0.22	0.22
212	0.90	0.89	-	0	-	-	-	-	-	-	-
213	0.90	0.92	-	-	-	-	-	-	-	-	-
214	0	0	0.5	0	0	0	0	0.14	0.43	0.25	0
ANG	0.34	**0.41**	0.22	0	0.06	0.13	0.19	0.10	0.16	0.17	**0.28**

3.2 Recommending Facet-Values

Given a broad or exploratory query, a search system typically returns a long list of results. A faceted search system can help users navigate through the result list to identify item of interest by recommending facet-value conditions for refining the query at each navigation step. Essentially, the task is to construct a hierarchy of facet-values that can cover all the relevant results in the result list. As all the results in the result list could be of users' interest and it is assumed that the faceted search interface can present no more than 20 facet-value conditions at a time and each result page contain no more than 10 results, the task can be reformulated more precisely as to

construct a hierarchy of facet-values that covers all the results in the result list with each leaf node covering no more than 10 results and the fan-out of each non-leaf node no more than 20.

User's interaction cost with a hierarchy of facet-values, i.e. the number of results, facets or facet-values that the user examined, is proportional to the size of the hierarchy. The larger the hierarchy is, the more facet-values a user could potentially examine. So our main goal of the task is to construct a minimum hierarchy satisfying the conditions. We then divided the goal into two sub-goals:

1. Cover the current result list with a minimum number of facet-values at each branching node of the hierarchy.
2. Minimize the average path length of the hierarchy.

That is, we intend to construct a hierarchy that is neither too wide nor too deep. However, these two sub-goals compete with each other. The first sub-goal closely resembles the well-known NP-hard set-cover problem, which is commonly solved by a greedy algorithm [14]. The greedy algorithm adds at each step the facet-value that covers the maximum number of uncovered results until the whole result set is covered. But this could probably result in very long paths in the hierarchy. For instance, at the node of facet-value fv_1, current result list contains 100 results, and a facet-value fv_2 covers 99 results and will be selected by the greedy algorithm, then next at the node of fv_2, the greedy algorithm may select a facet-value fv_3 that covers 98 results, and so on. A long path, $fv_1(100) \rightarrow fv_2(99) \rightarrow fv_3(98) \rightarrow \ldots$, is thus formed in the hierarchy. This could cause a high interaction cost. Also in practice, users may find such a facet-value condition useless since they can hardly recognize any change in the result list once they choose it to refine the query. We adopt a simple heuristic to avoid such a problem. We set a threshold for the number of results that a facet-value reduces the current result list. In our experiments, the threshold is set 20. That is, if a facet-value reduces the result list by less than 20 results, it will not be selected to be recommended. The whole algorithm for recommending a list of facet-values given a list of results is shown in Algorithm 1.

Algorithm 1 Recommending Facet-Values

Input: current result list R_c
Output: a list of recommended facet-values L
begin
1 $L := \Phi$;
2 $R := R_c$;
3 FV := set of all possible facet-values in R;
4 $fv := \text{argmax}_{fv \in FV \&\& (|R|-|R_{fv}|) \geq 20}\{|R_{fv}|\}$; //$R_{fv}$: set of results in R that are covered by fv
5 append fv to L;
6 $R := R - R_{fv}$;
7 if ($R \neq \Phi$ && $|L| < 20$) then
8 goto 3;
9 endif;
10 return L;
end.

In Algorithm 1, line 4 is to select the facet-value that covers the most unseen results while reducing the result list by at least 20 results. Once a facet-value is selected into the recommended list, all the results it covers would be removed from the result list. These are presented in line 5 and 6. The process continues until all the results in the given result list are covered or the number of recommended facet-values reaches the upper bound 20, which is set by the track.

In addition, we extend Algorithm 1 by taking into account the relevance scores of the unseen results covered by a facet-value. That is, at each step of the greedy algorithm, i.e. at line 4 in Algorithm 1, the facet-value with the highest relevance score instead of the most frequency on the unseen result set is selected to be added into the recommended list. The relevance score of a facet-value is defined as the sum of the relevance scores of all the unseen results covered by the facet-value.

3.3 Results

Two static facet-value hierarchies are generated using Algorithm 1 and its extension respectively, both based on the reference result list provided by the track. Table 5 presents the evaluation results for them in terms of NGs and ANG under *first-match* user model. We call Algorithm 1 "most-frequent" strategy, and its extension "most-relevant" strategy. Note that the run generated by "most-frequent" strategy is our submitted run, whose ID is p2-2011Simple1Run1. In the table, we ignore the topics that the reference result list contains no relevant result, which has no effect on the evaluation results.

Table 5. Evaluation results of Algorithm 1 and its extension in terms of NGs and ANG.

algorithms	202	203	204	205	207	208	209	211	214	ANG
most-frequent	0.21	0.86	0.91	0.81	0.94	0	0	0.6	0	**0.33**
most-relevant	0.21	0	0.9	0.73	0.94	0	0	0.64	0.64	0.31

The results of two runs show no significant difference, even though the ANG of most-frequent strategy is a little greater than that of most-relevant one. In general, most-frequent strategy performs better on topics whose relevant results appear late in the result list, e.g. topic 203, while most-relevant strategy is better on topics whose relevant results appear early in the result list, e.g. topic 214. Compared with all other submitted runs to INEX generated on the reference result list, our static hierarchy is the most effective one in terms of ANG. The best performed run is p4-UAms2011indri-cNO-scr2 submitted by University of Amsterdam, but based on the result list generated by Indri, which contains relevant results for all topics while the reference result list contains no relevant results for four topics. Thus the results are not comparable. So in INEX 2012, all the submitted runs to the faceted search task should be required to be on the same result list.

4 Conclusions and Future Work

In this paper, we presented our work on INEX 2011 Data-Centric Track. We participated in both the ad hoc and faceted search tasks. We employ structured language models to capture structure both in data and in query and utilize them in ad hoc search on data-centric XML. But the experimental results on INEX are not promising for the structured language models. We are going to investigate this further on to gain more and deeper insights into structured data retrieval models and techniques.

For the faceted search task, we proposed to use a cost-based metric and user-simulation model to evaluate all the runs. The results show that it is a feasible way for faceted search evaluation. But given the very small number of topics and runs, the robustness of the metric could not be established. We intend to test it on more faceted search systems with more topics, and compare its results with that of user studies, to see how real and how robust it is. For the first year of the task, we simply tested two basic strategies for recommending facet-values, redundancy and relevance based approaches. As our future work, we are also interested in studying more sophisticated approaches for recommending facet-values.

References

1. D. Hiemstra, "Statistical Language Models for Intelligent XML Retrieval", Intelligent Search on XML data, H. Blanken et al. (Eds.), 2003.
2. P. Ogilvie, J. Callan, "Language Models and Structured Document Retrieval", INEX 2003.
3. P. Ogilvie, J. Callan, "Hierarchical Language Models for XML Component Retrieval", INEX 2004.
4. P. Ogilvie, J. Callan, "Parameter Estimation for a Simple Hierarchical Generative Model for XML Retrieval", INEX 2005.
5. P. Ogilvie, J. Callan, "Combining Document Representations for Known-Item Search", SIGIR 2003.
6. Z. Nie, Y.Ma, S. Shi, J. Wen, W. Ma, "Web Object Retrieval", WWW 2007.
7. J. Kim, X. Xue, W.B. Croft, "A Probabilistic Retrieval Model for Semistructured Data", ECIR 2009.
8. D. Pektova, W.B. Croft, Y. Diao, "Refining Keyword Queries for XML Retrieval by Combining Content and Structure", ECIR 2009.
9. X. Li, Y. Wang, A. Acero, "Extracting Structured Information from User Queries with Semi-Supervised Conditional Random Fields", SIGIR 2009.
10. N. Sarkas, S. Paparizos, P. Tsaparas, "Structured Annotations of Web Queries", SIGMOD 2010.
11. C. Zhai, X. Lu, X. Ling, X. He, and et al, "UIUC/MUSC at TREC 2005 Genomics Track", TREC 2005.
12. T. Tao, C. Zhai, "Regularized Estimation of Mixture Models for Robust Pseudo-Relevance Feedback", SIGIR 2006.
13. Qiuyue Wang, Georgina Ramírez, Maarten Marx, Martin Theobald, Jaap Kamps, "Overview of INEX 2011 Data-Centric Track", INEX 2011.
14. David S. Johnson, "Approximation algorithms for combinatorial problems", Journal of Computer and System Sciences, 9:256-278, 1974.