Generating Pseudo Test Collections for Learning to Rank Scientific Articles

Richard Berendsen, Manos Tsagkias,
Maarten de Rijke, and Edgar Meij

ISLA, University of Amsterdam, Science Park 904,
1098 XH Amsterdam, The Netherlands
{r.w.berendsen,e.tsagkias,derijke,edgar.meij}@uva.nl

Abstract. Pseudo test collections are automatically generated to provide training material for learning to rank methods. We propose a method for generating pseudo test collections in the domain of digital libraries, where data is relatively sparse, but comes with rich annotations. Our intuition is that documents are annotated to make them better findable for certain information needs. We use these annotations and the associated documents as a source for pairs of queries and relevant documents. We investigate how learning to rank performance varies when we use different methods for sampling annotations, and show how our pseudo test collection ranks systems compared to editorial topics with editorial judgements. Our results demonstrate that it is possible to train a learning to rank algorithm on generated pseudo judgments. In some cases, performance is on par with learning on manually obtained ground truth.

1 Introduction

Recent years have seen increasing interest in generating pseudo test collections for training and evaluation purposes. This is primarily motivated by the costs associated with obtaining manual relevance assessments. Most approaches to generating ground truth leverage some kind of human behavior, such as annotation, hyperlinking, or simply using a search engine. Beitzel et al. [3] use the Open Directory Project, a large scale annotation effort targeting web pages in general. They assume relevance of documents to the title of the category they are listed under to generate relevance judgments. More recently, Asadi et al. [1] use anchor texts as queries and assume linked-to documents are potentially relevant documents. Web search is characterized by heterogeneous and high volume content and usage data. We investigate the generation of pseudo test collections in the less studied and more specialized domain of digital libraries.

Digital libraries are increasingly publishing their content online allowing people to access, browse, and search the archives. This type of content is typically semi-structured and manually annotated using rich descriptors. These characteristics differentiate it from web documents, and many retrieval methods have been developed to exploit them, improving retrieval effectiveness [6]. Modern Information Retrieval (IR) algorithms—especially in the form of learning to rank (LTR) methods—are able to learn to combine relatively uncertain evidence from individual features and typically improve retrieval effectiveness when large amounts of training data are available [14].

In this paper, we focus on generating pseudo test collections which can be used to optimize retrieval algorithms for ad-hoc search on domain-specific, semi-structured documents. The most commonly used method for generating pseudo test collections is to sample and group documents in a collection by a certain criterion, and generate queries for these groups [1, 3]. In the domain of digital libraries, rich annotations are often available in the form of thesaurus terms, classification codes, or other descriptors that can be used as grouping criteria. Our leading intuition is that people provide this metadata in order to make documents better findable with regard to certain information needs. In this paper, we use such annotations to group documents in *topics*, and generate simulated queries (*pseudo-queries*) for and from these topics. The set of documents assigned to a topic is considered to be the relevant set of documents for the topic.

In the pseudo test collection generation process there are three key challenges that shape our research questions and contributions: (a) how to use annotations for grouping documents, (b) which documents to allow in the groups, and (c) how to simulate queries. The common, and cornerstone ingredient among these challenges is the sampling of annotations. Not all annotations are equally specific (compare, e.g., "United States of America" and "workaholism"). Developing methods for sampling descriptors from different metadata fields can help manipulating the generality and specificity of the resulting groups and therefore the resulting performance of LTR. In this work we tackle each of these challenges, using the domain-specific characteristics of ad-hoc search in scientific articles.

We discuss related work in Section 2. We present our methods in Section 4, conduct experiments in Section 5, report on our results in Section 6, discuss our findings in Section 7, and conclude in Section 8.

2 Related work

We generate our pseudo test collections from the GIRT corpus. The GIRT corpus was first used in CLEF 2000 for the cross-lingual IR subtask [11], and later for monolingual domain-specific retrieval [12]. The usefulness of annotations as query expansion terms and reformulation was soon discovered [17]. Stemming and morphological analysis were the main emphasis in the CLEF 2007 monolingual version of the domain-specific task [5]. In CLEF 2008, groups using variants of pseudo relevance feedback managed to obtain the best performance [6, 16]. These findings suggest that the use of annotations can prove useful for simulating topics, and the adaptation of pseudo relevance feedback ideas can help in the query simulation process.

The issue of creating and using pseudo test collections is a longstanding and recurring theme in IR, see, e.g., [22, 23]. Over the years, several attempts have been made to either simulate human queries or to generate relevance judgments without the need of human assessors for a range of tasks. Azzopardi et al. [2]

Algorithm 1 Algorithm for creating pseudo test collections for semi-structured domain-specific collections

```
1: Sample an annotation dimension $k$.
2: Sample an annotation $i$ from $A_k$.
3: Simulate query $q_{i,k}$.
```

simulate queries for known-item search and investigate several term weighting methods for query generation. Kim and Croft [10] generate a pseudo test collection for desktop search. Huurnink et al. [8] use click-through data to simulate relevance assessments, and later they evaluate the performance of query simulation methods in terms of system rankings [9]. They find that incorporating document structure in the query generation process results in more realistic query simulators.

In the realm of web search, Beitzel et al. [3] use documents listed under the categories in the Open Directory Project as relevant documents for queries that they generate from the titles of these categories. Most similar to our work is the work by Asadi et al. [1]. They use anchor texts to generate queries, and treat the documents linked to as pseudo-relevant documents for training a learning to rank system. Our work differs in the domain characteristics: we have no anchor texts, but we have rich metadata, like authors, co-authors, year of publication, keywords, and classifications.

3 Problem Statement

We first define the problem of generating pseudo test collections for semi-structured documents, and then describe our approach to this problem. A pseudo test collection is defined here as consisting of a set of generated queries Q and, for each query $q \in Q$, a set of documents assumed to be relevant, R_q. Given this definition, there are two main steps involved: (a) simulating the query and (b) simulating the relevant documents.

Our idea is to use the document annotations for this. Let a document d be annotated using k annotation "dimensions," each corresponding to a separate descriptor field. Each document has a set of $A_k := \{\alpha_{1,k}, \ldots, \alpha_{i,k}\}$ annotations corresponding to the k-th dimension. We can estimate a relevant set of documents $R_{i,k}$ for the i-th annotation in the k-th dimension from all documents that share $\alpha_{i,k}$. From the documents in $R_{i,k}$, we can also estimate a simulated query. This way of thinking about the problem breaks it down to the subproblems listed in Algorithm 1.

Our goal is to develop sampling methods that optimize the effectiveness of a learning to rank system in the setting of semi-structured domain-specific retrieval. In contrast to other pseudo test collection research, we are not primarily interested in developing methods that produce pseudo test collections similar to manually crafted test collections. We choose to evaluate our methods on the end-to-end performance of an LTR system, i.e., train on pseudo test collections generated by our methods, and test on manually crafted collections.

4 Sampling Methods

Below we discuss instantiations for all the steps in Algorithm 1; we begin with sampling an annotation dimension (STEP 1), then sampling annotations (STEP 2), and simulating queries (STEP 3).

STEP 1: Sampling annotation dimensions. We start with Step 1 in Algorithm 1. In the GIRT collection, there are three main annotation dimensions: `METHOD` can be any of 40 research methods, e.g., "descriptive study," `CLASSIFICATION` is a a classification code, e.g., "Labor Market Policy," and `CONTROLLED` is a thesaurus term, e.g., "social partnership." The first two (`METHOD`, `CLASSIFICATION`) dimensions cover broad topics, while annotations from `CONTROLLED` range from very broad to very narrow topics.

We sample annotations in two ways: from each dimension individually, and from all dimensions simultaneously. In the first case, we generate pseudo test collections using only annotations from one dimension, `CONTROLLED`, because it offers a range of more general and more specific coverage, just like we would expect in queries. In the second case, we take the cross product $A_{METHOD} \times A_{CLASSIFICATION} \times A_{CONTROLLED}$ and the relevant sets of documents consist of documents that are annotated with the triple of annotations over the three dimensions.

STEP 2: Sampling annotations. For Step 2 of Algorithm 1 we use two techniques for sampling annotations from annotation dimension `CONTROLLED`: randomly sampling single annotations, and randomly sampling pairs of annotations (sampling from $A_{CONTROLLED} \times A_{CONTROLLED}$), where the relevant sets of documents have both annotations. In the first case we observed that annotations ranged from broad to specific. Very specific annotations were associated with a very small number of documents, while some others were found very broad and were associated with a large fraction of the documents in the collection. Our second sampling method using pairs of annotations aims at accounting for this phenomenon: documents that have both annotations are intuitively more on topic than documents that have only one of the two.[1] Our third sampling strategy samples annotations from $A_{METHOD} \times A_{CLASSIFICATION} \times A_{CONTROLLED}$, as already noted above. In all three cases, to ensure that our sampled annotations are neither too broad or too specific, we select single annotations or pairs of annotations that are associated with between 100 and 1000 documents. The lower bound warrants enough training examples for the learning to rank system, while the upper bound discards very broad annotations.

STEP 3: Simulating queries. For simulating the queries in STEP 3 of Algorithm 1, we use two approaches: (i) use the annotations as queries, and (ii) extract query terms from the simulated relevant set of documents.

[1] We also experimented with sampling using larger numbers of annotations. The number of documents associated with them was found small, therefore of little use for training LTR systems.

Our first query simulation method is straightforward. Query terms are sampled from the content of sampled annotation(s). Our second simulation method is inspired by the observation that pseudo relevance feedback helps to improve retrieval effectiveness [6]. For extracting query terms from the relevant set of documents, we choose to use the log-likelihood ratio (LLR) score [15]. Our choice is motivated by the fact that most documents in the GIRT collection lack abstracts, which raises data sparsity issues due to the short length of titles. In this setting, probabilistic methods for query simulation [2] which build on language redundancy may prove less useful due to sparsity issues.

LLR is defined as the symmetric Kullback-Leibler divergence of the expected and observed term probability in two corpora, one being a background corpus. In other words, terms are ranked by how discriminative they are for both corpora. Stopwords, or common terms will rank lower because they also occur in the background corpus. For our purposes, we set one corpus to be documents in the relevant set $R_{i,k}$, and the other to consist of the rest of the documents in the collection. For every $R_{i,k}$ terms are ranked in descending order by their log-likelihood ratio score. To generate the query we take the top-T terms ranked by LLR. In all our experiments, T is set to 10.

5 Experimental Setup

We describe our research questions and the experiments we conduct to answer them. We evaluate our methods of constructing pseudo test collections with regard to their effectiveness for training an LTR system which is then tested on the GIRT collection *and* with regard to the system rankings they produce.

Our main research question is whether using annotations found in semi-structured scientific documents are useful for simulating relevant sets of documents, and queries for training a learning to rank system. We focus on the following questions:

Sampling methods What is the effect of our sampling methods on LTR retrieval effectiveness? Do they generalize in different topic sets of the same collection? Is performance of our sampling methods different from performance obtained by training on editorial topics and judgments?

System rankings Are the generated pseudo test collections useful for evaluation purposes, i.e., do they produce similar system rankings as manual collections?

We generate pseudo test collections that use both single annotations and pairs of annotations from the `CONTROLLED` dimension, and triples of annotations over all three dimensions (i.e., `METHOD`, `CLASSIFICATION`, `CONTROLLED`). In each case, we kept only topics with between a hundred and a thousand documents, resulting in the following numbers of pseudo topics: 2,073, 7,039 and 4,161, respectively. Each of these sampling methods is coupled with two query simulation methods: using keywords and using LLR. Further, we investigate the generalization of our methods by using two topic sets of the GIRT collection, i.e., from 2007, and 2008. This results in 12 experimental conditions.

We evaluate our generated pseudo test collections in two ways: (a) on the retrieval effectiveness of an LTR system, (b) on the similarity of system rankings they produce and system rankings produced on real topics. For the first type of evaluation we use two topic sets, from the CLEF domain specific track: the 2007 topics, and the 2008 topics. We are interested in how training on our pseudo test collections compares to training on real topics. More concretely: How does training on the 2007 topics compare to training on the pseudo test collections when the learned models are tested on the 2008 topics?; and vice versa for the 2008 topics. In addition, we generate two"oracle" runs for each year, namely, an LTR system that trains and tests on the manual topics and assessments in the respective year. For the second type of evaluation on similarity of system ranking, we compare rankings of retrieval systems on manual topics and assessments, and the generated pseudo test collections using Kendall's τ, following [24].

Dataset. We use the collection used in the CLEF domain specific track in 2008 in our experiments. It has two corpora, the GIRT corpus and the CSA SA corpus; for collection statistics see [18].

Learning to rank. For retrieval we use a learning to rank approach. We use a perceptron based algorithm from [19, 20] which was set to optimize performance for the area under the ROC curve. We use two sets of features: (a) query-independent, and (b) query dependent. Table 1 lists 11 query-dependent features (top-half), which are the outputs of off-the-shelf retrieval systems, and 9 query-independent features.

For the query-dependent features we use the Indri, and Terrier retrieval frameworks. For Indri indexing, we use a Porter stemmer, but no stopword removal. For Terrier indexing, we use single-pass indexing, with stopword removal followed by Porter stemming. Both with Indri and Terrier we index all fields, also the keyword field. We normalize features as follows. For the Indri language modeling runs (Indri-LM, Indri-BOW, Indri-BUW, Indri-PRF) we take the exponential of the scores. Then, for each feature, we normalize by dividing by the maximal value for that feature over all documents. In addition to the query dependent features listed in Table 1, we use the query clarity feature by [4].

Our query-independent features include degree-centrality and closeness-centrality. These are properties of nodes in an undirected graph that can be used as measures of influence or centrality in a collaboration network [7]. We calculated them on the co-author graph where nodes are authors and edges exist between authors who co-authored at least one paper, using NetworkX.[2] We assumed that two author fields refer to the same author if and only if the strings match exactly. Query-independent features have values equal to or greater than zero. We normalize each feature by dividing it through its maximal value over all documents.

For our retrieval experiments, we report on average mean precision (MAP). Statistical significance testing is done using Fisher's pairwise randomization

[2] http://networkx.lanl.gov

Table 1. Query-dependent, and query-independent features for learning to rank. For the features that use properties of authors, we calculate four different values, one based on the first author, and three calculated based on all authors: the maximal, minimal and average value.

Abbr	Description and parameters
Query-dependent features	
Indri-tf-idf	Tf-idf run, with $k_1 = 1.2$ and $b = 0.75$.
Indri-okapi	Okapi BM25 run, with $k_1 = 1.2$, $b = 0.75$, and $k_3 = 7$.
Indri-LM	Language modeling, with Dirichlet smoothing, $\mu = 2500$.
Indri-BOW	LM with boolean ordered window.
Indri-BUW	LM with boolean unordered window.
Indri-PRF	pseudo-relevance feedback(which is based on [13]), we use the 10 top pseudo-relevant documents, we extract 10 terms, we give the original query 0.5 weight and use $\mu = 0$.
Terrier-tf-idf	Tf-idf run, with $k_1 = 1.2$, $b = 0.75$.
Terrier-DFRee	a parameter free DFR (Divergence from Randomness) model.
Terrier-PL2	another DFR run, with $c = 1.0$.
Terrier-QE	a query expansion run, with DFR model Bose-Einstein 1. Query is expanded with the top 10 terms, obtained from the top 3 documents.
Terrier-DSM	a DFR proximity dependence model, with proximity ngram length of 2, $SD = 1$, FD=1, and using pBiL. For this model, block indexing has to be performed. We set block.size to 1.
Query-independent features	
docLength	Number of terms in title and abstract.
nAuthors	Number of authors of article.
age	Age of publication (2008 - publication year).
Pubs	Nr. of publications by authors {max,first,avg,min}.
CoAuth	Nr. of co-authors of authors {max,first,avg,min}.
Degree	Degree-centrality of authors {max,first,avg,min}.
Close	Closeness-centrality of authors {max,first,avg,min}.
Pagerank	Pagerank of authors {max,first,avg,min}.

test [21], with $\alpha = 0.001$. We use a conservative α level to keep Type I errors under control, as we are making many pairwise comparisons.

6 Results

Our first experiment aims at answering the question whether training on pseudo test collections leads to different performance from training on editorial test collections. In Table 2 we list performances of our learning to rank algorithm on two sets of queries: the topics (title only) for the 2007 and 2008 editions of the CLEF Domain-Specific track. In the first column it is specified on which topics we train. In the second column the way of obtaining the queries is listed. In the third column we report MAP obtained on the 2007 test topics. We list in

Table 2. MAP performance of our learning to rank approach on the CLEF Domain-Specific 2007 and 2008 topics. ($A_{CONTR.}$ and $A_{CLASSIF.}$ are short for $A_{CONTROLLED}$ and $A_{CLASSIFICATION}$.)

Editorial test collections			
Topics		2007	2008
2008, title only		0.2347	*0.3158*
2007, title only		*0.2226*	0.2970
Pseudo test collections			
Annotations	Query generation	2007	2008
$A_{CONTR.}$	use keywords	**0.1985**	0.2734
$A_{CONTR.}$	using LLR	**0.1155**	**0.1869**
$A_{CONTR.} \times A_{CONTR.}$	from keywords	**0.2091**	0.2866
$A_{CONTR.} \times A_{CONTR.}$	using LLR	**0.1240**	**0.1959**
$A_{METHOD} \times A_{CLASSIF.} \times A_{CONTR.}$	from keywords	**0.1329**	**0.1609**
$A_{METHOD} \times A_{CLASSIF.} \times A_{CLASSIF.}$	using LLR	**0.1979**	0.2602

boldface the runs that are significantly different from the run that was trained on the 2008 queries. In the last column, MAP on the 2008 topics is given. We list in boldface the runs that differ significantly from the run trained on the 2007 data. The two oracle runs—runs that train and test on the same queries—are given in italics.

When we evaluate on the 2008 test topics, we see that three of our six methods of generating a pseudo test collection yield performance that is similar to training on the 2007 test topics: the differences are not statistically significant. This result provides first evidence for the utility of our pseudo test collection generation methods.

Looking at which methods perform well, we see that for $A_{CONTROLLED}$, it is best to use terms occurring in the annotation as query terms, rather than generating a query with LLR, which is worse on both the 2007 and 2008 topics, even though the difference is only significant on the 2007 topics. We observe a similar result for $A_{CONTROLLED} \times A_{CONTROLLED}$; in this case using LLR is significantly worse for both 2007 and 2008. However, for $A_{METHOD} \times A_{CLASSIFICATION} \times A_{CONTROLLED}$, generating the query with LLR is more successful, significantly so for the 2008 topics.

Evaluating on the 2007 test topics yields a different picture. In this case all our methods are significantly outperformed by a learning to rank system trained on the 2008 topics.

We now take a look at the oracle runs. On 2008 test topics, the oracle run is best. Even this run, however, does not improve significantly over $A_{CONTROLLED}$ (using keywords) or $A_{CONTROLLED} \times A_{CONTROLLED}$ (using keywords). It also does not improve significantly over the run that trains on the 2007 topics. On the 2007 test topics, the oracle run is improved by the run that was trained on the

2008 topics, but the difference is not significant. The oracle run obtains a higher score than all our pseudo test collection generation methods, but the differences with $A_{CONTROLLED} \times A_{CONTROLLED}$ and $A_{METHOD} \times A_{CLASSIFICATION} \times A_{CONTROLLED}$ (LLR) are not significant.

6.1 Performance of Individual Features

For completeness, we list scores of our individual features in Table 3, ordered decreasingly by MAP on 2008 topics. The best query-dependent feature is Terrier-QE. However, for 2007, it does not improve significantly over the other Terrier features. Also, with regard to the learning to rank runs: for 2007, it does not significantly outperform the runs that trained on 2008 topics, the 2007 topics, or $A_{CONTROLLED} \times A_{CONTROLLED}$ (using keywords). It is significantly better than all other runs for 2007. For 2008, Terrier-QE does not significantly outperform Indri-tf-idf, nor the other Terrier features. With regard to the learning to rank runs, it does not significantly outperform the runs that train on the 2007 topics, the 2008 topics, or $A_{CONTROLLED}$ (using keywords). All other runs are significantly outperformed.

Some query-dependent feature scores are very high, and even outperform some of our learning to rank approaches. Our main focus is not on showing that we can outperform the best query-dependent feature. Rather, it is to show that we can use pseudo-topics and pseudo-judgments for training with the same effectiveness as editorial topic and judgments.

Table 3. MAP performance of our individual query-dependent features.

Abbr	2007	2008
Indri-tf-idf	0.2028	0.2723
Indri-okapi	0.1821	0.2707
Indri-LM	0.1835	0.2051
Indri-PRF	0.1854	0.1984
Indri-BUW	0.0733	0.1678
Indri-BOW	0.0531	0.1344
Terrier-QE	0.2599	0.3360
Terrier-DFRee	0.2183	0.3107
Terrier-DSM	0.2355	0.3085
Terrier-tf-idf	0.2381	0.2941
Terrier-PL2	0.2277	0.2794

6.2 Using Pseudo Test Collections For Evaluation

In principle, pseudo test collections can be used for evaluation purposes. In Table 4 we list Kendall's tau values between system rankings produced by different test collections. The systems ranked here are the same retrieval algorithms we used for our query dependent features.

There is a reasonable correlation between how the 2007 and 2008 topics rank our query-dependent features. The first two rows (and the first two columns) show how all pseudo systems rank systems compared to editorial topics. There are no negative correlations here. It is interesting to note that the pseudo test collection with the strongest correlation with an editorial test collection is $A_M \times A_{CL} \times A_{CT}$; the method that uses documents associated with an annotation triple (`METHOD,CLASSIFICATION,CONTROLLED`). This is in stark contrast with our previous observation that this pseudo test collection should not be used to train a learning to rank system on.

Table 4. Kendall's tau values between system rankings produced by different test collections.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
(1) 2007	1.000	0.745	0.309	0.294	0.382	0.294	0.636	0.404
(2) 2008		1.000	0.418	0.110	0.564	0.110	0.891	0.220
(3) A_{CT}			1.000	-0.147	0.564	-0.147	0.382	-0.037
(4) A_{CT} (LLR)				1.000	-0.110	0.982	0.000	0.800
(5) $A_{CT} \times A_{CT}$					1.000	-0.110	0.600	0.000
(6) $A_{CT} \times A_{CT}$ (LLR)						1.000	0.000	0.800
(7) $A_M \times A_{CL} \times A_{CT}$							1.000	0.110
(8) $A_M \times A_{CL} \times A_{CT}$ (LLR)								1.000

7 Discussion

We have shown that it is possible to use the rich annotations available in digital libraries collections for training a learning to rank system. We assumed that people annotate documents to make them better findable for certain information needs. We identified three main steps, addressing what kind of annotations to use, how to sample annotations, and how to generate queries. We tackled all three steps and showed that it is possible to generate pseudo test collections in the digital library domain on which a learning to rank system can be trained, such that in some cases performance is indistinguishable from training on editorial topics and judgments. In particular, when testing on the 2008 topics, for three pseudo test collections it holds that training on them yields performance on par with training on 2007 editorial judgments. There is room for improvement with regard to training on the 2008 topics: this strategy outperforms our methods when tested on the 2007 topics.

There are some limitations in our work, which we aim to address in future work. One of them is that our learning to rank algorithm is unable to outperform our best query-dependent feature. We plan to experiment with other learning to rank algorithms and to go beyond using such an algorithm as a black-box.

Another limitation is that we used off-the-shelf retrieval algorithms, and did not tune their parameters. This may limit the quality of our features. It is easy to improve with learning to rank over some of these features, but it is a much harder problem to improve over the best feature. We plan to tune parameters for every query-dependent feature. By tuning them on pseudo test-collections, we can show another way to put pseudo test-collections to good use.

There are some interactions that we do not yet fully understand. One of them is the following. Recall that Asadi et al. [1] sample non-relevant documents from the bottom of a retrieval algorithm ranked list, and we followed this procedure. We chose Indri-LM, but noticed that the choice of algorithm to use has a considerable impact on performance. For example, selecting the Indri-tf-idf algorithm instead of Indri-LM made oracle run performance drop from about MAP 0.30 to MAP 0.25 for 2008 topics. Our choice of the Indri-LM retrieval function was arbitrary, as of yet we have a limited understanding of the properties such a retrieval function should have.

Performance of our query-independent features was also disappointing. The Pegasos [20] algorithm we used for learning to rank learns a linear model, and the weights for all our query-independent features were close to zero. We used 24 document independent features in this paper, but none of them seemed promising enough in a learning to rank setting in order to use them in the query generation process. In future work, we plan to use richer collections which give us the opportunity to test stronger query-independent features based on the citation graph.

8 Conclusion

We have shown that it is feasible to generate pseudo test collections for training a learning to rank system on scientific document collections. We proposed three pseudo test collection generation methods for which we could show that for one of our test sets, training on these collections is just as effective as training on editorial topics and judgments. We pointed to interesting directions for future work and areas where we need to deepen our understanding.

Acknowledgements. This research was partially supported by the European Union's ICT Policy Support Programme as part of the Competitiveness and Innovation Framework Programme, CIP ICT-PSP under grant agreement nr 250430, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreements nr 258191 (PROMISE Network of Excellence) and 288024 (LiMoSINe project), the Netherlands Organisation for Scientific Research (NWO) under project nrs 612.061.814, 612.061.815, 640.004.802, 380-70-011, 727.-011.005, 612.001.116, the Center for Creation, Content and Technology (CCCT), the Hyperlocal Service Platform project funded by the Service Innovation & ICT program, the WAHSP and BILAND projects funded by the CLARIN-nl program, the Dutch national program COMMIT, and by the ESF Research Network Program ELIAS.

9 References

[1] N. Asadi, D. Metzler, T. Elsayed, and J. Lin. Pseudo test collections for learning web search ranking functions. In *SIGIR '11*, pages 1073–1082. ACM, 2011.
[2] L. Azzopardi, M. de Rijke, and K. Balog. Building simulated queries for known-item topics: an analysis using six european languages. In *SIGIR '07*, pages 455–462. ACM, 2007.
[3] S. Beitzel, E. Jensen, A. Chowdhury, and D. Grossman. Using titles and category names from editor-driven taxonomies for automatic evaluation. In *CIKM '03*, pages 17–23. ACM, 2003.
[4] S. Cronen-Townsend and W. Croft. Quantifying query ambiguity. In *HLT 2002*, pages 104–109. Morgan Kaufmann Publishers Inc., 2002.

[5] G. M. Di Nunzio. Working notes CLEF 2007, Appendix C, Results of the Domain Specific Track. *Working notes CLEF 2007*, 2007.

[6] G. M. Di Nunzio. Working notes CLEF 2008, Appendix D, Results of the Domain Specific Track. In *Working notes CLEF 2008*, 2008.

[7] D. Easley and J. Kleinberg. *Networks, crowds, and markets*. Cambridge University Press, 2010.

[8] B. Huurnink, K. Hofmann, and M. de Rijke. Simulating searches from transaction logs. In *SIGIR 2010 Workshop on the Simulation of Interaction*, 2010.

[9] B. Huurnink, K. Hofmann, M. de Rijke, and M. Bron. Validating query simulators: an experiment using commercial searches and purchases. In *CLEF '10*, pages 40–51. Springer-Verlag, 2010.

[10] J. Kim and W. B. Croft. Retrieval experiments using pseudo-desktop collections. In *CIKM '09*, pages 1297–1306. ACM, 2009.

[11] M. Kluck and F. Gey. The domain-specific task of CLEF-specific evaluation strategies in cross-language information retrieval. In *Cross-Language Information Retrieval and Evaluation*, pages 48–56. Springer, 2001.

[12] M. Kluck and M. Stempfhuber. Domain-specific track CLEF 2005: Overview of results and approaches, remarks on the assessment analysis. In *Working notes CLEF 2005*, 2005.

[13] V. Lavrenko and W. Croft. Relevance based language models. In *SIGIR '01*, pages 120–127. ACM, 2001.

[14] T.-Y. Liu. *Learning to Rank for Information Retrieval*. Springer, 2011. ISBN 978-3-642-14266-6.

[15] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, 1999.

[16] E. Meij and M. de Rijke. The University of Amsterdam at the CLEF 2008 Domain Specific Track - parsimonious relevance and concept models. In *CLEF '08 Working Notes*, 2008.

[17] V. Petras. How one word can make all the difference - using subject metadata for automatic query expansion and reformulation. In *Working notes CLEF 2005*, 2005.

[18] V. Petras. The domain-specific track at CLEF 2008. In *Working notes CLEF 2008*, 2008.

[19] D. Sculley. Combined regression and ranking. In *KDD '10*, pages 979–988. ACM, 2010.

[20] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *24th international conference on Machine learning*, pages 807–814. ACM, 2007.

[21] M. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *CIKM '07*, pages 623–632. ACM, 2007.

[22] J. Tague and M. Nelson. Simulation of user judgments in bibliographic retrieval systems. In *SIGIR '81*, pages 66–71, 1981.

[23] J. Tague, M. Nelson, and H. Wu. Problems in the simulation of bibliographic retrieval systems. In *SIGIR '80*, pages 236–255, 1980.

[24] E. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. *Information Processing & Management*, 36(5):697–716, 2000.