Entity Ranking Using Click-Log Information

Davide Mottin*
University of Trento
mottin@disi.unitn.eu

Themis Palpanas
University of Trento
themis@disi.unitn.eu

Yannis Velegrakis
University of Trento
velgias@disi.unitn.eu

Abstract

Log information describing the items the users have selected from the set of answers a query engine returns to their queries constitute an excellent form of indirect user feedback that has been extensively used in the web to improve the effectiveness of search engines.

In this work we study how the logs can be exploited to improve the ranking of the results returned by an entity search engine. Entity search engines are becoming more and more popular as the web is changing from a web of documents into a "web of things". We show that entity search engines pose new challenges since their model is different than the one documents are based on. We present a novel framework for feature extraction that is based on the notions of entity matching and attribute frequencies. The extracted features are then used to train a ranking classifier.

We introduce different methods and metrics for ranking, we combine them with existing traditional techniques and we study their performance using real and synthetic data. The experiments show that our technique provides better results in terms of accuracy.

Keywords: Entity Ranking, Click logs, Feature extraction, Ranking metrics

*Corresponding author

1 Introduction

Existing web search engines have resorted to the exploitation of log files [28] in order to improve the accuracy and relevance of their results. Log files provide an excellent source of information not only about the queries the users are posing, but also the answers these users consider as correct. Information Retrieval [22] and machine learning techniques [25, 26, 27] have been extensively used to analyze the logs and build models that can predict the most prominent answers to a given query, which can in turn be promoted to higher positions in the answer set they belong. Since log files contain the list of results, the sessions and the clicked documents we refer to them as click-logs.

Recently, we are witnessing an increasing interest in entity-based search and retrieval [8]. This is a direct consequence of the observation that the web, modeled today as a collection of documents, is not accurately reflecting the cognitive model humans use when searching for some specific piece of information. Humans are not thinking in terms of documents, but in terms of entities that may be persons, locations, events, etc. This idea is gaining support in the web search community [14] and is also one of the basic principles of dataspaces [17]. Furthermore, the semantic web community is interested in building the infrastructure that will transform the current web from a web of documents into a web of objects, i.e., entities [38]. Entity search engines have already appeared both in industry [42, 14, 47, 45], and in academia [9, 33], with major industrial players like Microsoft's Bing [34, 11], Yahoo! [44, 40], and Google [37], indicating a strong interest.

Entity searching is fundamentally different from document searching [3, 7]. An entity is an artifact that models a real world object. It is typically uniquely identified, and has a set of characteristics modeled as attribute name-value pairs. As such, the attributes are defining the semantics of an object and have a much stronger relationship between them, than what keywords in a document have. A keyword in a document is typically an indication that the document is related to the keyword but is is not easy to understand how it is related. For instance, answering queries such as `location=Lombardia` in a document search engine could be wrongly interpreted as documents containing the word location and Lombardia and their synonyms. Although the search engines are now able to perform semantic matching they cannot interpret this query as a request of the user of a particular characteristic of an entity. Furthermore, a document may contain reference to different entities, thus, the flat vector representation that most search engines use to model documents does not suffice. An entity, on the other hand, uniquely identifies, and represents in a compact way, a very specific concept, e.g., a person, a place, or a fictional object.

A keyword query in an entity search engine is intended to specify the characteristics of an entity. The search engines employ various techniques for similarity searching to identify the entities that satisfy these characteristics as much as possible, and rank them accordingly [16]. However, this selection may not reflect what the majority of the users are looking for. Ideally, the search engine would take into consideration the user intentions and rank entities that have a high popularity in higher positions in the result list. This is why they tried to exploit other information such as query logs.

Although reranking approaches that use log analysis have been systematically studied in the context of document search engines, we claim that new techniques are needed in the context of entity search engines, because of the following reasons. First, documents are typically represented as a flat vector of keywords, while the different attributes of an entity have more specific semantics. For instance, matching the value of the attribute `name` for people plays a much more important role than matching other attributes like the `city` or the `country`. The above claim is also verified by the results of our experimental evaluation with real data from an entity search engine (discussed in Section 7). Second, in document search engines there is a set of relevant (i.e., correct) answers to a query. In contrast, there is only a single entity that correctly answers a query to an entity search engine. We do not consider retrieval queries, i.e., queries of the form "Give me all the persons in London".

In this work, we present a novel approach for reranking results of *entity-based* search engines. In this way, the proposed method improves the entity identification. Our approach is based on the implicit feedback extracted by analyzing the logs, and on machine learning techniques [13, 51]. To the best of our knowledge, this is the first approach that has specifically been designed to solve the problem of reranking entities. In contrast to existing log analysis techniques that typically promote popular elements within the results, a major novelty of our approach is that we go further and we can make predictions even for undisclosed queries, by trying to reason about the motivations that a user has selected an entity in an answer set. Furthermore, our approach is independent of the matching technique used by the entity search engine, which makes it applicable in almost every situation.

More specifically, we make the following contributions:

- we propose a novel method to solve the *entity* reranking problem capturing interdependencies between entities in a result set alongside any relationships among query keywords and entity attributes;
- we introduce effective approaches for extracting implicit user feedback from the query logs of an entity search engine;
- we provide a new metric for evaluating the accuracy of the results with respect to the most selected entities and the ranking of the entities in a result set; and
- we experimentally validate the proposed approach using real datasets.

The remainder of the document is structured as follows: Section 2 provides a motivating example that illustrates our goals. Section 3 makes an overview of the scientific literature on topics related to our work and highlight our differences. Section 4 provides a set of preliminary definitions and explanations, alongside a formal definition of the problem we try to solve. Section 5 describes our solution, and Section 6 introduces evaluation metrics suitable for

e_1:
Name: Milan
Latitude: 45.8534
Longitude: 10.875
Country: Italy

e_2:
Name: Milan
ANSIname: Milan
Latitude: 45.4773
Longitude: 9.1815
Feature: populated place

e_3:
Name: Milan
ANSIname: Milan
Latitude: 45.8534
Longitude: 10.875
Feature: 1st-order admin. division seat

e_4:
Name: Univ. of Milan
Homepage: www.unimi.it
City: Milano
Country: Italy

e_5:
Name: Univ. of Milan
Tag: university
Domain: university
First_sentence: The University of Milan is a university located in Milan, Italy
OfficialName: Università di Milano

Figure 1: An answer set to the query `Milan`.

our problem. Section 7 presents our experimental results. Concluding remarks and possible future directions can be found in Section 8.

2 Motivating example

Consider a user that submits the query `Milan` to an entity search engine. The engine returns the five entities illustrated in Figure 1. The first entity represents the province of Milan, the second its capital city, the third the administrative division of Milan, the fourth the web page of the University of Milan, and the fifth the actual University of Milan. The order in which the results are returned reflects some internal algorithm of the search engine that we consider as a black-box, since we do not care about it.

Assume now that by analyzing the log files, we observe that most of the times that the query `Milan` was issued the users selected entity e_2, some number of times e_3 was selected, and a few times e_1 was selected, while entities e_4 and e_5 were never selected. Based on this information, it would have been desired to rerank the result set by placing e_2 first, then e_3 and then e_1.

What we are interested in, however, are the reasons that a user is actually recognizing a specific entity and selects it. An entity is characterized by its attributes. Thus, the attributes are those that allow the user to differentiate among the entities in the result set. In our particular example of Figure 1, one can see that all the entities have an attribute name with a value that matches the query keyword, i.e., the value `Milan` This means that the attribute name is not actually a distinguishing feature. The attributes longitude and latitude, on the other hand, appear in all the three entities e_1, e_2 and e_3 that have been selected at least one, but do not appear in e_4 or e_5 that were never selected.

This means that the values of these two attribute may play some important role in the user's decision to select an entity from the answer set of the specific query. The attribute `country:Italy`, on the other hand, does not match any query keyword and it appears in both selected and unselected entities, thus, it is not clear how much it can help as a discriminating feature.

Consider now the case in which the query `Milan Province` is executed on the search engine for the first time. This means that the logs have actually no information about any user intentions for this particular query. One could use information retrieval techniques and count the number of times each attribute is selected given a keyword. Unfortunately, this approach assumes attributes (and as a consequence, entities) as independent, ignoring valuable information about interconnections that entities may have based on their attributes and the user selections. The existing logs for instance, as previously explained, indicate that for the keyword Milan the attributes `latitude`, `longitude` and `feature` play some important role. This information can be used to rerank the results of the keyword query `Milan Province`.

Our challenging task is to develop a mechanism that, by analyzing the logs, can build a model for reranking entities in the result set by taking into consideration not only the popularity of attributes (as most information retrieval techniques do), but also the interdependencies among the entities, and which can actually use that knowledge even for queries not previously seen.

3 Related work

Ranking is a broad field of research in databases, information retrieval and the web. It deals with methods of sorting the query results in a way that the highest in the list is the one that more likely is the one for which the user was seeking. Using click-log information to improve ranking is a well studied problem [25] since it provides an implicit feedback on the accuracy of the results. They typically use machine learning techniques to learn a click model or develop a total order function that agrees with the user preferences. RankNet [5] is a neural network for learning the ranking function using gradient descent. Cohen et al. [13] proposed the Rank SVM method and demonstrated that the problem is NP-complete. The major challenge in that work was to find a consistent set of features that represents documents in a proper way. A similar challenge is present in our setting and is one of the main focuses of our work, but in the context of entities. Many of these approaches require a consistent training-set of labeled instances. [51] proposed a method that uses click-logs as implicit preference feedback (relative judgments), and for a large set of click-logs, it can have similar success to the methods that use explicit labeling. Another boosting algorithm has been described in [50], which is easier to train and implement, and outperforms traditional ranking methods such as the Ranking SVMs [13], the BM25 [36] and the RankBoost [19].

A number of extensions to the above idea have been proposed. One is to use query sessions to improve accuracy [35]. Some have claimed that the ranking is tightly coupled with the search method, but most search methods are based on the relationship between the query and each document. This leads to what is called, a *local ranking*. Considering the relationship among the individual documents during search and retrieval leads to what is called a *global ranking* [23]. Our work is independent of the search method and utilizes the logs and the attributes of the entities in the result set to understand how they relate to each other, leading that way into a global ranking. As a different perspective, clustering has also been proposed to be applied on the click-logs [48], which can lead into an identification of users or query categories and further assist the search.

Another group of studies [22] involves the construction of user click models that abstract the sequence of user selections in a web search. It introduces a probability framework based on a Bayesian approach. The results show that the method outperforms traditional models, such as the User Browsing Model [18] and the Dependent Click Model [21].

The majority of the above works are centered around document ranking and have not been studied in the context of entities. Our setting can be more effective when applied on entity search engines by exploiting their characteristic feature of consisting of attributes describing their properties. This allows us to reason about names and values of these attributes, as well as their relationships to the query and, last but not least, across entities. A document contains a set of entities and so document ranking techniques intrinsically computes a matching between the query and the words. We demonstrate that with entities there exists better matching schemes that allow us to connect similar entities and boost only the most relevant for the user.

In the area of Top-k query answering, finding and ranking entities within a database has been studied [6], but on databases with fixed schemas, thus, the kinds of attributes that the entities may have is limited. Text Retrieval [43] and XML Retrieval [46] are two initiative for the discovering and ranking entities on the web. Our method can be used on top of these techniques in order to exploit log information and improve the ranking.

Certain IR techniques have been proposed to improve entity searching and ranking [1] using click-logs. The main difference from our work is that entities there are web documents and relationships among them are explicit through hyperlinks. The EntityRank system [10], although it has goals similar to ours, considers entities to be any recognizable unit within a document, and not a structure containing a set of attributes. Click-log information has been used also for ranking XML documents [30], however, the queries there are structured queries, thus understanding the intended query semantics is not as challenging as ours. Apart from the click logs, others [15] have also considered the use of external knowledge, i.e., knowledge bases, to improve entity ranking.

4 Problem statement

We assume the existence of an infinite set of identifiers $\mathcal{O}$, an infinite set of names $\mathcal{N}$ and an infinite set of atomic values $\mathcal{V}$. Let $\mathcal{A}$=$\mathcal{N}\times\mathcal{V}$ be the set of all possible attributes. An entity is a pair $\langle i, A\rangle$ where $i\in\mathcal{O}$ and $A\subseteq\mathcal{A}$ and finite. We use $\mathcal{E}$ to refer to the infinite set of all possible entities.

A *query* is a finite set $\langle k_1, \ldots, k_n\rangle$ with $k_i\in\mathcal{N}\cup\mathcal{V}$ for i=1...n. A ranking function, denoted as $\preceq$ is a total order on the elements of $\mathcal{E}$. A list $L\subseteq\mathcal{E}$ is ranked according to a ranking function $\preceq_r$, denoted as $L^{\preceq_r}$, if pos(e_i)$\leq$pos(e_j) if and only if $e_i\preceq_r e_j$, where e_i,$e_j\in L$ and pos(e) denotes the position of entity e in the list L.

A log entry is a triple $\langle q, L^{\preceq_e}, e\rangle$, where q is a query, $L^{\preceq_e}$ is a list of entities ranked according to a ranking function $\preceq_e$ specified by the search engine, and $e\in L^{\preceq_e}$ the entity that the user chose being the one he or she was looking for when posing the query q. We stress the fact that e is the *single* correct answer to q.

We will use the notation $\mathcal{L}_q$ to refer to all the log entries in a log that have the same query q.

Problem. Given a set of log entries $\mathcal{L}$, we need to build a model that will allow us at run time, for a given query q, with $L^{\preceq_e}$ being the list returned by the search engine for that query, to construct a ranked list $L^{\preceq_r}$ of entities such that the entities in $L^{\preceq_r}$ are those in $L^{\preceq_e}$ and a ranking-quality f is maximized.

Note that the above problem definition is general, and does not assume any particular ranking-quality function f. In this study, the objective is to rank the correct answer for each query and user (remember that different users may look for different answers when issuing the same query) as close as possible to the top position.

5 Proposed approach

As a first step the log entries need to be converted into some mathematical structure that can be consumed. Many approaches typically eliminate log entries that fall outside a certain distribution [51]. This significantly reduces the size of the logs and eliminates noisy data. We chose not to do so in order to be able to test our approaches with the actual logs that typically include noise.

We represent the information from the logs to a given query q through a vector. In particular, for a given answer to a query q, we construct a vector whose length is the same as the number of entities in its answer set $L^{\preceq_e}$. Recall that for a given query q the list $L^{\preceq_e}$ is always the same. Different techniques can be used to populate such a vector, the difficult part is to find meaningful representation of the knowledge. One technique is to set the ith element of the vector to value 1 if for the ith entity e_i in $L^{\preceq_e}$, there is a log entry $\langle q, L^{\preceq_e}, e_i\rangle$, i.e., the entity e_i has been selected at least once. All the remaining entries in the vector are set to 0. We refer to this technique as SEL. The SEL technique is easily implementable but does not offer a fine-grained distinction based on the number of times an entity has been selected as opposed to the other entities in the result set.

A second technique that achieves such a distinction is to set the value of each position in the vector to the selected probability $P(C_i = 1)$, i.e., the probability that the respective entity in the result list has been selected. Such a probability is easily calculated by the log entries. In particular, for a given query q with an answer set $L^{\preceq_e}$, the ith position of the vector is assigned the value $\frac{c_{e_i}}{c_{total}}$, where e_i is the element in the ith position in the list $L^{\preceq_e}$, c_{e_i} is the number of log entries that are about the query q and for which the element e_i is the one selected, while c_{total} is the number of log entries related to the query q in general. We refer to this technique as the SELPROB.

Finally, a third technique is to set to 1 only the vector value that corresponds to the *most* selected entity. We refer to this technique as the SEL1. All three feedback extraction techniques are illustrated in Figure 2.

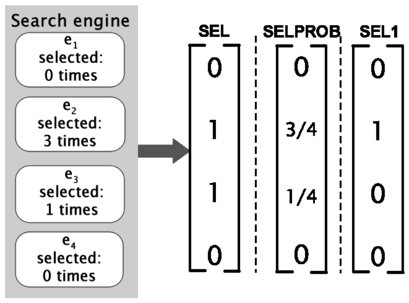


Figure 2: Feedback extraction methods.

We use a classifier in order to model the user behavior. To use a classifier we need to represent the information in the logs through features. Before describing these features, let us make a short introduction to the classifier. We use SVMRank [25], but other classifiers can also be used. SVMRank is a special SVM that learns ranking functions.

Given the results of the query engine and the logs, we need to quantify how related an entity e is to a query q. We denote this relevance as $rel(e, q)$, and compute it as

$$rel(e, q) = w \cdot \Phi(e, q)$$

where w is a vector and Φ is a function generating a feature vector. SVM will learn the vector w, subject to certain constraints. In particular, if $e_j \preceq e_i$, then it should hold that $rel(e_j, q) \leq rel(e_i, q)$, which means that

$$w \cdot \Phi(e_j, q) < w \cdot \Phi(e_i, q)$$

To allow some of the preference constraints to be violated, as is typically done in SVM classifications, a margin and non-negative slack variables are added.

This brings the preference constraint to the form

$$w \cdot \Phi(e_i, q) \geq w \cdot \Phi(e_j, q) + 1 - \xi_{ij}$$

Although we cannot efficiently find a w that minimizes the number of violated constraints, we can minimize the upper bound for it, i.e., $\sum \xi_{ij}$.

To maximize the SVM margin the following convex quadratic optimization problem needs to be solved:

$\min_{w,\xi_{ij}} \frac{1}{2} w \cdot w + C \sum_{ij} \xi_{ij}$
subject to: $w \cdot \Phi(e_i, q) \geq w \cdot \Phi(e_j, q) + 1 - \xi_{ij}$
and $\xi_{ij} \geq 0$, $\forall q, i, j$

Our goal is to propose a proper feature vector Φ. To improve the performance of SVM, the feature vectors are normalized to a vector $\hat{\Phi}$ with Euclidean distance 1 using the formula:

$$\hat{\Phi} = \frac{\Phi}{\|\Phi\|}$$

In existing work [23, 51, 35] related to document retrieval, a feature vector Φ, encapsulating the information about the query and the document characteristics, is typically composed by the concatenation of three kinds of smaller vectors. The first is a vector describing features of the query only independently of the documents. The second kind is a vector describing features of the document, independently of what the query is, and finally, the third kind is a vector consisting of document features related to the query.

Unfortunately, such a model is not sufficient for our case, where it is desired to represent relationships among the entity attributes, needed to model the relationship across entities in the same query set, and among entity attributes and the query, needed to model the importance of each attribute for the given query. Thus, we present three new ways of constructing the feature vectors for our case.

5.1 Entity Set Comparison

The first method is based on the idea that entities can be seen as sets of attribute name-value pairs. Doing so, we can exploit set properties to model the relationship between the query terms and the attributes of the entities in the same result list, as well as the relationship among the attributes of these entities. These sets are then transformed into features. We create a feature vector for every query q in the logs.

Since an entity attribute consists of a name and a value, a keyword in a query may match the name or the value. Assume a query $q = \langle k_1, k_2, \ldots k_n \rangle$, and $L^{\preceq_e}$ the list of elements returned by the query engine. We define the *query matching* set for every entity $e \in L^{\preceq_e}$ as:

$$QM_e^q = \{(n, v) | n : v \ \mathit{attribute\ of}\ e \wedge v \simeq k \wedge k \in q\}$$

To compare the attributes of an entity to the attributes of all the other entities we need to consider how their names and values match. Given two attributes (n_1, v_1) and (n_2, v_2), we define four cases on the way they can be related:

1. **not matching**: $n_1 \simeq n_2$ and $v_1 \neq v_2$
2. **imperfect matching**: $n_1 \neq n_2$ and $v_1 \simeq v_2$
3. **perfect matching**: $n_1 \simeq n_2$ and $v_1 \simeq v_2$
4. **incomparable**: $n_1 \neq n_2$ and $v_1 \neq v_2$

Note that the equality between two values ($\simeq$) here is not necessarily strict equality. It may be substring matching, or any other form of matching desired. In all our experiments we employ the traditional JaroWinkler [49] matching scheme. The choice of the measure is dependent to the data. Thus we do not specifically address this problem in this work.

Based on the above, for each entity e in the answer set $L^{\preceq_e}$ of every query $q=\langle k_1, k_2, \ldots, k_n\rangle$ in the logs we further define the following four sets:

$$PM_e^q = \left\{ (n,v) | n \simeq n_i, v \simeq v_i \ s.t.\ (n_i, v_i) \in \bigcup_{e_j \in L_q, e_j \neq e} e_j \right\}$$

$$IM_e^q = \left\{ (n,v) | n \neq n_i, v \simeq v_i \ s.t.\ (n_i, v_i) \in \bigcup_{e_j \in L_q, e_j \neq e} e_j \right\}$$

$$NM_e^q = \left\{ (n,v) | n \simeq n_i, v \neq v_i \ s.t.\ (n_i, v_i) \in \bigcup_{e_j \in L_q, e_j \neq e} e_j \right\}$$

$$I_e^q = \left\{ (n,v) | n \neq n_i, v \neq v_i \ s.t.\ (n_i, v_i) \in \bigcup_{e_j \in L_q, e_j \neq e} e_j \right\}$$

Computing these sets requires $O(nv)$ time, where n is the number of different attribute names and v the number of different attribute values in the set of entities in the query result. However, experimental results demonstrate that, since $n \ll v$, the time to produces such features is reasonable. An important property of the above sets is the completeness of the knowledge expressed by them. In particular:

$$PM_e^q \cup IM_e^q \cup NM_e^q \cup I_e^q = e$$

Example 1. Suppose that a user poses the query `Milano` to the search engine and the search engine responds with a list that contains two entities. e_1:$(name : Milano, country : Italy, zipcode : 20121, population : 1321113)$ and e_2:$(name : Luca, lastname : Milano, country : Italy)$, from which the user selects the first one. The keyword Milano in the query matches only the attribute value of the name attribute in e_1 and of the lastname in e_2. Thus, $QM_{e_1}^q = \{(name, Milano)\}$ and $QM_{e_2}^q = \{(lastname, Milano)\}$. Furthermore, $PM_{e_1}^q = \{(country, Milano)\}$, $IM_{e_1}^q = \{(name, Milano)\}$, $NM_{e_1}^q =$

$\{(name, Milano)\}$ and $I^q_{e_1} = \{(zipcode, 20121), (population, 1321113)\}$.

We create a feature vector $\Phi_{e,q}$, for every pair of query q in the logs and entity e in the answer set $L^{\preceq_e}$ of the query q. Let n_1, …, n_k be the set of all the attribute names of the attributes in the entities within the whole result set $L^{\preceq_e}$. The vector $\Phi_{e,q}$ has the form:

$$\Phi_{e,q} = \begin{bmatrix} \phi(n_1, e, q) \\ \phi(n_2, e, q) \\ \vdots \\ \phi(n_k, e, q) \end{bmatrix}$$

Each of the $\phi(n, e, q)$ is a block of five single-value rows, constructed as follows:

$$\phi(n, e, q) = \begin{bmatrix} \mathbf{1}\ if\ (n, v) \in QM^q_e \textbf{ else 0} \\ \mathbf{1}\ if\ (n, v) \in PM^q_e \textbf{ else 0} \\ \mathbf{1}\ if\ (n, v) \in IM^q_e \textbf{ else 0} \\ \mathbf{1}\ if\ (n, v) \in NM^q_e \textbf{ else 0} \\ \mathbf{1}\ if\ (n, v) \in I^q_e \textbf{ else 0} \end{bmatrix}$$

Note that in the above specification of $\phi(n, e, q)$, the value v of the attribute can be whatever, i.e., it is enough one (any) attribute with the name n in the respective set to satisfy the condition.

We refer to this method of generating the feature vector as *Full*. As an alternative, one can ignore the name matching and produce only three sets out of the five mentioned earlier. These three sets will be the QM^q_e as before, but instead of the four others, one set $M^q_e = PM^q_e \cup IM^q_e$ is created for modeling the matching values, and one $NN^q_e = NM^q_e \cup I^q_e$ for those not matching. We refer to this approach as *Simple*.

5.2 Entity Centric IR (ECIR)

Instead of using features purely based on the entity structure as was previously described, one can use traditional IR techniques but adapted to the case of entities. In particular, in each entity we consider what is called the *title* part, which are attributes of major identification importance, like name, title, last name, etc [4]. We also consider the entity *content* as the set of all the values of its attributes, and the entity as a *whole*, consisting of all the attributes names and values. Having done so, traditional IR techniques [31] can be applied to analyze the logs and build a feature vector.

The feature vectors consist of features in three main categories, 24 features in total. The first category is about *query* features and includes the query length, the number of query keywords, the IDF feature values, the IDF titles and the IDF feature names and values. The second category is about *entity* features and includes the number of words in the values, the number of words in titles, the number of words in attribute names and values, the number of attributes, the number of numeric attributes, whether the entity is a person, whether it is a an organization, a place or something else [4]. The last category contains features related to the relationship between the entity and the query. These features are the TF-IDF score, the BM25 score and the sum of TF for the values [32], the titles, and the remaining attributes as a whole. All the features are summarized in Table 1.

Query features	Entity features	Query-entity features
query length	number of words titles	tf-idf titles
number of keywords	word number content	tf-idf content
average keywords length	word number whole entity	tf-idf whole entity
idf titles	attribute number	BM25 titles
idf content	numeric attributes number	BM25 content
idf whole entity	is a person	BM25 whole entity
	is an organization	sum of tf titles
	is a place	sum of tf content
	is other	sum of tf whole

Table 1: Feature descriptions for *ECIR* method.

5.3 Induced popularity (IP)

A third alternative is to measure the popularity of each attribute by analyzing the logs and then using this information as a base for the entity ranking. For instance, consider two entities that represent two persons, both called Jordan, but the first is a basketball player while the second is a clerk. Knowing that the first entity is more popular, one can conclude that the `job=basketball player` is more popular than the `job=clerk`. We refer to this idea as *induced popularity*.

To build features out of this notion one can select a threshold and look at the number of popular attributes above a specific threshold T, characterizing the number of times that an attribute has to be selected in order to be considered popular. In this way an attribute is defined as popular if in the logs it has been selected at least T times.

The number of popular attributes following this diverse notion of popularity is the value of the feature we add to the example set in the classifier. We propose two different configurations: the first is taking into account only selected entities, while the second considers also attributes of non-selected entities. In the experiments, we refer to them as *SIP* (Selected Induced Popularity) and *N/SIP* (Non-Selected / Selected Induced Popularity), and we consider four different thresholds (3, 5, 7, and 9).

5.4 Traditional IR

Apart from the above three methods that we propose, the entities can be seen as documents, hence, traditional IR techniques can be used. Of course, not all the IR techniques are applicable. For instance, the page-rank or other URL-related features are not applicable. However, the vast majority of the existing techniques could be used. More specifically, we implemented the features described in Microsoft Learning to Rank benchmark [31]. From that list we extract a total of 23 features.

6 Evaluation Measures

6.1 IR Measures

There is a number of different document ranking evaluation techniques. Among them, the MAP and the NDCG are widely used [2, 29, 50].

The NDCG measures the distance between a re-ranking and an optimal ranking obtained by reordering the results according to relevance judgment that is provided as ground truth. Evidently, NDCG cannot be directly applied in our setting, where we only have a single correct answer (not a set of relevant answers with a varying degree of relevance).

The other classical metric in the document ranking literature is MAP. In that context, results are classified as relevant or irrelevant (with respect to the given query), and the final ranking is considered good when all relevant results are in the top positions, irrespective of the relative order of these relevant results.

MAP is defined as the mean of the average precision scores for a query. Let $L^{\preceq_r}$ be a ranked list of entities, and $R \subseteq L^{\preceq_r}$ the subset of entities which are relevant. The average precision score is defined as:

$$MAP(L^{\preceq_r}) = \frac{1}{|R|} \sum_{e_i \in R} Prec(pos(e_i)), \tag{1}$$

where $Prec(i)$ is the number of relevant entities in the first i positions, divided by i. The final MAP score is obtained by summing together the MAP of each query divided by the number of queries.

Example 2. Consider an original ranking that includes entities e_1, selected once, and e_2, selected five times, and assume the two possible final rankings depicted below.

original ranking

1	2	3	
	$e_1 : 1$	$e_2 : 5$	...

final ranking A

1	2	
e_1	e_2	...

final ranking B

1	2	
e_2	e_1	...

In the case of final ranking A, MAP is 1 since the two relevant entities are in the top positions of the result list. However, our intuition says that this ranking is not perfect, because e_1 is way less preferred than e_2, and should therefore be ranked second. This is the situation in final ranking B, which is of higher quality. Nevertheless, the MAP value is the same as before.

6.2 Average Entity Precision (AEP)

The example above, illustrates the deficiency of MAP in our problem setting, i.e., the fact that it cannot differentiate among rankings that may have different quality, because it treats all relevant (i.e., selected) entities equally. In the problem we consider though, there is only one relevant result for each query.

In order to remedy this situation, we introduce AEP, which measures the mean precision among references of the same query. In doing so, AEP includes in the computation the user selections (i.e., how many times each entity is selected), and additionally uses the rank positions of the entities as weights. AEP is formally defined as follows.

$$AEP_q = \frac{1}{|\mathcal{L}_q|} \sum_{(q,L,e)\in\mathcal{L}_q} \frac{1}{\mathrm{pos}_L(e)}, \tag{2}$$

where $\mathrm{pos}_L(e)$ is a shorthand notation that represents the position of entity e in L.

Example 3. Continuing the previous example, the value of AEP for final ranking A is 0.58, while for B becomes 0.92. This result is intuitive, since it more accurately captures the user selections.

7 Experimental Evaluation

In this section, we present the results of the experimental evaluation of the proposed approach, compared to the original ranking and to traditional IR techniques used in the document re-ranking literature. In all cases, we evaluate the performance of the algorithms using 10-fold cross validation.

We used the SVMLight* implementation of SVMRank [24], which is a widely used ranking SVM classifier. Regarding the choice of a string similarity method (used by the *Simple* and *Full* methods), we tested several alternatives (perfect equality, Levenstein distance, SoftTFIDF, JaroWinkler) [12], and decided to use the JaroWinkler.

All algorithms were coded in Java 1.6, and run in i686 Intel(R) Xeon(R) CPU X3220 2.40GHz machine with 4Gb of main memory.

We have designed all experiments such that they are reproducible. To this end, we have built a webpage [41], which contains all datasets and code used in this work.

* `http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html`

Dataset	Log entries	Dist. queries	Avg sel. ent.	Max sel. ent.
D-Okkam	2,321	116	3.2	7
D-Top	3,934	2,199	1.8	5
D-Mid	3,791	2,272	1.7	4
D-Low	3,697	2,281	1.6	4

Table 2: Characteristics of the datasets.

7.1 Datasets

In the following paragraphs we describe in detail the datasets we used in our evaluation. Their characteristics are summarized in Table 2.

7.1.1 D-Okkam datasets

We obtained a real dataset, D-Okkam, by observing the behavior of users when interacting with the OKKAM [33] entity search engine. The system allows users to search for a named entity, and to select the correct entity from the result list. We logged the queries (drawn from a restricted list), search engine result lists, and selected entities for a period of one month. We gathered 2,321 log entries from 129 distinct users, corresponding to 116 different queries.

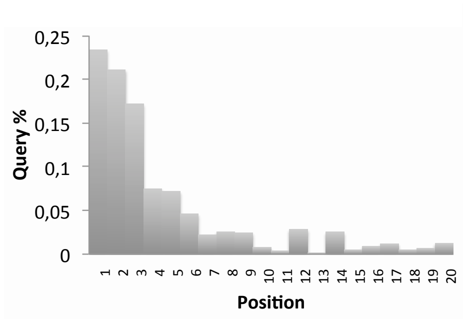
Figure 3(a) depicts the distribution of the position in the result list of the entities selected by the users. We note that for the majority of the queries different users were selecting different entities as the correct answer (indicating the different search goals of the users), with an average of 3 (max=7) selected entities per query.

7.1.2 D-Top, D-Mid, D-Low datasets

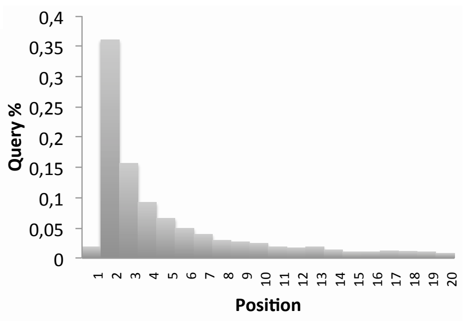
In order to evaluate our techniques on datasets with different characteristics, as well as to test their time performance, we generated datasets based on real data as follows.

We had in our disposal a query log from Microsoft MSN search engine, from which we randomly extracted 5,370 queries on distinct named entities. These queries were later manually matched against the entities stored in OKKAM, producing for each query a set of relevant entities that included the correct entities (remember that we may have more than one when aggregating the behavior of many users), as well. We removed queries that shared at least one selected entity, so as to be sure that in our experiments the classifier is properly generalizing and not simply learning how to rank the same set of entities. Then, we generated three datasets, D-Top, D-Mid and D-Low, by randomly shuffling the answer set of a query, thus, producing a corresponding result list that included the correct entities. These datasets measured 2,250 queries and 3,850 log entries on average.

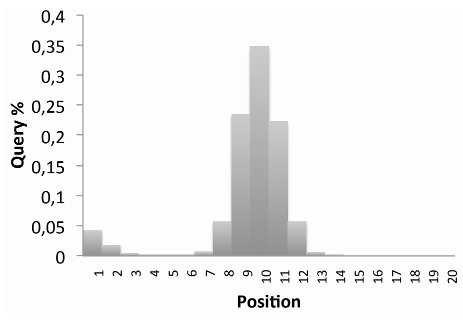
The D-Top dataset simulates a situation where the correct entities are already listed in the top positions of the result list, following a Zipfian distribution from position 2 (Figure 3(b)). This is the hardest case for our techniques, since there is little room for improving these result lists. The D-Mid and D-Low simulate the cases when most of the correct entities are listed in the middle (following a Gaussian with $\mu = 9$, $\sigma^2 = 1$) and bottom (following a Gaussian with $\mu = 19$, $\sigma^2 = 1$) positions of the result lists. The two distributions are depicted in Figures 3(c) and 3(d), respectively. (These graphs show that some correct entities are still ranked in the top positions: this happens when the result list only contains $2 - 3$ entities.) The D-Top, D-Mid and D-Low datasets, too, have an average of 2 (max=5) selected entities per query.



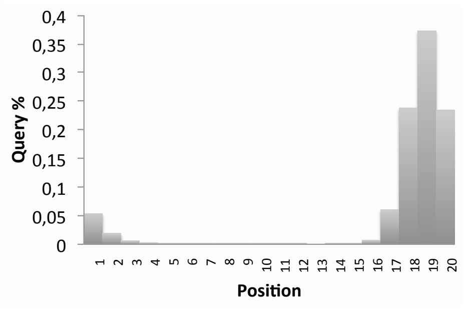
(a)



(b)



(c)



(d)

Figure 3: Click distribution of datasets (left to right): D-Okkam, D-Top, D-Mid, D-Low.

7.2 Ranking Quality

We evaluated our methods with the MAP and AEP measures, using different feedback extraction strategies and different sets of features for the classification model (refer to Section 5). We also propose and evaluate hybrid feature schemas that combine different techniques. The resulting feature vectors are obtained training the classifier on features computed using two different base-methods. In particular, we compare the following approaches (introduced in Section 5).

	Sel		**SelProb**	
	MAP	AEP	MAP	AEP
Original	0.398	0.333	0.398	0.333
Traditional IR	0.754	0.671	0.754	0.671
Simple	0.712	0.633	0.712	0.633
Full	0.712	0.633	0.712	0.633
ECIR	0.757	0.672	0.757	0.672
Simple + ECIR	0.717	0.630	0.717	0.630
Full + ECIR	0.717	0.630	0.717	0.630
ECIR + SIP	0.805	0.709	0.804	0.708
ECIR + N/SIP	0.806	0.709	0.806	0.710
Simple + SIP	**0.813**	0.716	0.811	0.715
Simple + N/SIP	**0.813**	**0.717**	0.811	0.715
Full + SIP	0.810	0.715	**0.815**	**0.718**
Full + N/SIP	**0.813**	0.716	0.811	0.715

Table 3: Results for D-Top.

Simple and *Full*, *ECIR*, and combinations of the above among themselves and with *IP* methods, namely *SIP* and *N/SIP*. We also compare against *Traditional IR*, a classifier using traditional IR features. The baseline is *Original*, which represents the original result list produced by the search engine.

The experimental results are grouped according to the method used. Due to lack of space, we do not report results for the Sel1 feedback extraction strategy, because it either leads to small improvements, or no improvements at all. Its low performance is explained by the fact that it contains partial information, only storing the single most selected correct entity and discarding all the rest, which in our case turns out to be a suboptimal strategy.

We first report results on the datasets shown in Tables 3, 4 and 5, for D-Top, D-Mid, and D-Low, respectively. We mark with bold the best improvement over *Original*. We also performed a two-tailed t-test to establish statistically significant values. The test showed that all measurements are statistically significant with a p-value ≤ 0.001.

The results show that in all cases the classifier produces accurate ranking functions, able to reorder the query results in a proper way. For all three datasets, the proposed approaches are able to correctly reorder the entity result lists, achieving improvements (over the original ranking) for both MAP and AEP of up to more than 4 times for D-Low. As expected, the improvements are less pronounced for the other two datasets, but there is still a significant improvement of up to more than 2 times for D-Top. When compared to *Traditional IR*, the proposed techniques achieve results that are 4% (D-Low) to 18% (D-Mid) better.

For these datasets, we observe that *ECIR* performs similarly to *Traditional IR*. In contrast, the IP-based features help the classifier achieve better performance, achieving the best results in combination with *Simple* and *Full*, closely followed by *ECIR*.

	Sel		**SelProb**	
	MAP	AEP	MAP	AEP
Original	0.214	0.176	0.214	0.176
Traditional IR	0.794	0.720	0.816	0.740
Simple	0.695	0.633	0.695	0.633
Full	0.695	0.633	0.695	0.633
ECIR	0.803	0.725	0.803	0.725
Simple + ECIR	0.717	0.641	0.717	0.641
Full + ECIR	0.717	0.641	0.717	0.641
ECIR + SIP	0.925	0.796	0.924	0.795
ECIR + N/SIP	0.923	0.795	0.924	0.796
Simple + SIP	**0.933**	**0.803**	0.931	**0.802**
Simple + N/SIP	0.932	0.802	**0.932**	**0.802**
Full + SIP	0.931	0.801	**0.932**	**0.802**
Full + N/SIP	0.930	0.801	0.931	0.801

Table 4: Results for D-Mid.

Table 6 summarizes the results for the D-Okkam dataset. In this table, we mark with an asterisk the values that are *not* statistically significant, i.e., having p-value > 0.05. By analyzing Table 6, we observe that the proposed approach results in improved rankings (overall). Moreover, we note that in some cases, these improvements are significant. In particular, we achieve the best performances with *Simple + N/SIP* and *Full + SIP*, with an improvement in AEP of 26% and 18%, respectively, with respect to *Original*. In general, Induced Popularity methods perform better than the rest. *Traditional IR* features, on the other hand, do not have a good performance: they worsen the results for MAP, and only modestly improve the AEP measure. Although *Simple* and *Full* methods, achieve the worse results in MAP, a more accurate analysis revealed that for small datasets like D-Okkam these methods tend to produce high variability in MAP. However, they achieve good performance in conjunction with other techniques such as Induced Popularity.

We notice that several of the MAP values are not statistically significant. A closer look at the data shows that there is a high variability in the observed results, which is not the case for AEP. The reason is that MAP does not distinguish among the selected entities (see discussion in Section 6), while AEP is a measure that focuses on the most selected of the correct entities, which the results demonstrate that are consistently pushed higher in the final ranking.

The results validate the observations we made with the bigger data (even though the improvements are smaller), that the hybrid techniques achieve the best results, especially those that use the IP-based features. Though, we note that in this dataset *Traditional IR* performs worse than *ECIR*, which is an indication that *Traditional IR* is less suitable for entity ranking than our proposed techniques. In summary, we can say that with the proposed approach the users are (on average) indeed experiencing a better performance of the search engine.

Overall, our experimental evaluation shows that the proposed techniques

	Sel		**SelProb**	
	MAP	AEP	MAP	AEP
Original	0.179	0.154	0.179	0.154
Traditional IR	0.914	0.794	0.914	0.794
Simple	0.682	0.630	0.682	0.630
Full	0.682	0.630	0.682	0.630
ECIR	0.907	0.789	0.907	0.789
Simple + ECIR	0.789	0.677	0.789	0.677
Full + ECIR	0.789	0.677	0.789	0.677
ECIR + SIP	0.947	0.818	0.947	0.818
ECIR + N/SIP	0.946	0.817	0.947	0.817
Simple + SIP	**0.950**	**0.821**	0.949	0.820
Simple + N/SIP	**0.950**	**0.821**	0.949	0.820
Full + SIP	**0.950**	**0.821**	**0.951**	**0.822**
Full + N/SIP	0.948	0.820	0.948	0.819

Table 5: Results for D-Low.

result in better performance when compared to the traditional IR approach, because they are in a position to exploit the unique features that entities have. The hybrid techniques that use the IP-based features achieve the best results, but when considering time performance (Section 7.4), as well, two of these hybrid techniques, *Simple+N/SIP* and *Full+SIP*, are the methods of choice. These two techniques achieve the best results (or are very close to the best) across different feedback methods and datasets for all the cases we evaluated in our experiments.

Regarding the feedback extraction method, it seems that there is significant difference between Sel and SelProb. We are currently collecting a larger real dataset, which will further help showcase the differences among the proposed techniques.

7.3 Classification Model

In this section, we make some observations on the classification model, based on our experimental results. As we discussed in Section 5, the classification model we use in this study is linear, and the model representation is a weight vector w. Features with positive weight affect the ranking in a positive way, while a negative weight means a negative impact of the feature for the final rank. This analysis refers to the D-Okkam dataset and SelProb feedback method (similar results hold for Sel), and in the next paragraph we briefly present some of the most important findings. Table 7 presents a summary of the main positive and negative feature weights for the most significant methods.

Our analysis shows that *Traditional IR* is mostly affected by the term frequency and the BM25, while *ECIR* is positively affected by the number of attributes and the number of characters of the entity. In both cases, the number of covered terms has a negative impact on the classifier.

	Sel		**SelProb**	
	MAP	AEP	MAP	AEP
Original	0.661	0.442	0.661	0.442
Traditional IR	0.641	0.478*	0.627	0.465*
Simple	0.580	0.505*	0.585	0.514*
Full	0.618	0.451*	0.623	0.438*
ECIR	0.677*	0.514	**0.701***	0.526
Simple + ECIR	**0.680***	0.458	0.680*	0.462
Full + ECIR	0.652*	0.498	0.680	0.462
ECIR + SIP	0.656*	0.486	0.659*	0.496
ECIR + N/SIP	0.662*	0.504	0.675*	0.525
Simple + SIP	0.665*	0.517	0.667*	0.543
Simple + N/SIP	0.677*	**0.555**	0.664*	0.543
Full + SIP	0.671*	0.520	0.668*	**0.547**
Full + N/SIP	0.668*	0.494	0.667*	0.545

Table 6: Results for D-Okkam.

Traditional IR	weight	Simple + ECIR	weight	Simple + N/SIP	weight
Normalized Min tf	0.48	Entity Size	4.41	Legal name QM	2.03
Max tf-idf	0.45	Keywords in attrs	2.53	Org. name QM	1.62
Covered term ratio	-1.19	Title length	-0.97	Middle name QM	-1.16
Covered term	-1.19	Number of # attrs	-1.58	Country QM	-1.35

ECIR	weight	ECIR + N/SIP	weight	Full + SIP	weight
Number of attrs	0.80	Popular attrs (thr 3)	0.39	Incountry QM	2.57
BM25 of values	0.46	BM25 titles	0.34	Legal name QM	2.12
Title length	-0.88	Title length	-0.39	Country QM	-1.08
Keywords in attrs	-1.85	Unpop. attrs (thr 3)	-0.53	Middle name QM	-1.12

Table 7: Main positive and negative feature weights.

In contrast, hybrid combinations of *ECIR* and the other methods are strongly affected by the size of the entity and the number of query keywords in the attribute values. Interestingly, the idf of the values and the number of numeric attributes have negative weights. In addition, it is worth noting that for *Full+ECIR* and *Simple+ECIR* the models are almost the same, and the set comparison features do not affect the classification task in a significant way. On the contrary, even though popularity-based features decisively contribute to improving the precision of the models, we observed that the winner methods, i.e. *Simple+N/SIP* and *Full+SIP*, are positively influenced by both the popularity-based features and the attribute features.

We additionally report an analysis of the correlations between pairs of features. From a statistical perspective, uncorrelated features are more likely to bring new information for the classification task. On the other hand, a strong correlation with another feature is a signal of redundancy and so feature reduction techniques, such as principal component analysis [20], can be carried out in order to reduce the dimensionality and the complexity of the model. Below we report as correlation measure the Spearman rank coefficient [39], which computes the correlation among ranked variables. We tested all the pairs with a t-test and the null hypothesis of non-correlation, and we report the top-6 correlated features in Tables 8 and 9. All reported correlations are statistically significant.

Table 8 shows the top-6 highly correlated and non-correlated pairs for the *Simple + N/SIP* and *Full + SIP* methods. One of the top correlations is between the features "Homeunitn QM" and "Fullname M" (refer to section 5.1 for the definition of QM and M) that correspond to the webpage of the University of Trento and the name of a person. This correlation is explained by the fact that the webpage of the university contains the names of its employees. Similarly, the popularity features, i.e. "Selected ≥ 9" and "Selected ≥ 7" (where 7 and 9 are the thresholds introduced in Section 5.3), are strongly correlated. On the other hand, semantically different attributes, such as "Longitude" and "Population", are not correlated. We note that the "Incountry" attribute in *Full + SIP*, which is the most influential (as discussed earlier) attribute in the model, is also correlated with other attributes. As expected, the popularity based features are not correlated with anyone of the entity-set features. For this reason, popularity based features are important for the classification task, and this motivates the good results of the *Simple + N/SIP* and *Full + SIP* methods.

To compare our approach to the IR approaches, we list in Table 9 the top-6 highly correlated and non-correlated features for the *Traditional IR* and *ECIR* methods. Unsurprisingly, we notice a strong correlation between idf, tf-idf (and also their normalized versions). The *Traditional IR* approach presents several redundancies in the features, e.g., in the aggregated scores sum and mean (in order to compute the mean we have to first compute the sum). On the other hand, the number of covered terms, i.e., the number of terms in the entity that are also query keywords, are unrelated to most of the IR metrics, such as tf-idf and idf. It is interesting to note that the top-6 correlated *ECIR* features exhibit correlations that are weaker than the corresponding top-6 features of *Traditional IR*. This observation explains why *ECIR* performs better than *Traditional IR*, since *ECIR* uses more informative features. Finally, naturally correlated scores (e.g., idf and tf-idf) are uncorrelated with respect to different parts of the entity (i.e. titles vs values) validating the claim that entities cannot be treated as documents.

The above results highlight characteristics of the features that may be useful in order to further improve the classification model. A detailed study of how to best integrate this knowledge in our method is part of our future work.

7.4 Time Performance

In order to measure the time performance, we created five new datasets with different number of queries, namely 1k, 5k, 10k, 25k and 50k (by repetidly sampling from D-Top). In Figure 4 we report the time needed for the preprocessing and classifier model learning tasks. We note that the results are simply indicative of the trends, and the absolute times can be significantly reduced by optimizing and parallelizing the operations.

Simple + N/SIP		
Feature 1	Feature 2	Corr.
Homeunitn QM	Fullname M	0.999
Fullname QM	Fullname M	0.999
Faculty NN	University NN	0.999
Selected $\geq$ 9	Selected $\geq$ 7	0.999
Created By M	Is Part of M	0.999
Is part-of M	Category M	0.999
Unselected $\geq$ 9	Occupation NN	0.0007
Domain NN	Found. Date M	0.002
Longitude NN	Population M	–0.003
Birthdate NN	Unselected $\geq$	-0.004
Domain QM	Legal name M	-0.004
University QM	University NN	-0.005

Full + SIP		
Feature 1	Feature 2	Corr.
Incountry NM	Name NM	0.999
Name NM	Longitude NM	0.999
Longitude NM	Latitude NM	0.999
Latitude NM	Country Code NM	0.999
Country code NM	Attribution NM	0.999
Longitude NM	Latitude NM	0.999
Unselected $\geq$9	Occupation I	0.0007
Last name M	Empoyer QM	0.001
Domain I	Found. date M	0.002
Longitude I	Population M	0.003
Birthdate I	Unselected $\geq$5	-0.004
Domain QM	Legal name M	-0.004

Table 8: Top-6 correlated and non-correlated features in *Simple + N/SIP* (left) and *Full + SIP* (right) methods.

Traditional IR		
Feature 1	Feature 2	Corr.
Sum tf-idf	Mean tf-idf	1
Covered term	Covered term %	1
Norm. mean tf-idf	Norm. sum tf-idf	1
Sum tf	Mean tf	0.999
Sum tf	Max tf	0.970
Mean tf	Max tf	0.969
Min tf-idf	BM25	-0.007
Covered term	idf	0.009
Covered term %	idf	0.009
Covered term %	Variance tf	0.010
Covered term	Variance tf	0.010
Covered term	Max tf-idf	0.014

ECIR		
Feature 1	Feature 2	Corr.
idf values	idf entity	1
BM25 entity	BM25 values	0.993
Values length	Entity length	0.981
tf entity	tf values	0.949
tf-idf values	tf-idf entity	0.948
BM25 titles	tf titles	0.865
idf values	tf-idf entity	-0.0001
idf entity	tf-idf entity	-0.0001
query length	tf entity	-0.0003
BM25 values	tf-idf titles	-0.0006
is-place	avg key len.	-0.002
values length	tf values	-0.002

Table 9: Top-6 correlated and non-correlated features in *Traditional IR* (left) and *ECIR* (right) methods.

Preprocessing involves the computation of the set of features. The results (refer to Table 10) show that the required time is 0.5 sec/query for *Simple* and *Full*, and up to 2 sec/query for the IR-based techniques. *ECIR* is the slowest, due to the computation-intensive feature set it involves. Learning is a task whose time requirements depend on the type of classifier we choose (quadratic for the SVM in our case). Figure 4 depicts the time needed for SVM to learn models for some of the proposed methods. The graph shows that the proposed techniques require more time to learn than *Traditional IR*. We plan to investigate this issue further, in order to render the models of the proposed techniques easier to learn.

Finally, ranking time is the time needed to compute the final weights using the model produced by the SVM. Since we employ a linear model that can easily fit in main memory, this time is negligible.

8 Conclusions

This work tackled the problem of reordering a list of entities retrieved by an entity search-engine, based on the extraction of implicit feedback from click-logs. We proposed three novel approaches to extract relevant features from click-

	time/query
Traditional IR	1.4
Simple	0.4
Full	0.6
ECIR	2.0

Table 10: Average preprocessing time per query (sec).

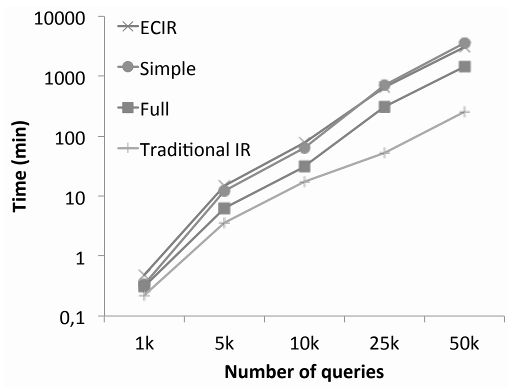


Figure 4: Learning time (y-axis is in log-scale).

logs, and analyzed three feedback extraction methods. Features are extracted considering the relationships among the attributes in different entities in the result list. Moreover, we proposed novel uses of the IR features that exploit the structure of the entities. We also introduced novel measures, suitable for evaluating the results in this new context. The experimental evaluation on real datasets shows that the proposed methods are more effective when compared to traditional IR techniques (that are document aware). The results indicate that considering entities as plain text does not help in the classification task, while the proposed approaches can improve the results by a significant amount.

In the future we can work on integrating a user model with the semantic information of some other external sources, such as knowledge bases, ontologies and word-nets.

We are currently working on the collection of a large real dataset that can help further improve and better evaluate the techniques proposed in this area.

References

[1] A. Agarwal, S. Chakrabarti, and S. Aggarwal. Learning to rank networked entities. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 14–23. ACM, 2006.

[2] Eugene Agichtein, Eric Brill, and Susan Dumais. Improving web search

ranking by incorporating user behavior information. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26, New York, NY, USA, 2006. ACM.

[3] S. Agrawal and S. Chaudhuri. Automated ranking of database query results. In *In CIDR*. Citeseer, 2003.

[4] Barbara Bazzanella, Themis Palpanas, and Heiko Stoermer. Towards a general entity representation model. In *IRI*, pages 431–432, 2009.

[5] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. Learning to rank using gradient descent. In *ICML*, pages 89–96, 2005.

[6] K. Chakrabarti, V. Ganti, J. Han, and D. Xin. Ranking objects by exploiting relationships: computing top-k over aggregation. In *SIGMOD Conference*, pages 371–382. Citeseer, 2006.

[7] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Probabilistic ranking of database query results. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 888–899. VLDB Endowment, 2004.

[8] T. Cheng and K.C.C. Chang. Entity search engine: Towards agile best-effort information integration over the web. In *Proc. of CIDR*, pages 108–113. Citeseer, 2007.

[9] T. Cheng, X. Y., and K. C. Chang. Supporting entity search: a large-scale prototype search engine. In *SIGMOD Conference*, pages 1144–1146, 2007.

[10] T. Cheng, X. Yan, and K.C.C. Chang. Entityrank: searching entities directly and holistically. In *Proceedings of the 33rd international conference on Very large data bases*, pages 387–398. VLDB Endowment, 2007.

[11] Tao Cheng, Hady Wirawan Lauw, and Stelios Paparizos. Fuzzy matching of web queries to structured data. In *ICDE*, pages 713–716, 2010.

[12] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, pages 73–78, 2003.

[13] William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *J. Artif. Intell. Res. (JAIR)*, 10:243–270, 1999.

[14] N. N. Dalvi, R. Kumar, B. Pang, R. Ramakrishnan, A. Tomkins, P. Bohannon, S. Keerthi, and S. Merugu. A web of concepts. In *PODS*, pages 1–12, 2009.

[15] Gianluca Demartini, Claudiu S. Firan, Tereza Iofciu, and Wolfgang Nejdls. Semantically enhanced entity ranking. In *WISE*, pages 176–188, 2008.

[16] X. Dong and A. Y. Halevy. Indexing Dataspaces. In *SIGMOD*, pages 43–54, 2007.

[17] X. Dong, A. Y. Halevy, and J. Madhavan. Reference Reconciliation in Complex Information Spaces. In *SIGMOD*, pages 85–96, 2005.

[18] Georges Dupret and Benjamin Piwowarski. A user browsing model to predict search engine click data from past observations. In *SIGIR*, pages 331–338, 2008.

[19] Yoav Freund, Raj D. Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. In *ICML*, pages 170–178, 1998.

[20] K.P. FRS. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[21] Fan Guo, Chao Liu 0001, and Yi Min Wang. Efficient multiple-click models in web search. In *WSDM*, pages 124–131, 2009.

[22] Fan Guo, Chao Liu, Anitha Kannan, Tom Minka, Michael J. Taylor, Yi Min Wang, and Christos Faloutsos. Click chain model in web search. In *WWW*, pages 11–20, 2009.

[23] Shihao Ji, Ke Zhou, Ciya Liao, Zhaohui Zheng, Gui-Rong Xue, Olivier Chapelle, Gordon Sun, and Hongyuan Zha. Global ranking by exploiting user clicks. In *SIGIR*, pages 35–42, 2009.

[24] Thorsten Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods*, pages 169–184, Cambridge, MA, USA, 1999. MIT Press, MIT Press.

[25] Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142, 2002.

[26] Thorsten Joachims, Laura A. Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR*, pages 154–161, 2005.

[27] Thorsten Joachims and Filip Radlinski. Search engines that learn from implicit feedback. *IEEE Computer*, 40(8):34–40, 2007.

[28] Diane Kelly and Jaime Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2):18–28, 2003.

[29] Ravi Kumar and Sergei Vassilvitskii. Generalized distances between rankings. In *WWW*, pages 571–580, 2010.

[30] H.L. Lau and W. Ng. A multi-ranker model for adaptive xml searching. *The VLDB Journal*, 17(1):57–80, 2008.

[31] T.Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 workshop on learning to rank for information retrieval*, pages 3–10. Citeseer, 2007.

[32] C.D. Manning, P. Raghavan, and H. Schutze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[33] Zoltán Miklós, Nicolas Bonvin, Paolo Bouquet, Michele Catasta, Daniele Cordioli, Peter Fankhauser, Julien Gaugaz, Ekaterini Ioannou, Hristo Koshutanski, Antonio Maña, Claudia Niederée, Themis Palpanas, and Heiko Stoermer. From web data to entities and back. In *Proceedings of the*

22nd international conference on Advanced information systems engineering, CAiSE'10, pages 302–316, Berlin, Heidelberg, 2010. Springer-Verlag.

[34] Jeffrey Pound, Stelios Paparizos, and Panayiotis Tsaparas. Facet discovery for structured web search: a query-log mining approach. In *SIGMOD Conference*, pages 169–180, 2011.

[35] Filip Radlinski and Thorsten Joachims. Query chains: learning to rank from implicit feedback. In *KDD*, pages 239–248, 2005.

[36] Stephen E. Robertson and Steve Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR*, pages 232–241, 1994.

[37] Anish Das Sarma, Xin Luna Dong, and Alon Y. Halevy. Data integration with dependent sources. In *EDBT*, pages 401–412, 2011.

[38] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The Semantic Web Revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006.

[39] C. Spearman. The proof and measurement of association between two things. *The American journal of psychology*, 15(1):72–101, 1904.

[40] `http://developer.yahoo.com/geo/geoplanet`. Yahoo! geoplanet.

[41] `http://disi.unitn.eu/~mottin/ida/`.

[42] `http://entitycube.research.microsoft.com`. Entitycube.

[43] `http://ilps.science.uva.nl/trec-entity/`. Trec 2011 entity track.

[44] `http://semsearch.yahoo.com`. Yahoo! semantic search.

[45] `http://sindice.com`. Sindice.

[46] `http://www.inex.otago.ac.nz/tracks/entity-ranking/entity-ranking.asp`. Inex 2010 entity ranking track.

[47] `http://www.okkam.biz`. Okkam enterprise.

[48] Ji-Rong Wen, Jian-Yun Nie, and HongJiang Zhang. Clustering user queries of a search engine. In *WWW*, pages 162–168, 2001.

[49] W.E. Winkler. The state of record linkage and current research problems. In *Statistical Research Division, US Census Bureau.* Citeseer, 1999.

[50] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *SIGIR*, pages 391–398, 2007.

[51] Zhaohui Zheng, Keke Chen, Gordon Sun, and Hongyuan Zha. A regression framework for learning ranking functions using relative relevance judgments. In *SIGIR*, pages 287–294, 2007.