ReFER: effective Relevance Feedback for Entity Ranking

Tereza Iofciu[1], Gianluca Demartini[1],
Nick Craswell[2], and Arjen P. de Vries[3]

[1] L3S Research Center, Hannover, Germany
{iofciu,demartini}@L3S.de
[2] Microsoft Redmond, WA, USA
nickcr@microsoft.com
[3] Centrum Wiskunde & Informatica, Amsterdam, Netherland
arjen@acm.org

Abstract. Web search increasingly deals with structured data about people, places and things, their attributes and relationships. In such an environment an important sub-problem is matching a user's unstructured free-text query to a set of *relevant* entities. For example, a user might request 'Olympic host cities'. The most challenging general problem is to find relevant entities, of the correct type and characteristics, based on a free-text query that need not conform to any single ontology or category structure. This paper presents an entity ranking relevance feedback model, based on example entities specified by the user or on pseudo feedback. It employs the Wikipedia category structure, but augments that structure with 'smooth categories' to deal with the sparseness of the raw category information. Our experiments show the effectiveness of the proposed method, whether applied as a pseudo relevance feedback method or interactively with the user in the loop.

1 Introduction

Finding entities of different types is a challenging search task which goes beyond classic document retrieval and also single-type entity retrieval such as expert search [1]. The motivation for this task is that many 'real searches' are not looking for documents to learn about a topic, but really seek a list of specific entities: restaurants, countries, films, songs, etc. [10]. Example needs include 'Formula 1 drivers that won the Monaco Grand Prix', 'Female singer and songwriter born in Canada', 'Swiss cantons where they speak German', and 'Coldplay band members', just to name few.

This is a new interesting task that goes beyond standard search engine's matching between user query and document features. In the Entity Ranking (ER) scenario the user is looking for a set of entities of the same type with some common properties, e.g., 'countries where I can pay in Euro'. This query is answered by current web search engines with a list of pages on the topic 'Euro zone', or ways to pay in Euros, but not with a list of country names.

The complexity of this novel search task lays in the multi-step solution that should be adopted. Firstly, the system has to understand the user query, what is the entity type and which are its properties. Similarly to expert search, the index should contain entities instead of just documents, and the entity type should be represented in and matched against the user query. Therefore, several techniques from research fields such as Information Extraction and Natural Language Processing (NLP) could be used as well in order to first identify entities in a document collection. Moreover, a hierarchy of possible entity types and relations among entities and their types has to be considered [13, 16]. Initial attempts to ER have recently been presented. The main approaches build on top of the link structure in the set of entities [12], use passage retrieval techniques, language models [13], or NLP based solutions [6].

In this paper, we propose ReFER: a graph-based method to take advantage of relevance feedback (RFB) in entity retrieval, exploiting either example entities provided by the user, or the top-k results from an ER system. We show how the combination of RFB results with the initial system improves search effectiveness for all runs submitted to the Initiative for the Evaluation of XML Retrieval (INEX)[4] 2008 XML Entity Ranking track. The proposed method is designed based on the Wikipedia setting used at INEX but it could be adapted to other settings such as the one of tag recommendation (i.e., tagged web pages compared to Wikipedia articles belonging to Wikipedia categories).

The rest of this paper is structured as follows. We start with a summary of related work in the area of entity ranking. Section 3 then discusses the two main properties of Wikipedia and how these are exploited in this paper. The proposed algorithm is presented in Section 4. Section 5 then discusses our experimental results, showing how our graph-based method improves performance of previously proposed entity ranking techniques. The final Section summarizes our main conclusions.

2 Related Work

The first proposed approaches for ER [3–5] mainly focus on scaling efficiently on Web dimension datasets but not much on search quality while we aim to improve effectiveness of the ER task. Approaches for finding entities have also been developed in the Wikipedia context. Pehcevski et al. [12] use link information for improving effectiveness of ER in Wikipedia. Demartini et al. [6] improve ER effectiveness by leveraging on an ontology for refining the Wikipedia category hierarchy. Compared to these approaches, we propose an orthogonal view on the problem that can be applied to any ER approach via RFB. Balog et al. [2] propose a model that considers RFB techniques on the category structure of Wikipedia to improve effectiveness of ER. In this paper we show how our method can be effectively applied also on top of their system.

A different approach to the problem is to rank document passages that represent entities. In [20] the authors present an ER system that builds on top of

[4] `http://www.inex.otago.ac.nz/`

an entity extraction and semantic annotation step followed by running a passage retrieval system using appropriate approaches to re-rank entities. In [7] a similar approach is used to rank entities over time.

A related task is the *entity type ranking* defined in [17]. The goal is to retrieve the most important entity types for a query, e.g., Location, Date, and Organization for the query Australia. Our algorithm exploits entity type information from the entity-category graph in order to find the most important entity types. Another related task is *expert finding* which has been mainly studied in the context of the TREC Enterprise Track [1]. The entity type is fixed to people and the query is finding knowledgeable people about a given topic. Entity ranking goes beyond the single-typed entity retrieval and relevance is more loosely defined. More recently, at the TREC Entity track the task of finding entities related to a given one was studied [15]. As our approach works on the entity category structure, we focus on ER performed in the Wikipedia setting.

3 Category Expansion in Wikipedia

This paper presents an entity ranking model based on assigning entities to 'smooth categories'. This in turn is based on the Wikipedia link and category structure. This section describes the two key properties of Wikipedia we rely on to develop our model. The next section describes the smooth category model.

Wikipedia is a free encyclopedia with 2.7 million English articles written by volunteers.[5] It is a collaborative website with an editorial process governed by a series of policies and guidelines.[6] Wikipedia has two properties that make it particularly useful for ER. The first is that many of its articles are dedicated to an entity, so the entity ranking problem reduces to the problem of ranking such articles. The Wikipedia guidelines prescribe that an entity should have at most one article dedicated to it, according to the *content forking* guidelines. Thus the entity ranking model does not need to eliminate duplicates. Many real-world entities have no Wikipedia page, according to the *notability* guidelines. To be included, an entity should have significant coverage in multiple independent, reliable sources. For example, the model can rank major-league baseball players according to some entity-ranking query, but not players in youth baseball leagues, since youth players rarely meet the notability criteria.

In this setting, a simple ER solution is to rank Wikipedia pages in a standard IR system. If we search in a List Completion manner(i.e. query by example), for 'John F. Kennedy' in an index of Wikipedia pages, the top-ranked articles are: 'John F. Kennedy', 'John F. Kennedy International Airport', 'John F. Kennedy Jr.', 'John F. Kennedy Library' and 'John F. Kennedy assassination in popular culture'. The IR system has succeeded in finding pages relevant to the topic of JFK. However, if the information need were related to finding US presidents, the system has not succeeded. It did not find entities of a similar type. As a concluding remark, note, some articles do not pertain to an entity (e.g., 'Running'); we have to rely on the entity ranking model to avoid retrieving these.

[5] `http://en.wikipedia.org/wiki/Wikipedia`
[6] `http://en.wikipedia.org/wiki/Wikipedia:Policies_and_guidelines`

The second useful property of Wikipedia is its rich link and category structure, with the category structure being of particular interest when finding entities of similar type. Intuitively, one would say that if two entities are related by satisfying an information need, they should have at least one common category. The more common categories two entities belong to, the more related they are likely to be. The usefulness of Wikipedia's link structure has been confirmed in the INEX entity ranking experiments: participants found that category information, associations between entities and query-dependent link structure improved results over their baselines [18]. However, as Wikipedia is a collaborative effort, no strict rules enforce the guidelines for linking between entities or assigning entities to categories. Entities may belong to many categories describing its different aspects, and no limit exists on the number of categories an entity could get assigned. For example the Wikipedia page describing 'Albert Einstein' links to a wide variety of entities, including specific information such as 'Theory of relativity' and 'Nobel Prize in Physics', but also more generic facts like 'Germany' and 'Genius'. Considering the Wikipedia category structure, 'Albert Einstein' belongs to some sixty categories, varying from 'German Nobel laureates' and 'German immigrants to the United States' to '1879 births'.

The categories of a page are not weighted by their importance, so we do not know which is more important, and a page may also be missing from important categories. For example, in our snapshot of Wikipedia the article on South Korea is in the categories: 'South Korea', 'Liberal democracies' and 'Divided regions'. There are attributes of South Korea that are not described by categories.

4 Link-based Relevance Feedback for Entity Ranking

In this section we describe ReFER, our RFB algorithm based on the link structure of the Wikipedia model, and we then present ways of integrating it with existing ER systems.

In our model we assume a collection of categories $C = \{c_1, .., c_n\}$ and a collection of entities $E = \{e_1, .., e_m\}$. Each entity e_i corresponds to a Wikipedia page, and each category c_i is a tag describing an entity.

Definition 1. *An entity e_i is a tuple $\langle uri, desc, C_{e_i}, R_{e_i} \rangle$ where uri is the entity's identifier, desc is a string describing e_i (given by the Wikipedia article), $C_{e_i} \subseteq C$ is the set of categories describing the entity and $R_{e_i} \subseteq E \setminus \{e_i\}$ is the set of entities that are refered to from the Wikipedia url of e_i.*

One can easily see that given a collection of entities and categories, we can retrieve two types of connections. The first type is between two entities and we denote it with $link = \langle e_i, e_j \rangle$. The second type of connection is between an entity and a category and we denote it with $edge = \langle e_i, c_j \rangle$. In addition we distinguish between two types of edges according to the process that created them. The 'hard' edges of entity e_i are the ones that can be directly generated using C_{e_i}, i.e., $C_H(e_i) = \{c | c \in C_{e_i}\}$. The 'smooth' edges can be inferred through the categories of the referred entities, i.e., $C_S(e_i) = \{c | c \in C_{e_j} \forall e_j \in R_{e_i}\}$.

4.1 The ReFER Algorithm

Our entity ranking algorithm can be described as propagation of weights through a directed acyclic graph. The graph has nodes in three layers: an 'input' layer of entities, an 'intermediate' layer of hard and smooth categories and a ranked 'output' layer of entities connected to the 'intermediate' categories. Weights propagate through graph and is proportional to the number of links, hard edges and smooth edges.

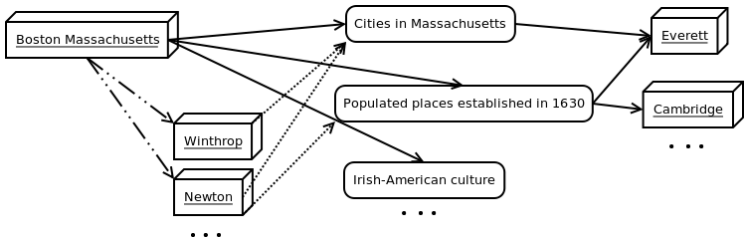

Fig. 1. Three layer graph, with input node entity 'Boston, Massachusetts'. Solid edges indicate hard categories, dashed edges indicate smooth categories.

For the example in Figure 1, if the article on *Boston Massachusetts* is in the category *Cities in Massachusetts*, and links to several pages that are also in that category, then the article's input node is connected to *Cities in Massachusetts* node via both a hard edge and a smooth edge. In our example, category *Cities in Massachusetts* will be weighted higher than category *Irish-American culture*, as the latter has no smooth edges leading to it. Smooth categories can add extra weight to hard categories, and also make associations with new categories. For 'South Korea', the original category that is most strongly supported is 'Liberal Democracies', since seven of the articles linked-to by the 'South Korea' article are 'Liberal Democracies'. The page is associated to 26 smooth categories, out of which 14 contain the word Korea. There is though some noise in the smooth categories, like 'Constitutional Monarchies' and 'History of Japan'. In order to reduce the amount of noisy smooth categories for an entity e_i we filter out the ones with less than 2 entities from R_{e_i} belonging to them.

Given a query q we activate a certain set of nodes E_q as input for our algorithm. Then for each category node in $C_H(e_i) \cup C_S(e_i)$, where $e_i \in E_q$, we sum the incident edge weights from active input nodes from E_q. For category c_j let us denote the total incoming hard-edge weight as h_{c_j} and smooth-edge weight as s_{c_j}. In our initial experiments, we noticed that the hard-category 'coordination' between the input nodes is important. If there is one category that is common to most of the active input nodes, then that category is extremely important, and should massively outweigh the other categories. This led us to develop the following exponential category weighting heuristic:

$$cw(c_j) = \frac{\alpha^{h_{c_j}} + s_{c_j}}{\log(catsize(c_{c_j}) + \beta)}, \tag{1}$$

where $catsize(c_j)$ is the number of Wikipedia pages in the category c_j and α and β are parameters[7], β being used so that the logarithm does not return negative values. The log down-weights very large categories, since these are unlikely to be discriminative. Akin to stopword removal, we eliminate categories with many entities (in our setup we considered a threshold of 1000 entities).

If there is a category that is common to all input nodes in E_q, then it will have high h and a much higher weight than any other category. For example, if the input nodes are a number of entities in the category *Cities in Massachusetts*, then that category will dominate the rest of the entity ranking process. If there is not a dominant category, then both hard and smooth categories come into play under this weighting scheme.

To rank entities, we propagate and sum the category weights to the output layer. The final entity ranking weight of output node e_k includes a popularity weight $P(e_k)$:

$$ew(e_k) = (\sum_{j=1}^{n} cw(c_j)) * P(e_k). \quad (2)$$

The popularity weight is based on the Wikipedia link graph where node e_k has indegree IN_k, such that $P(e_k) = min(\theta, \log(IN_k))$, θ being a parameter[8]. Static rank, a well-known concept from Web search, is a query-independent indicator that a certain search result is more likely to be relevant (see, for example, PageRank [11]). We found that connectivity in Wikipedia is an indicator that an entity is well-known, and therefore possibly a good search result.

4.2 ReFER Bootstrap and its Application to ER systems

The algorithm we propose is query independent as it just needs an initial set of entities where to start from. ER systems start from keyword queries provided by the user in order to generate a ranked list of results. We propose three ways of running our algorithm and combining it with existing ER systems.

In the first scenario the user provides also a small set of example relevant entities. We can use such set as the active nodes E_q from input layer I. We would thus obtain a ranked list of entities ordered by decreasing $ew(e_k)$ scores. It is then possible to merge, for example by means of a linear combination, the obtained ranking with one produced by an ER system which uses keywords provided by the user. In this paper we perform ranking combination in the following way[9]:

$$rank(e_k, q) := \lambda \cdot baseline(e_k, q) + (1 - \lambda) \cdot ReFER(e_k), \quad (3)$$

[7] Experimentally exploring the parameter space we obtained best results with $\alpha = 10$ and $\beta = 50$.

[8] Experimentally exploring the parameter space we obtained best results with $\theta = 5$.

[9] A different option would be to combine RSVs of the baseline ER system with $ew(e_k)$ scores. Due to the variety of approaches that lead to the scores in different ER systems, we could estimate such scores transforming the rank of entity e_k for query q; we carried out experiments computing the rank-based scores as $(1000 - rank)$ and $(1/rank)$. As the conclusions resulting from both transformations turned out identical we perform a simpler combination of ranks.

where $rank(e_k, q)$ is the new rank for entity e_k on query q, $\lambda \in [0, 1]$, $baseline(e_k, q)$ is the rank assigned by the baseline system, and $ReFER(e_k)$ is the rank assigned to e based on the scores computed by Formula 2.

A second approach would be to use results of an ER system in order to bootstrap our algorithm (i.e., as elements of the input layer). Thus, in a pseudo-RFB fashion, we consider top-k retrieved entities as being part of E_q. Again, in this way we would obtain a ranked list of entities by running the ReFER algorithm. We can now combine the two available rankings by, for example, in a linear combination.

A third approach, is the RFB one. After the ER system retrieves results for a query, the user selects relevant results present in top-k. We can use selected relevant results as elements of active input layer E_q. Again, we can combine the two rankings (the original one and the one generated based on Formula 2) by a linear combination.

5 Experimental Results

In this section we present an experimental evaluation of the proposed model for RFB in ER. We start describing the test collection we use and we then evaluate effectiveness of different applications to existing ER baseline systems.

5.1 Experimental Setting

The Entity Ranking track at INEX has developed a test collection based on Wikipedia. We perform our experiments on this test collection, for an objective and comparable evaluation. We will consider our RFB approach successful if it improves consistently upon the measured performance for most (or all) of the runs submitted to the track, essentially using the participant runs as baselines. This is an especially challenging goal in case of runs that already use the Wikipedia link structure between entities and/or categories.

The document collection used for evaluating our approach is the 2006 Wikipedia XML Corpus[8] containing 659,338 English Wikipedia articles. In INEX 2008, 35 topics have been selected and manually assessed by the participants[10]. An example of an INEX 2008 Entity Ranking Topic is presented in Table 1. The track distinguishes between the XML Entity Ranking (XER) and the List Completion (LC) tasks. In the XER task, participants use topic category and topic title; in the LC case, the example entities provided in the topics can be used by the system (and should not be presented in the results). Because the assessment pool has been created using stratified sampling, the evaluation metric used is xinfAP [19], an estimation of Average Precision (AP) for pools built with a stratified sampling approach. The ranked list of entities produced by ER systems is divided into disjoint contiguous subsets (strata) and then entities are randomly selected (sampled) from each stratum for relevance judgement. The metric xinfAP is then computed exploiting the estimation of Precision at each relevant document for each stratum.

[10] The test collection we used is available at: `http://www.L3S.de/~demartini/XER08/`.

Table 1. INEX Entity Ranking Topic example.

Title	Italian Nobel prize winners
Categories	#924: Nobel laureates
Examples	#176791: Dario Fo; #744909: Renato Dulbecco; #44932: Carlo Rubbia

5.2 Using Topic Examples

In order to evaluate the combination of ReFER with previously proposed ER systems, we decided to apply our algorithm to all the submitted runs at INEX 2008 as baselines as well as to the top performing runs of a later method tested on the same collection [2]. We then combine the results with baseline systems following Formula 3.

We performed such experiment with both XER and LC runs. The values of xinfAP for the original runs and the combination with the ReFER run are presented in Figure 2 for the XER task. The Figure shows how in all cases the

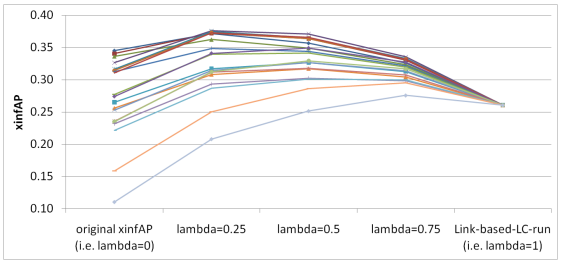

Fig. 2. Comparison of INEX 2008 XER runs merged with ReFER using topic examples.

combination of the baseline with ReFER improves the quality of the original ER system. For the runs where the initial baseline performs well (a high xinfAP), the best average value for lambda is close to 0.25 (i.e., giving more importance to the baseline). Baselines that did not perform that well require a higher λ of 0.75, giving more importance to ReFER results. For both tasks, the value of λ that yields best absolute improvement (i.e. 6.4% for XER and 5.2% for LC) is 0.5, so we present the following experiment results only for this combination strategy.

5.3 Content Based Pseudo Relevance Feedback

How does the ReFER approach perform as compared to standard content based pseudo-RFB? As we do not have access to the retrieval systems used to create the various runs, we implemented a system independent method. From each run we start from the top k retrieved results, from which we take top n common terms. The terms are ranked based on the cumulated TF-IDF score from the k documents. Next, we search with both the topic title and the top n common terms in our index of the INEX Wikipedia and retrieve ranked lists of results for each run. We then combine such result set with the corresponding original run by applying Formula 3 with $\lambda = 0.5$.

Experimental findings show that this method performed best on average when using top 5 common terms from top 10 retrieved documents. The maximum absolute improvement achieved by the content based approach is of 2% on average. Also, the content based method improved only 79% of the 19 runs.

5.4 Pseudo Relevance Feedback

Instead of using the example entities provided in the topic we can use top-k retrieved results from each run. In this way, we build a system that requires no user involvement, but that just builds on top of another method for ER.

For each query q we activate the k nodes in the input layer that correspond to the top-k retrieved results from the baseline run. Figure 3 (a) shows the xinfAP values for the original runs and for the combination (i.e., Formula 3 with $\lambda = 0.5$) with such pseudo-RFB run, for different values of k.

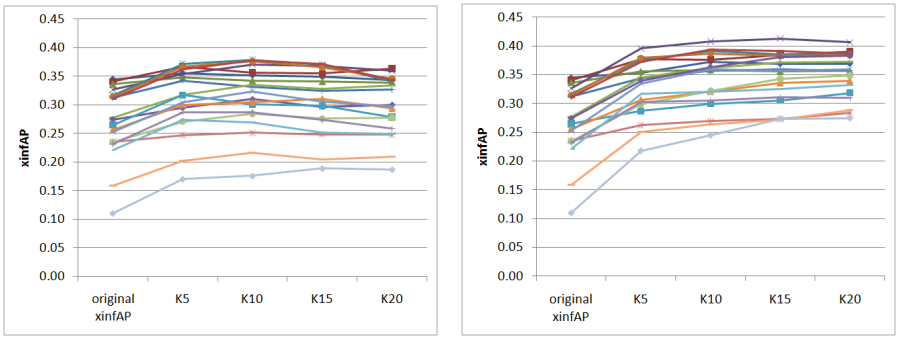

Fig. 3. Improvement of xinfAP for each run using all (a) or only relevant results (b) in top-k retrieved as seed of the algorithm, combining with $\lambda = 0.5$.

The effectiveness is always improved for each k. In Table 2 it is possible to see that, on average, $K = 10$ gives best improvement both for xinfAP and for the expected P@20 (as used in [19]). A t-test shows that the xinfAP improvement using $k = 10$ and $\lambda = 0.5$ over each baseline is statistical significant ($p \leq 0.05$) for all systems but one, where $p = 0.53$.

Table 2. xinfAP and expected P@20 measured for different values of k for pseudo-RFB and the relative improvement obtained by the combination over the original runs.

	xinfAP				expected P@20			
	K=5	K=10	K=15	K=20	K=5	K=10	K=15	K=20
Original	0.270	0.270	0.270	0.270	0.307	0.307	0.307	0.307
pseudo-RFB	0.266	0.275	0.267	0.256	0.284	0.290	0.277	0.269
Combination $\lambda = 0.5$	0.308	0.313	0.307	0.300	0.327	0.328	0.319	0.315
Relative Improvement	14%	16%	14%	11%	7%	7%	4%	3%

The results show how a small but effective seed leads to good results after applying the score propagation. When analysing the contribution of unique relevant results from the baseline and the pseudo-RFB we can see (Table 3) that most of the relevant results are present in both runs while only 4 relevant entities out of 21, on average, are not retrieved.

Table 3. Average unique contribution of relevant results (pseudo-RFB).

	K=5	K=10	K=15	K=20
Relevant in baseline	5.158	4.654	4.557	4.495
Relevant in pseudo-RFB	3.289	3.544	3.555	3.425
Relevant in both	10.694	11.198	11.296	11.358
Missed relevant	4.010	3.754	3.744	3.873

5.5 Relevance Feedback

In the next scenario we assume entity ranking in an interactive setting where the user can click on the relevant entities in the top-k results returned by the baseline system (i.e., RFB). Because assessing the relevance of entities returned can be considered to take a much lower effort than reading documents in a traditional information retrieval setting, we believe the ER setting justifies measuring the improvement in quality of the full displayed list (as opposed to the rank freezing or residual ranking methodologies that are more appropriate in the ad-hoc retrieval case [14]). When performing an entity retrieval task, the user's aim is not to read new relevant documents, but rather to obtain a precise and complete list of entities that answers the query. Thus, we use only relevant entities in top-k as seed to our algorithm. For xinfAP, it is possible to see how the algorithm obtains best performances with $k = 20$ (cf. Table 4).

Table 4. xinfAP and expected P@20 measured for different values of k for RFB and the relative improvement obtained by the combination over the original runs.

	xinfAP				expected P@20			
	K=5	K=10	K=15	K=20	K=5	K=10	K=15	K=20
Original	0.270	0.270	0.270	0.270	0.307	0.307	0.307	0.307
RFB	0.281	0.310	0.320	0.327	0.295	0.332	0.339	0.347
Combination $\lambda = 0.5$	0.327	0.341	0.347	0.350	0.386	0.382	0.380	0.381
Relative Improvement	21%	26%	29%	30%	26%	24%	24%	24%

If we compare Table 2 and Table 4 we can see that in the pseudo-RFB case, the best improvement is obtained using the first 10 retrieved results. In the RFB scenario, given that input entities are all relevant, the higher the value of k, the better the improvement. We did not study the effect of $k > 20$ because we do not expect a user to select relevant results lower than rank 20. A t-test confirms statistical significance ($p \leq 0.05$) of the improvement in xinfAP between the run using $k = 20$ and $\lambda = 0.5$ and each of the baselines.

If we analyze the contribution of unique relevant results from the baseline and the RFB results (Table 5) we see that the baseline contributes more than the pseudo-RFB part. Compared to the contribution of uniquely relevant entities in the pseudo-RFB scenario (see Table 3), we find however that blind feedback works better with respect to this aspect. This result can be explained by the fact that when considering system-topic pairs in almost 20% of the cases there are no relevant results in top-k retrieved results. There are only 7 topics for which all systems had relevant results in top 5 retrieved results. Thus in the RFB scenario we cannot apply our algorithm for all the system-topic pairs, whereas for pseudo-RFB the algorithm is applied also using only non-relevant entities.

Table 5. Average unique contribution of relevant results (RFB).

	K=5	K=10	K=15	K=20
Relevant in baseline	7.14	5.78	5.32	4.95
Relevant in RFB	2.02	2.65	2.96	3.11
Relevant in both	8.71	10.07	10.54	10.91
Missed relevant	5.28	4.65	4.34	4.19

5.6 Hard vs. Smooth Categories

What is the benefit of using hard and smooth categories? In order to observe the effect of using smoothed categories along with hard categories we experimented with various sets of categories both in the pseudo-RFB and RFB cases (see Table 6). We used as input nodes top k=10 retrieved results from the baseline (for the RFB case we only used the relevant from top 10 retrieved results, amounting to 3.63 results per topic). In both cases the use of smooth categories improves the overall performance of the analyzed systems. Furthermore, in the pseudo-RFB case, where also non-relevant entities are used as seed, the smoothed categories have a higher impact on the overall improvement.

Table 6. xinfAP measured for k=10 in the pseudo-RFB and RFB cases. The relative improvement obtained by the combination over the baseline is also shown.

	pseudo-RFB			RFB		
	C_H	C_S	$C_H \cup C_S$	C_H	C_S	$C_H \cup C_S$
Baseline	0.270	0.270	0.270	0.270	0.270	0.270
Pseudo-RFB/RFB	0.269	0.126	0.2753	0.306	0.097	0.310
Combination $\lambda = 0.5$	0.308	0.213	0.313	0.338	0.220	0.341
Relative Improvement	14%	-21%	16%	25%	-19%	26%

6 Conclusions and Further Work

Entity Ranking is a novel search task that goes over document search by finding typed entities in a collection. The retrieved entities can be used, for example, for a better presentation of web search results. In this paper, we presented a model for RFB in the entity retrieval scenario. The proposed model is based on weight propagation in a directed acyclic graph that represents links between entity descriptions. We have used as experimental setting the Wikipedia as a repository of such entity descriptions and have evaluated our approach on the INEX 2008 benchmark.

We have used the submitted runs as baselines and have shown, firstly, that performing fusion with the result of our algorithm using relevant entity examples as initial seed always improves over the baseline effectiveness. We have also evaluated our algorithm using as seed the top-k retrieved results in a pseudo-RFB fashion. The experiments demonstrate that, while in all cases the baselines were improved, using top 10 results yields the best improvement. Finally, we have shown how an emulated interactive feedback session (by using only the relevant entities in the top-k retrieved results) leads to an even higher improvement when performing a fusion with the baseline (i.e., a 0.12 absolute improvement in xinfAP using the relevant entities encountered in top 20).

We conclude that the proposed approach can be easily applied to any ER system in order to improve search effectiveness, and that the model performs well on the test collection we used. A limitation of this work is the use of a single test collection. As future work, we aim at evaluating our approach on a different ER setting such as, for example, graph-based tag recommendation [9].

References

1. P. Bailey, N. Craswell, A. De Vries, and I. Soboroff. Overview of the TREC 2007 Enterprise Track. *Proceedings of TREC-2007, Gaithersburg, MD*, 2008.
2. K. Balog, M. Bron, and M. de Rijke. Category-based query modeling for entity search. In *ECIR*, pages 319–331, 2010.
3. H. Bast, A. Chitea, F. Suchanek, and I. Weber. Ester: efficient search on text, entities, and relations. In *SIGIR '07*, pages 671–678, New York, USA, 2007. ACM.
4. T. Cheng and K. Chang. Entity Search Engine: Towards Agile Best-Effort Information Integration over the Web. *CIDR2007*, pages 108–113, 2007.
5. T. Cheng, X. Yan, and K. C.-C. Chang. Entityrank: Searching entities directly and holistically. In *Proceedings of VLDB*, pages 387–398, 2007.
6. G. Demartini, C. Firan, T. Iofciu, R. Krestel, and W. Nejdl. Why finding entities in wikipedia is difficult, sometimes. *Information Retrieval*, 13(5):534–567, 2010.
7. G. Demartini, M. M. S. Missen, R. Blanco, and H. Zaragoza. Entity summarization of news articles. In *SIGIR*, pages 795–796, 2010.
8. L. Denoyer and P. Gallinari. The Wikipedia XML corpus. *ACM SIGIR Forum*, 40(1):64–69, 2006.
9. Z. Huang, W. Chung, T. Ong, and H. Chen. A graph-based recommender system for digital library. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 65–73. ACM New York, NY, USA, 2002.
10. R. Kumar and A. Tomkins. A characterization of online search behavior. *IEEE Data Eng. Bull.*, 2009.
11. L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. 1998.
12. J. Pehcevski, A.-M. Vercoustre, and J. A. Thom. Exploiting locality of wikipedia links in entity ranking. In *Proceedings of ECIR*, pages 258–269, 2008.
13. H. Rode, P. Serdyukov, and D. Hiemstra. Combining document- and paragraph-based entity ranking. In *SIGIR*, pages 851–852, 2008.
14. I. Ruthven and M. Lalmas. A Survey on the Use of Relevance Feedback for Information Access Systems. *Knowl. Eng. Rev.*, 18(2):95–145, 2003.
15. P. Serdyukov, K. Balog, P. Thomas, A. Vries, and T. Westerveld. Overview of the TREC 2009 Entity Track, 2009.
16. T. Tsikrika, P. Serdyukov, H. Rode, T. Westerveld, R. Aly, D. Hiemstra, and A. P. de Vries. Structured document retrieval, multimedia retrieval, and entity ranking using pf/tijah. In *INEX*, pages 306–320, 2007.
17. D. Vallet and H. Zaragoza. Inferring the most important types of a query: a semantic approach. In *SIGIR*, pages 857–858, 2008.
18. A. P. Vries, A.-M. Vercoustre, J. A. Thom, N. Craswell, and M. Lalmas. Overview of the INEX 2007 entity ranking track. In *INEX*.
19. E. Yilmaz, E. Kanoulas, and J. A. Aslam. A simple and efficient sampling method for estimating ap and ndcg. In *SIGIR*, pages 603–610, 2008.
20. H. Zaragoza, H. Rode, P. Mika, J. Atserias, M. Ciaramita, and G. Attardi. Ranking very many typed entities on wikipedia. In *CIKM*, pages 1015–1018, 2007.