# Topic Difficulty Prediction in Entity Ranking

Anne-Marie Vercoustre[1], Jovan Pehcevski[2], and Vladimir Naumovski[2]

[1] INRIA, Rocquencourt, France
anne-marie.vercoustre@inria.fr
[2] Faculty of Management and Information Technologies, Skopje, Macedonia
{jovan.pehcevski,vladimir.naumovski}@mit.edu.mk

**Abstract.** Entity ranking has recently emerged as a research field that aims at retrieving entities as answers to a query. Unlike entity extraction where the goal is to tag the names of the entities in documents, entity ranking is primarily focused on returning a ranked list of relevant entity names for the query. Many approaches to entity ranking have been proposed, and most of them were evaluated on the INEX Wikipedia test collection. In this paper, we show that the knowledge of predicted *classes* of topic difficulty can be used to further improve the entity ranking performance. To predict the topic difficulty, we generate a classifier that uses features extracted from an INEX topic definition to classify the topic into an experimentally pre-determined class. This knowledge is then utilised to dynamically set the optimal values for the retrieval parameters of our entity ranking system. Our experiments suggest that topic difficulty prediction is a promising approach that could be exploited to improve the effectiveness of entity ranking.

## 1 Introduction

The INitiative for Evaluation of XML retrieval (INEX) started the XML Entity Ranking (XER) track in 2007 [4] with the goal of creating a test collection for entity ranking using the Wikipedia XML document collection [6]. The XER track was run again in 2008, introducing new tasks, topics and pooling strategies aiming at improving the XER test collection [5]. The objective of the two INEX XER tracks was to return names of entities rather than full documents that correspond to those entities as answers to an INEX topic. Different approaches to entity ranking have been proposed and evaluated on the two INEX Wikipedia XER test collections, which resulted in many advances to this research field. However, little attention has been put on the impact of the different types (or classes) of topics on the entity ranking performance.

Predicting query difficulty in information retrieval (IR) has been the subject of a SIGIR workshop in 2005 that focused both on prediction methods for query difficulty and on the potential applications of those predicted methods [2]. The applications included re-ranking answers to a query, selective relevance feedback, and query rewriting. On the other hand, the distinction between easy and difficult queries in IR evaluation is relatively recent but offers important insights and new perspectives [8, 23]. The difficult queries are defined as the ones on which the evaluated systems are the less successful. The initial motivation for query difficulty prediction was related to the need for evaluation measures that could be applied for robust evaluation of systems across different collections [19, 20]. Recently, Mizzaro [13] also advocated that the evaluation measures should reward systems that return good results on difficult queries more than on easy ones, and penalise systems that perform poorly on easy queries more than on difficult ones.

In this paper, we build on the above arguments and develop a method for query (topic) difficulty prediction in the research field of XML entity ranking. Our approach is based on the generation of a topic classifier that can classify INEX XER topics from a number of features extracted from the topics themselves (also called static or a-priori features) and possibly from a number of other features calculated at run time (also called dynamic or a-posteriori features). The main goal is to apply the topic difficulty prediction to improve the effectiveness of our entity ranking system that was evaluated as one of the best performing XER systems at INEX 2007 and 2008 [4, 5].

## 2 Topic difficulty classification

In this section, we classify the INEX XER topics by their difficulty. After a brief description of the two INEX XER tracks, we present our methodology to identifying the different classes of topics.

### 2.1 XML entity ranking at INEX

The two INEX XER test collections comprise 46 topics (2007) and 35 topics (2008) with corresponding relevance assessments available, most of which were proposed and assessed by the track participants [4, 5]. Figure 1 shows an example of an INEX 2007 XER topic definition. In this example, the `title` field contains the plain content only query, the `description` provides a natural language description of the information need, and the `narrative` provides a detailed explanation of what makes an entity answer relevant. In addition to these fields, the `categories` field provides the target category of the expected entity answers (used in the entity ranking task), while the `entities` field provides a few examples of the expected entity answers (used in the list completion task).

Eight participating groups submitted in total 35 XER runs in 2007, while six participants submitted another 33 XER runs in 2008. The INEX XER track does not offer a large test collection, however this is currently the only test collection available for the purposes of our topic difficulty classification. In this paper we focus on the list completion task, corresponding to a training data set comprising a total number of 46 topics and 17 runs, and a testing data set comprising a total number of 35 topics and 16 runs.

```
<inex_topic>
<title>circus mammals</title>
<description>
I want a list of mammals which have ever been tamed to perform in circuses.
</description>
<narrative>
Each answer should contain an article about a mammal which can be a part
of any circus show.
</narrative>
<categories>
   <category id="138">mammals</category>
</categories>
<entities>
   <entity id="379035">Asian Elephant</entity>
   <entity id="4402">Brown Bear</entity>
</entities>
</inex_topic>
```

**Fig. 1.** INEX 2007 XER topic definition.

### 2.2 Identifying classes of topics

The classification of topics into groups is based on how well the runs submitted by participating systems answered to the topics. For each topic, we calculate the topic difficulty using the Average Average Precision (AAP) measure [14]. AAP is the average AP of all the submitted runs for a given topic: the higher the AAP, the easier the topic.

We define two methods for grouping topics into classes depending on the number of groups we want to build, either two or four classes to experiment with two different types of classes. For grouping the topics into two classes (Easy and Difficult), we use the mean AAP measure as a splitting condition: if AAP for a given topic is superior to the mean AAP (calculated across all topics) then the topic is classified as Easy otherwise it is classified as Difficult. For grouping the topics into four classes (Easy, Moderately_Easy, Moderately_Difficult, and Difficult), we use the mean AAP and the standard deviation around the mean as a splitting condition:

**if** AAP >= (mean AAP + stDev) **then** Easy topic
**if** AAP >= (mean AAP) **and** AAP < (mean AAP + stDev) **then** Moderately_Easy topic
**if** AAP >= (mean AAP - stDev) **and** AAP < (mean AAP) **then** Moderately_Difficult topic
**if** AAP < (mean AAP - stDev) **then** Difficult topic

The above two or four classes of INEX XER topics are then used as a basis for evaluating our automatic feature-based topic classification algorithm, as described in Section 4.

## 3 Our entity ranking system

Our approach to identifying and ranking entities combines: (1) the full-text similarity of the entity page with the query; (2) the similarity of the page's categories with the categories of the entity examples; and (3) the link contexts found in the top ranked pages returned by a search engine for the query.

We developed an XER system that involves the following modules.

1. The *topic module* takes an INEX topic as input and generates the corresponding full-text query and the list of entity examples.
2. The *search module* sends the query to a search engine[3] and returns a list of scored Wikipedia pages. The assumption is that a good entity page is a page that answers the query.
3. The *link extraction module* extracts the links from a selected number of highly ranked pages, together with the information about the paths of the links (XML paths). The assumption is that a good entity page is a page that is referred to by a page answering the query; this is an adaptation of the HITS [9] algorithms to the problem of entity ranking.
4. The *linkrank module* calculates a weight for a page based (among other things) on the number of links to this page. The assumption is that a good entity page is a page that is referred to from contexts with many occurrences of the entity examples. A coarse context would be the full page that contains the entity examples. Smaller and better contexts may be elements such as paragraphs, lists, or tables [16].
5. The *category similarity module* calculates a weight for a page based on the similarity of the page categories with the categories attached to the entity examples. The assumption is that a good entity page is a page associated with a category close to the categories of the entity examples [18].
6. The *full-text module* calculates a weight for a page based on its initial search engine score.

The global score $S(t)$ for a target entity page is calculated as a linear combination of three normalised scores, the linkrank score $S_L(t)$, the category similarity score $S_C(t)$, and the full-text score $S_Z(t)$:

$$S(t) = \alpha S_L(t) + \beta S_C(t) + (1 - \alpha - \beta) S_Z(t) \tag{1}$$

where $\alpha$ and $\beta$ are two parameters whose values can be tuned differently depending on the entity retrieval task.

Details of the global score and the three separate scores can be found in previous publications [16, 18]. In this paper we are interested in automatically adapting the values of $\alpha$ and $\beta$ parameters to the topic, depending on the topic class. By predicting the optimal values for $\alpha$ and $\beta$ parameters that correspond to each class of topic difficulty, we aim at improving the performance score of our system over the current best performance score that uses pre-defined static values for the $\alpha$ and $\beta$ parameters.

[3] We used Zettair, an open source search engine: http://www.seg.rmit.edu.au/zettair/

# 4 Topic difficulty prediction

In this section we present our methodology for predicting topic difficulty. Our approach is based on generating a classifier to classify topics in two or four classes (as described in Section 2). The classifier is built using features extracted from the INEX topic definition. We use the open source data mining software Weka [21] developed by the University of Waikato. Weka is a collection of machine learning algorithms for data mining tasks that, given a training set of topics, can generate a classifier from the topics and their associated features.

## 4.1 Topic features

From the specific structure of the INEX 2007 XER topics we developed 32 different a-priori features. We call these *a-priori* (or static) features because each of them can be calculated by using only the topic definition before any processing is made by our system. The features include the number of words (excluding stop-words) found in the topic title, the topic description and the topic narrative, respectively; the number of verbs and sentences found in the description or narrative part, as well as the number of words in the union or intersection of these parts of the topic. Additional features are built using the ratio between previous features, for example the ratio between the number of words in the title and the description, or the ratio between the number of sentences in the description and the narrative. The idea is that if the narrative needs more explanation than the title or the description, it may be because good answers could be difficult to identify among many irrelevant pages. Counting verbs required some natural language processing. We used the NLTK toolkit software [12] which especially helped with the different features concerning verb forms.

Other a-priori features are related to the target categories and the entity examples listed in the topic. The target categories can be either very broad or very specific and they represent indications about the desired type of answer entities, not hard constraints. There could be a correlation between the number of target categories and the topic performance that we wanted to identify. Other features involve not just the topic description but also the INEX Wikipedia test collection, for example, the number of Wikipedia pages attached to the target categories. We also count the number of different Wikipedia categories attached to the entity examples in the topic. Finally we create features that represent the union or intersection of target categories and categories attached to the entity examples.

We also defined a few *a-posteriori* (dynamic) features that could be calculated at run time, i.e. when sending the topic (query) to our system. These features include the number of links from the highly ranked pages returned by the search engine, the number of contexts identified in those pages and the number of common categories attached to the entity examples and the answer entities.

**Table 1.** Nine topic features that correlated well with the topic difficulty prediction. In the table, `w` stands for words, `narr` for narrative, `desc` for description, `t_cat` for target categories, and `e_cat` for categories attached to entity examples. For example, `#sent_narr` is the number of sentences in the narrative part of the INEX topic.

| Number | Description | Features |
|---|---|---|
| 1 | Topic definition features | #sent_narr |
| 2 – 6 | Ratio of topic definition features | #w_title/#w_narr, #w_intersec(title,narr)/#w_union(title,narr) #w_intersec(desc,narr)/#w_union(desc,narr), #w_intersec(title,desc)/#w_union(title,desc,narr), #w_intersec(desc,narr)/#w_union(title,desc,narr) |
| 7 – 8 | Topic definition and Wikipedia features | #pages_per_t_cat, #intersec(e_cat) |
| 9 | Ratio of topic definition and Wikipedia features | #intersec(e_cat)/#union(e_cat) |

### 4.2 Topic classifier

The next step was to identify among the 32 features those that best correlated with the topic difficulty, i.e. the features that would be usable by a classifier to predict between different classes of topics. We first generated many classifiers, each one associated with a random subset of the 46 INEX 2007 XER topics. The classifiers were generated using the Weka j48 classifier based on the well known Quinlan's C4.5 statistical classifier [17], with each training topic subset classified using the topic classification explained in Section 2. We then manually analysed all the decision trees generated by Weka to identify the features that were actually used by the generated classifiers. As a result, we could extract a small subset of nine features that correlated well with topic difficulty.

Table 1 shows the nine features used to generate the training topic subsets. We discovered that the dynamic (a-posteriori) features had no influence on the generated classifiers, and so we only used the a-priori features.

### 4.3 Training and testing topic sets

For each training subset of INEX 2007 XER topics that we used previously for generating a classifier, we used the testing set comprising all the 35 INEX 2008 XER topics for evaluating the performance of this classifier. We tried many different mostly random combinations of training topic subsets, but because of their relatively small sizes on average the accuracy of the correctly classified instances was around 71%.

To improve the accuracy, we used a well known approach that combines several decision trees, each generated from slightly different topic sets. This is known as Random Forests [1] and was used in query prediction by Yom-Tov et al. [22]. Before implementing the combined classifier, we carefully built the training topic set for each individual predictor so that the included topics were representative of different features.

**Table 2.** Accuracy achieved by the six two-class classifiers on the 35 INEX 2008 topics.

| | **Class** | |
|---|---|---|
| | **2** | |
| **Classifier** | Correct | Incorrect |
| 1 | 24/35 (68%) | 11/35 (32%) |
| 2 | 25/35 (71%) | 10/35 (29%) |
| 3 | 25/35 (71%) | 10/35 (29%) |
| 4 | 25/35 (71%) | 10/35 (29%) |
| 5 | 24/35 (68%) | 11/35 (32%) |
| **combined** | 26/35 (**74%**) | 9/35 (**26%**) |

We manually divided the training set of 46 INEX 2007 XER topics into four subsets of around 35 topics each. We had to do it manually in order to get as much different and representative topics as possible, especially because of the small topic subset sizes. So those four subsets and the one with all 46 topics made five different training sets from which we built five separate classifiers. For these and the final combined classifier we also had to build a testing topic set that does not include any of the training topics. The INEX 2008 XER topic set was used for this purpose.

### 4.4 Validation of topic difficulty prediction

The final topic difficulty prediction classifier was built using a simple voting system which is the reason why we needed an odd number of classifiers. For building a two-class classifier the voting algorithm is trivial: for a topic we get a prediction from the five classifiers and count the number of predictions as Easy and the number of predictions as Difficult; the majority gives the prediction for the final classifier. For example, if the predictions from the five classifiers are [diff, easy, easy, diff, diff], the combined prediction is Difficult.

The combined two-class classifier resulted in a precision of 74% on our testing set which is better than what we could achieve with a single classifier. Table 2 shows the accuracy achieved by each of the six classifiers.

We also considered the possibility of building a four-class classifier (Easy, Moderately_Easy, Moderately_Difficult, and Difficult). A similar voting algorithm is used by simply choosing diff : easy = 0:5 and diff : easy = 5:0 to be predicted as Easy and Difficult topics, respectively, with the diff : easy = (1:4 | 2:3) and diff : easy = (4:1 | 3:2) combinations resulting in Moderately_Easy and Moderately_Difficult topics, respectively. The combined four-class classifier resulted in an accuracy of 31% on our testing set which was much less than that achieved by the two-class classifier.

**Table 3.** Estimated values for optimal $\alpha/\beta$ system parameters, as measured by MAP using the 46 topics of the INEX 2007 XER training collection.

| | Class | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **2** | | | | **4** | | | | | | | |
| | Easy | | Diff | | Easy | | modEasy | | modDiff | | Diff | |
| **Measure** | $\alpha$ | $\beta$ | $\alpha$ | $\beta$ | $\alpha$ | $\beta$ | $\alpha$ | $\beta$ | $\alpha$ | $\beta$ | $\alpha$ | $\beta$ |
| MAP | 0.2 | 0.6 | 0.1 | 0.8 | 0.0 | 0.7 | 0.2 | 0.6 | 0.1 | 0.8 | 0.5 | 0.0 |

## 5 Applying topic difficulty prediction in entity ranking

Our objective with topic difficulty prediction was to improve the performance of our XER system by dynamically tuning the values for system parameters according to the predicted topic class. Specifically, for each of the 35 INEX 2008 topics in the testing set, we adapt the values for the $\alpha$ and $\beta$ system parameters in accordance with the estimated optimal values observed on the training set.

### 5.1 Choosing optimal system parameters by topic difficulty

We first estimated the optimal values for the system parameters by using the 46 INEX 2007 XER training topics and by also taking into account their corresponding topic difficulty classes. We generated all the possible 66 runs by respectively varying the values of $\alpha$ and $\beta$ from 0 to 1 by increment of 0.1. For each topic, we then measured the average precision (AP) for each run and ordered the runs by decreasing value of AP. This way we could identify the values of the two ($\alpha$, $\beta$) parameters that performed *best* for each individual topic. To estimate which ($\alpha$, $\beta$) pair would be *optimal* for a given topic difficulty class, we used the topics that belong to a particular class (such as Easy), and calculated the mean AP (MAP) for each run that appeared at least once among the ten highly ranked runs for a given topic.[4] We then ordered the runs by decreasing scores and identified the highest ranked run as the optimal ($\alpha$, $\beta$) pair for each topic difficulty class. We did this both for the two and the four classes of topic difficulty.

Table 3 shows the estimated optimal values for ($\alpha$, $\beta$) as measured by MAP, when using two or four classes of topic difficulty. Interestingly, with the four-class prediction the optimal parameter values for the Easy topics are ($\alpha = 0.0$, $\beta = 0.7$), i.e. the link score is ignored. For the Difficult topics the opposite effect is observed with the category score ignored and a high weight spread evenly on the link score $\alpha$ and the Zettair score (1 - $\alpha$ - $\beta$).

### 5.2 Evaluation of the predicted topic performance

We now use our combined topic difficulty prediction algorithm (described in Section 4) to tune and evaluate the performance of our XER system on the 35

[4] The value ten was determined experimentally on the XER training topic set.

**Table 4.** Evaluation of the predicted topic performance, as measured by MAP using the 35 topics of the INEX 2008 XER testing collection. The † symbol shows statistical significance over the Baseline run ($p < 0.05$).

<table>
<tr><th></th><th colspan="13">Class</th></tr>
<tr><th></th><th colspan="4">2</th><th></th><th colspan="8">4</th></tr>
<tr><th></th><th colspan="2">Easy</th><th colspan="2">Diff</th><th></th><th colspan="2">Easy</th><th colspan="2">modEasy</th><th colspan="2">modDiff</th><th colspan="2">Diff</th></tr>
<tr><th>Run</th><th>0.2</th><th>0.6</th><th>0.1</th><th>0.8</th><th></th><th>0.0</th><th>0.7</th><th>0.2</th><th>0.7</th><th>0.1</th><th>0.8</th><th>0.6</th><th>0.0</th></tr>
<tr><td><i>Baseline</i></td><td colspan="4">N/A</td><td>0.36280</td><td colspan="8">N/A</td></tr>
<tr><td><i>Predicted</i></td><td colspan="4"><b>0.38085</b></td><td></td><td colspan="8">0.30769</td></tr>
<tr><td><i>Optimal</i></td><td colspan="4">0.38705</td><td></td><td colspan="8">0.38431</td></tr>
<tr><td><i>Perfect</i></td><td colspan="4">N/A</td><td>0.45746†</td><td colspan="8">N/A</td></tr>
</table>

INEX 2008 XER testing topics. According to the estimated prediction, we aim at dynamically setting the $\alpha$ and $\beta$ parameters to their optimal values shown in Table 3. We did two sets of experiments, respectively with two and four classes of topic difficulty. We use MAP as our choice of evaluation measure.

To evaluate the benefits of using topic difficulty prediction, we compare the performances of four different runs:

1. *Baseline* run that does not use topic prediction with parameter vales set to ($\alpha$=0.2, $\beta$=0.6). This was the best performing entity ranking run at INEX 2007 (for the list completion task) when using the MAP measure.
2. *Predicted* run with parameter values set according to the estimated topic difficulty prediction on the training collection. The difficulty of a particular INEX 2008 XER testing topic was first predicted by our topic prediction algorithm, and the system parameter values were then set to the estimated optimal values for that topic difficulty class (as shown in Table 3).
3. *Optimal* run with parameter values set to the estimated optimal values on the training collection. Given the previously determined difficulty of a particular INEX 2008 XER testing topic (by applying the AAP measure on all the INEX 2008 submitted runs), the system parameter values were set to the estimated optimal values for that topic difficulty class. This is the best run we could aim at by using our topic difficulty prediction algorithm.
4. *Perfect* run with parameter values set to the best values (out of all the 66 value combinations) that can be achieved for each topic on the INEX 2008 XER testing collection. This is the run that produces the absolute best performance with our current XER system.

The results are presented in Table 4. The table shows that a two-class prediction of topic difficulty is performing better than the baseline (our last year best run), although the difference in performance is not statistically significant. These two runs were among the top four best performing runs at INEX 2008, all of which were submitted by our participating group. On the other hand, the four-class prediction of topic difficulty resulted in decreased performance, which is mainly due to the fact that the topic prediction algorithm is specifically designed for two-class rather than for four-class prediction. Although the results are promising, we recognise that the small size of the training and testing topic sets do not allow for very conclusive evaluation.

## 6 Related work

The approaches to query prediction can generally be grouped into two types: *static prediction* approaches, based on intrinsic characteristics of the query and possibly the document collection [8]; and *dynamic prediction* approaches, which use characteristics of the top ranked answers to the query [23].

Hao Lang et al. [11] evaluate query performance based on the covering topic score that measures how well the topic of the query is covered by documents retrieved by the system (dynamic prediction). Cronen-Townsend et al. [3] propose to predict query performance by computing the relative entropy (clarity score) between a query language model and the corresponding collection language model (static prediction).

Mothe and Tanguy [15] predict query difficulty based on linguistic features, using TreeTagger for part-of-speech tagging and other natural language processing tools. Topic features include morphological features (number of words, average of proper nouns, average number of numeral values), syntactical features (average conjunctions and prepositions, average syntactic depth and link span) or semantic features (average polysemy value). They found that the only positively correlated feature is the number of proper nouns, although the average syntactic link span and the average polysemy value also have some correlation with topic difficulty. We use some morphological or syntactic features in our topic prediction algorithm, but we also take advantage of the structure of the topic (title, description, narrative).

Kwok [10] uses Support Vector Machine (SVM) regression to predict the weakest and strongest queries in the TREC 2004 topic set (static prediction). Their choice of features include inverse document frequency of query terms and average term frequency in the collection. They found that features based on term frequencies could predict correctly even with short queries.

Yom-Tov et al. [22] predict query difficulty by measuring the contribution of closely related query terms, using features such as the overlap between the k-top answers to a sub-query (a query based on one query term) and to the full query. They experimented with two different query predictors: an histogram-based predictor and a modified decision tree. The difficulty predictor was used for selective query expansion or reduction.

Grivolla et al. [7] propose several classifiers to predict easy and difficult queries. They use decision tree and SVM types of classifiers, and select useful features among a set of candidates; the classifiers are trained on the TREC 8 test collection. The features are computed from the query itself (static features), the retrieval results and the knowledge about the retrieval process (dynamic features). They tested many different classifiers but did not combine them. Our approach is very similar to theirs, although we use different topic-specific features, a combined classifier and different test collection (INEX instead of TREC).

## 7 Conclusion and future work

We have presented our experiments in predicting topic difficulty and its application to the system we have developed for XML entity ranking. We demonstrated that it is possible to predict accurately a two-class level of topic difficulty with a classifier generated from a selected number of static features extracted from the INEX topic definition and the Wikipedia document collection. We also experimented with dynamic features from the intermediary results of the query processing but those features did not correlate well with the topic prediction. The more interesting result is related to the analysis of four classes of topic difficulty and their impact on the optimal parameter values for our XER system: for the Easy topics, the use of Wikipedia categories is very important while for the Difficult topics the link structure plays a very important role.

The application of topic prediction in tuning our system has shown encouraging improvement over our last year best result but we need a larger test collection to confirm the significance of our findings. The major limitation of our topic prediction approach is that it relies on the INEX topic definition that is much richer than standard Web queries. In the future we plan to develop a dynamic query prediction approach based (among other things) on the query similarity scores of the relevant entities retrieved by our XER system.

## Acknowledgements

Most of this work was completed while Vladimir Naumovski was doing his internship at INRIA in 2008.

## References

1. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
2. D. Carmel, E. Yom-Tov, and I. Soboroff. Predicting query difficulty - methods and applications. *SIGIR Forum*, 39(2):25–28, 2005.
3. S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *Proceedings of the 25th ACM SIGIR conference on Research and development in information retrieval (SIGIR'02)*, pages 299–306, Tampere, Finland, 2002.
4. A. P. de Vries, A.-M. Vercoustre, J. A. Thom, N. Craswel, and M. Lalmas. Overview of the INEX 2007 entity ranking track. In *Proceedings of the Sixth INEX workshop, Schloss Dagstuhl, Germany, 2007*, volume 4862 of *Lecture Notes in Computer Science*, pages 1–23, 2008.
5. G. Demartini, A. P. de Vries, T. Iofciu, and J. Zhu. Overview of the INEX 2008 entity ranking track. In *This volume*, 2009.
6. L. Denoyer and P. Gallinari. The Wikipedia XML corpus. *SIGIR Forum*, 40(1):64–69, 2006.

7. J. Grivolla, P. Jourlin, and R. de Mori. Automatic classification of queries by expected retrieval performance. In *Proceedings of the SIGIR workshop on predicting query difficulty*, Salvador, Brazil, 2005.
8. B. He and I. Ounis. Query performance prediction. *Information Systems*, 31(7):585–594, 2006.
9. J. M. Kleinberg. Authoritative sources in hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
10. K. Kwok. An attempt to identify weakest and strongest queries. In *Proceedings of the SIGIR workshop on predicting query difficulty*, Salvador, Brazil, 2005.
11. H. Lang, B. Wang, G. Jones, J.-T. Li, F. Ding, and Y.-X. Liu. Query performance prediction for information retrieval based on covering topic score. *Journal of Computer Science and technology*, 23(4):590–601, 2008.
12. E. Loper and S. Bird. NLTK: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics*, pages 63–70, Philadelphia, Pennsylvania, 2002.
13. S. Mizzaro. The good, the bad, the difficult, and the easy: Something wrong with information retrieval evaluation? In *Proceedings of the 30th European Conference on Information Retrieval (ECIR'08)*, volume 4956 of *Lecture Notes in Computer Science*, pages 642–646, 2008.
14. S. Mizzaro and S. Robertson. HITS hits TREC: Exploring IR evaluation results with network analysis. In *Proceedings of the 30th ACM SIGIR conference on Research and development in information retrieval (SIGIR'07)*, pages 479–486, Amsterdam, The Netherlands, 2007.
15. J. Mothe and L. Tanguy. Linguistic features to predict query difficulty. In *Proceedings of the SIGIR workshop on predicting query difficulty*, Salvador, Brazil, 2005.
16. J. Pehcevski, A.-M. Vercoustre, and J. A. Thom. Exploiting locality of Wikipedia links in entity ranking. In *Proceedings of the 30th European Conference on Information Retrieval (ECIR'08)*, volume 4956 of *Lecture Notes in Computer Science*, pages 258–269, 2008.
17. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
18. J. A. Thom, J. Pehcevski, and A.-M. Vercoustre. Use of Wikipedia categories in entity ranking. In *Proceedings of 12th Australasian Document Computing Symposium (ADCS'07)*, pages 56–63, Melbourne, Australia, 2007.
19. E. M. Voorhees. The TREC robust retrieval track. In *Proceedings of the Thirteenth Text Retrieval Conference (TREC 2004)*, 2004.
20. W. Webber, A. Moffat, and J. Zobel. Score standardization for inter-collection comparison of retrieval systems. In *Proceedings of the 31st ACM SIGIR conference on Research and development in information retrieval (SIGIR'08)*, pages 51–58, Singapore, 2008.
21. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques (2/E)*. Morgan Kaufmann, 2005.
22. E. Yom-Tov, S. Fine, D. Carmel, A. Darlow, and E. Amitay. Juru at TREC 2004: Experiments with prediction of query difficulty. In *Proceedings of the Thirteenth Text Retrieval Conference (TREC 2004)*, 2004.
23. Y. Zhou and W. B. Croft. Query performance prediction in web search environments. In *Proceedings of the 30th ACM SIGIR conference on Research and development in information retrieval (SIGIR'07)*, pages 543–550, Amsterdam, The Netherlands, 2007.