A Model for Ranking Entities and Its Application to Wikipedia

Gianluca Demartini, Claudiu S. Firan, Tereza Iofciu, Ralf Krestel, and Wolfgang Nejdl
L3S Research Center
Leibniz Universität Hannover
Appelstr 9a, 30167 Hannover, Germany
{demartini,firan,iofciu,krestel,nejdl}@L3S.de

Abstract

Entity Ranking (ER) is a recently emerging search task in Information Retrieval, where the goal is not finding documents matching the query words, but instead finding entities which match types and attributes mentioned in the query. In this paper we propose a formal model to define entities as well as a complete ER system, providing examples of its application to enterprise, Web, and Wikipedia scenarios. Since searching for entities on Web scale repositories is an open challenge as the effectiveness of ranking is usually not satisfactory, we present a set of algorithms based on our model and evaluate their retrieval effectiveness. The results show that combining simple Link Analysis, Natural Language Processing, and Named Entity Recognition methods improves retrieval performance of entity search by over 53% for P@10 and 35% for MAP.

1 Introduction

Finding entities on the Web is a new search task which goes beyond the classic document search. While for informational search tasks (see [7] for a classification of tasks) high precision document search can give satisfying results for the user, a different approach should be followed when the user is looking for specific entities. For example, when the user wants to find a list of “Brazilian female politicians” it is easy for a classical search engine to return documents about politics in Brazil. It is left to the user to extract the information about the requested entities from the provided results. Our goal is to develop a system that can find entities and not just documents on the Web.

Being able to find entities on the Web can become a new important feature of current search engines. It can allow users to find more than just Web pages: also people, phone numbers, books, movies, cars, or any other kind of items. Searching for entities in a collection of documents is not an easy task. Currently, we can see the Web as a set of inter-linked pages of different types, e.g. describing tasks, answering questions or describing people. Therefore, in order to find entities, it is necessary to do a preprocessing step of identifying entities in the documents. Moreover, we need to build descriptions of those entities to enable search engines to rank and find them given a user query.

Applying classical Information Retrieval (IR) methodologies for finding entities can lead to low effectiveness as seen in previous approaches [3, 9]. This is because entity search is a task different than document search. It is crucial to rely on consolidated information extraction technologies if we do not want to start with an already high error that the ranking algorithms can only increase.

In this paper we first propose a general model for finding entities and we show how this can be applied to different entity search scenarios. We generalize this search task and identify its main actors so that we can optimize solutions for different search contexts such as, for example, the Wikipedia corpus. Building on top of the designed model, we developed search algorithms based on Link Analysis, Natural Language Processing (NLP), and Named Entity Recognition (NER) for finding entities in the Wikipedia corpus. Moreover, we experimentally evaluated the developed techniques using a standard testbed for Entity Ranking (ER). We show that these algorithms improve significantly over the baseline and that the proposed approaches – incorporating Link Analysis, NLP and NER methods – can be beneficially used for ER. We evaluated our algorithms for entity ranking only on the Wikipedia scenario. It will be a future step to extend the approach to the entire Web of Entities.

The main contributions of this paper are:

- Proposing a general model for Entity Ranking (Section 2);
- Applying the model to enterprise, Web, and Wikipedia scenarios (Section 3);
- Creating a set of algorithms for finding entities in Wikipedia (Section 5);
• Evaluating the retrieval effectiveness of the algorithms presented (Section 6);

The paper is structured as follows: We start by defining the ER model (entities, queries, and the ER system) in Section 2. We then show three real-world application scenarios where such ER models are useful (Section 3). In Section 4 we present the environment we used for testing our ER approaches. The algorithms proposed are described in detail in Section 5, and their retrieval performance is evaluated in Section 6. Section 7 presents the previous work in this field. We finally conclude the paper and present future improvements in Section 8.

2 A Formal Model for Entity Ranking

Searching for information on the Web is a very common task that many search engines deal with. The difficult part is to distinguish if the answer to the user’s information need is just a fact that appears in different pages or if it is information about a specific object, an entity. Searching for named entities, such as “the first dog on the Moon” or abstract entities like “dog species bred in England” is quite different than searching for “tips and tricks on raising dogs” (i.e., informational queries).

The problem of ranking entities in IR can be split in several steps. First, the user’s information need has to be translated into a query which has to be interpreted and the entity need has to be extracted. The search engine has to understand what type of entity the user is searching for and what properties the retrieved entities should have. In the next step, relevant results are retrieved. The results have to be retrieved according to the entity description which include many properties, e.g., the type. We propose in the following section a model for the entire ER process that can be instantiated in a number of different contexts such as, for example, the Web.

2.1 Entities

The central part of the model is the set of entities. An entity $e^i$ is something that has separate and distinct existence and objective or conceptual reality. An entity is represented by its unique identifier, and by a set of properties described as $(<\text{attribute}>, <\text{value}>)$ pairs (see Figure 1). The properties of an entity can include, for example, its name or its type. Moreover, it is important to notice that relations can be present between entities. It is possible to model these relations as other properties using $(<\text{attribute}>, <\text{value}>)$ pairs where the value would be the target entity of the relation. This representation of relations is consistent with previous work on entity relation search [17].

We can now define the entity description $d(e^i) = \{ID^i, P^i\}$ for the entity $e^i$ as composed of an entity identifier $ID^i = id(e^i)$ and a set of properties $P^i = \{(a_1^i, v_1^i), \ldots, (a_n^i, v_n^i)\}$ of the type $(<\text{attribute}>, <\text{value}>)$ pairs. For example, the soccer player “Alexandre Pato” could have as ID the automatically generated unique identifier $ap12dH5a$ and properties such as $(\text{born}_{\text{en}}, 1989)$ or relations with other entities such as $(\text{playing}_\text{with}, \text{acm15hDJ})$ where $\text{acm15hDJ}$ is the ID of the soccer club “A.C. Milan”.

2.2 Data Sources

In order to create the entity descriptions $d(e^i)$ (see Section 2.1 and Figure 1) we need to extract data about entities from several sources. For example, for describing the company “Microsoft Corporation” we might want to harvest the Web in order to find all the facts and opinions about this entity. For this reason, we call data sources the provenance of the information we collect in an entity description. We define a data source $s_j$ as any passage of a digital document. This can be an XML element, a paragraph of an e-mail, a blog post on the Web. Each data source $s_j$ can be about one or more entities. The aggregation of all the data sources about the same entity $e^i$ (noted as $\bigcup s_j^i$) will create the properties part $P^i$ of the entity description $d(e^i)$ as defined in Section 2.1. This would define inferring the description of an entity as: $\bigcup_j s_j^i \Rightarrow P^i$. The relations between entities are also inferred from the data sources and are part of $P^i$.

2.3 Users’ Information Need

After modeling the entities and the data sources used for creating their description, we want to model a user searching for entities. We assume that a user has an information need, that is, she wants to find a list of entities that satisfy some properties. It is a user task to create, starting from the information need, a query, either using keywords or natural language, that can be processed by the system. The user query will describe the set of properties that an entity should satisfy for being relevant. For example, a query might indicate the type of entities to be retrieved (e.g., “cars”) and distinctive features (e.g., “German”, “hybrid”). A real-world example is given by the search engine Sindice.com where the user can issue queries like “Washington class:person” specifying the type of results she wants to get. A query $q$ is defined, similar to the entity descriptions, as a list of $(<\text{attribute}>, <\text{value}>)$ pairs. Thus, $q = \{(a_1, v_1), \ldots, (a_n, v_n)\}$. 

W

acm
2.4 Entity Ranking System

At this point, a collection of entity descriptions $D = \{d(e_1), \ldots, d(e_n)\}$ and a user query $q$ is available. An Entity Ranking System (ERS) will now take as input these two elements and will return a ranked list of entities $E = \{e_1, \ldots, e_j\}$ (Figure 2 shows a sketch of the flow inside the ERS). In order to do this, an ERS will hard-code a scoring function $\varphi(q, d(e_i))$ that returns a score (i.e., a real number) for a given user query $q$ and entity description $d(e_i)$. This score represents the confidence or probability of the entity $e_i$ of being relevant to the query $q$. In this way the ERS will be able to rank the entire set of entities according to the confidence of being relevant to the user query. Of course, the scoring function can take into account several evidences of relevance such as the comparison between properties in $q$ and properties in $d(e_i)$, the popularity value of the entities in the collection (e.g., PageRank), or give more importance to a particular property (e.g., the type of entities to be returned).

3 Application Scenarios for Entity Ranking

In this section we show how it is possible to instantiate the proposed model with several real world examples. After showing how to uniquely identify one entity, we will present three different entity search scenarios.

3.1 Generating Global Entity Identifiers

The output of an ERS is a list of entities, or, better, a list of identifiers representing the retrieved entities. It is, therefore, necessary to assign a global identifier (ID) to each entity in the collection as an initialization step.

Generation of unique identifiers is already widely needed and attempts to generate them are already undergoing. The OKKAM European Integrated Project is dealing with such ID generation on the Web. The main goal of the OKKAM project is to enable the Web of Entities. This will be accomplished by supporting the use of globally unique identifiers for entities with the aim that the same object will always be referred by the same identifier. The subpart of the entire OKKAM infrastructure that has the role of managing entity identifiers is the Entity Name System (ENS), presented in [5], as:

A service which can enable the reuse of globally unique URIs across semantic datasets produced in a fully decentralized and open environment.

The main functionalities of the ENS are: search for the identifier of an entity, generation of entity identifiers, matching entities present in the repository with external ones, and ranking entities by similarity to a given one.

Our general model for finding entities can build on top of the ENS service in the Web scenario as well as in an Organizational Knowledge Management context.
3.2 Ranking Consumer Products

As a first example, we describe in the following how it is possible to instantiate the proposed model to the context of particular types of entities such as products in an enterprise context. We can see each product of a company as an entity $e^i$ and we can easily imagine a customer searching for products providing a list of constraints. We can define $d(e^i)$ as the description of the product $e^i$ as defined by the company. For example, for a car manufacturer, a description could include the following attributes $(a_1^i, \ldots, a_n^i)$: price, exterior color, interior color, transmission, fuel type, engine, etc. The data sources for building the entity description $d(e^i)$, that is $\bigcup s_j^i$, can be the company database of cars, if available, or the cars catalogue. In this case the $ID(e^i)$ could be represented by the serial number of the car. Possibly, we can describe relations between two entities as for example, we can relate two cars $(e^1, e^2)$ by the relation $is.a_facelift.of$.

Putting together the entire set of entity descriptions built by the company we obtain the collection $D$. At this point the user can build a query $q$ which is a set of constraints of the type $(<attribute>,<value>)$ pairs, for example, $(price,20000), (exterior\_color,\text{red})$. At this point, using a ranking function $\phi$ that matches the constraints in $q$ with the set of $(<attribute>,<value>)$ pairs for each $d(e^i) \in D$ a list of cars ranked according to $\phi(q,d(e^i))$ can be returned to the user.

3.3 Ranking Entities on the Web

As second example of model instantiation we show how to perform the ER process on the Web. The definition of entity on the Web is not as trivial as in the enterprise context. We could, for example, take into account any entity $e^i$ which is mentioned in at least one Web document. This assumption will make the set of entities $D$ very large with implication on the efficiency of the ER process. In this case, the entity description $d(e^i)$ will contain information coming from several passages of different Web pages which are the data sources $s_j^i$. For example, the entity “George W. Bush” will contain information coming from many different Web sites such as, for example, whitehouse.gov and wikipedia.org. It is easy to imagine that one problem for building entity descriptions on the Web is the presence of contradicting values and of representational variations (i.e., the same entities represented in different ways due to, for example, misspellings). Moreover, different entities will have different attributes in their descriptions. For example, people will have an attribute $date\_of\_birth$ while companies will have an attribute $share\_capital$. Relations between entities could be inferred from links between Web pages. For example, the relation existing between the entity “George W. Bush” and the entity “White House” can be inferred by the large amount of links between pages discussing the first entity with pages describing the second one. In the querying process the user should be able to specify both the entity type and the entity properties which are desired. In the ranking function, more evidence, such as entity popularity, could be taken into account. For example, a query about “trees” during Christmas time might return at the top of the list entities of the type “Christmas trees”.

3.4 Ranking Entities in Wikipedia

The last example of instantiation of the model for ER is Wikipedia. In this case we consider in $D$ any entity $e^i$ that has its own page in Wikipedia. With this assumption we can easily see these pages as the entity description $d(e^i)$ and the set of the Wikipedia pages that describe an entity as the collection $D$. Of course, in Wikipedia there are pages which do not describe a particular entity as, for example, the “List of …” pages. The challenge is to identify which are not-entity pages and discard them from $D$. For each entity the $(<attribute>,<value>)$ pairs can be build, for example, out of the info-boxes of the Wikipedia pages which contain factual information about the described entity (for example, articles about people contain information about name, birth date, birth place, etc.). In the Wikipedia scenario the sources of information are the people and each $s_j^i$ contributing to $d(e^i)$ can be reconstructed from the edit history of each page allowing also to associate trust values in order to weight more particular sources (see also [1] about such computation). For defining the type property in $d(e^i)$ the Wikipedia category information can be used. Relations between entities can be discovered analyzing the Wikipedia internal links between pages. The query can be built by the user providing some keywords describing interesting properties plus the selection of a Wikipedia category in order to provide information about the type of entities which are requested. The ranking function $\phi(q,d(e^i))$ should use both information about the properties and the type in order to produce the best ranking.

4 Experimental Environment

For evaluating our algorithms designed on the proposed model we used an existing environment created especially for the purpose of ER. The ER track at the Initiative for the Evaluation of XML Retrieval (INEX) 2007\footnote{http://inex.is.informatik.uni-duisburg.de/2007/} made available an automatic evaluation environment for ER based on Wikipedia. We built our system around this environment for easier, objective, and comparable evaluation.
### 4.1 The INEX Entity Ranking Topic example.

The document collection used for evaluating our approaches is the Wikipedia XML Corpus based on an XMLified version of the English Wikipedia in early 2006 [11]. The collection contains 659,338 Wikipedia articles. On average an article contains 161 XML nodes, where the average depth of a node in the XML tree of the document is 6.72. The original Wiki syntax has been converted into XML, using general tags of the layout structure (like `article`, `section`, `paragraph`, `title`, `list`, and `item`), typographical tags (like `bold`, `emphatic`), and frequently occurring link-tags. For details see Denoyer and Gallinari [11].

The official topics have been manually assessed by the INEX 2007 participants. The set contains 46 total topics, 21 adapted ad hoc topics along with 25 ER designed topics. An example of an INEX 2007 Entity Ranking Topic is presented in Table 1.

Although this task seems easy given the Wikipedia corpus, we have to retrieve results matching the sought type with the given restrictions. Relevant results for the example given in Table 1 would thus be: “Beyond Good and Evil” or “Thus Spoke Zarathustra”, all of these being books by Nietzsche. Irrelevant results, although they still contain some information related to the Topic, would be: “List of works by Friedrich Nietzsche” (where we find links to all his books), or even “Friedrich Nietzsche” (the page describing the author). Another observation is that although categories in Wikipedia are clearly defined they do not contain all relevant entries. For example, not all books written by Friedrich Nietzsche are in the category “books by Friedrich Nietzsche”.

### 4.2 System Architecture

In this section we describe the architecture of the ERS we used. Starting from the raw structured XML documents, we created a Lucene\(^2\) inverted index with one Lucene document for each Wikipedia page. We created fields for each

\(^2\)http://lucene.apache.org

article element (e.g., `title`, `text`, `categories`, `links`) which are searchable in parallel with an integrated ranking. We used standard tools integrated in Lucene to remove stopwords and perform stemming on the remaining words in the Wikipedia corpus. The same procedure is then applied on the query at retrieval time. The retrieval is done using the default Lucene implementation, i.e., the Vector Space Model with cosine similarity and TF-IDF weighting.

After the creation of the index, the system can process the INEX Entity Ranking 2007 Topics. Different approaches are adopted for building queries out of INEX Topics (as shown later in detail in Section 5). The approaches, which can be used interchangeably or complementary, make use of:

- Standard IR search - Using the default implementation in Lucene to search query terms in the text;
- Link Analysis information - Exploring outgoing Links of Wikipedia entities to focus the search;
- Natural Language Processing - Identifying main concepts in the query and modifying it by adding useful terms or deleting unnecessary noise;
- Named Entity Recognition - Extracting and making use of available named entities in the query.

The INEX Topic is processed using these approaches and a disjunctive Lucene query is created starting from the Topic Title, along with the specified Categories from the Topic. After the generation of the query, the index can be queried and a ranked list of retrieved entities is generated as output.

### 5 Entity Ranking Algorithms

In this section we present our algorithms for improving entity retrieval. Queries for entity retrieval are built out of the Topics, enhanced with the different techniques, and searched throughout the different index fields.

We use the following notations for describing the algorithms:

- \(W^T = \{w^T_1, \ldots, w^T_n\}\) – the words in the given Topic Title;
- \(W^C = \{w^C_1, \ldots, w^C_n\}\) – the words in the given Topic Category;
- \(W^T_{Adj} = \{w^T_{Adj_1}, \ldots, w^T_{Adj_n}\}\) – the adjectives in the Topic Title;
- \(W^T_{Noun} = \{w^T_{Noun_1}, \ldots, w^T_{Noun_n}\}\) – the nouns in the Topic Title;
5.1 Baseline

As baseline we used a naïve approach, where the query is constructed from the Title information given in the Topic. We search the Title in the text fields of the indexed Wikipedia pages. Additionally, the Category in the given Topic is searched in the category field of the index. We do not make this part of the query mandatory as category information available in Wikipedia is not always accurate or sometimes not even present. Nevertheless, the existing entity-in-category information influences the results for the respective entities.

The query part searched in the Wiki page text will thus contain following terms:

\[ w_i \in W^T \]

For example, for the Topic in table 1 using the baseline (with stopword removal and stemming) a Lucene query for searching in the text and in the category of Wikipedia pages, is of the form:

\[
\text{text:}(\text{book written friedrich nietzsch}) \quad \text{category:}(\text{book book friedrich nietzsch})
\]

Table 2 shows the first 10 results for Topic #33 (“Books written by Friedrich Nietzsche”) using the baseline approach.

5.2 Link Based Approaches

Wikipedia, just like the Web, is highly interconnected. Search engines make use of link information for traditional Information Retrieval document ranking. Wikipedia pages, where each page represents an entity, has external links pointing to pages outside the Wikipedia corpus and internal links, which point to other Wikipedia entities. While external links are usually presented in a separate list at the end of the entity description, internal Wikipedia links appear inside the text. When indexing the entity pages, we have kept in the indexed text the names of the linked entities where they appear, and we have also indexed their titles in a separate field called outLinks to ensure that their importance is not lost in the entity text. In addition to the baseline approach, the textual part of the query is searched also in the outLinks index field. This approach can easily be combined with others to improve performance (e.g., searching the Topic Title in the text field AND in the outLinks field).

For example, some of the entities that Nicolaas Bloembergen links to are: Dutch, physicist, American, Harvard University, 1948, University of Utrecht, nuclear magnetic resonance, Lorentz Medal, Nobel Prize in Physics, laser spectrology, and others. There are many terms present in the list of linked entities. As the information in the linked entities field is more condensed than in the text field, linked entities matching improves the ranking of the search results.

5.3 Approaches Based on Synonyms and Related Words

Wikipedia, just as the general Web, presents its information in natural language. There is no formal representation and only limited structured information. After describing how to use the structured information, like category information or link structures, we examine different approaches exploiting natural language properties.

The first approach accommodates the fact that there are various ways of conveying the same meaning within natural language sentences or even words. This observation lead us to the conclusion that only using the present keywords in the Title, Description, or Category fields is not enough. We therefore extended the query using related words and synonyms of the extracted keywords. To identify nouns, whose synonyms and related words were used to extend the query we use part-of-speech tagging from LingPipe [2] – a suite of java libraries for Natural Language Processing. The part-of-speech tagger was trained on the manually labeled Brown corpus, a collection of various types of text documents, to obtain statistical models to perform part-of-speech tagging.

The synonyms and related words were automatically generated using the WordNet semantic lexicon [12]. WordNet can be seen as a dictionary that groups English words into sets of synonyms and stores the various semantic relations between these synonym sets (synsets). As there are several synsets available for each term in WordNet, we first perform Word Sense Disambiguation, as done [16], to choose the correct meaning for the nouns in the query. Then we extend the query with additional information about each noun: (1) add all synonyms from the previously identified synset; (2) add all words that have a relationship (except for antonyms) to the identified synset. The additional words are then used to enrich the query to improve the recall of our system:

\[ w_i \in W^T \cup \text{Synonyms}(W^T) \quad \text{or} \quad w_i \in W^T \cup \text{RelatedWords}(W^T) \]

5.4 Core Characteristics Approach

To make the query more precise, we examined the results for removing parts of the query. On the one hand we removed duplicate information in the title by finding synonym nouns occurring in the category field. This was achieved using WordNet as described in 5.3. Since we try to find entities and not categories, the idea is to remove category keywords from the query. Making use of synonym information makes this approach more robust and helps to extract core characteristics from the user query. On the other hand we used LingPipe’s part-of-speech Tagger to identify verbs, nouns, adjectives, etc. and removed all except nouns.
<table>
<thead>
<tr>
<th>Rank</th>
<th>Entity</th>
<th>Containing Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Beyond Good and Evil</td>
<td>books by friedrich nietzsche; 1886 books</td>
</tr>
<tr>
<td>2</td>
<td>On the Genealogy of Morals</td>
<td>books by friedrich nietzsche; 1887 books</td>
</tr>
<tr>
<td>3</td>
<td>Thus Spoke Zarathustra</td>
<td>books by friedrich nietzsche; 1885 books</td>
</tr>
<tr>
<td>4</td>
<td>The Birth of Tragedy</td>
<td>books by friedrich nietzsche; 1872 books</td>
</tr>
<tr>
<td>5</td>
<td>The Antichrist (book)</td>
<td>books by friedrich nietzsche; 1895 books</td>
</tr>
<tr>
<td>6</td>
<td>Human, All Too Human</td>
<td>books by friedrich nietzsche; psychology books; 1878 books</td>
</tr>
<tr>
<td>7</td>
<td>The Gay Science</td>
<td>books by friedrich nietzsche; philosophy books; 1882 books</td>
</tr>
<tr>
<td>8</td>
<td>The Twilight of the Idols</td>
<td>books by friedrich nietzsche; philosophy books; 1889 books</td>
</tr>
<tr>
<td>9</td>
<td>The Peasant War in Germany</td>
<td>books by friedrich engels</td>
</tr>
<tr>
<td>10</td>
<td>The German Ideology</td>
<td>books by karl marx and friedrich engels; philosophy books; marxism; 1932 books</td>
</tr>
</tbody>
</table>

Table 2. Top 10 baseline results for Topic #33 ("Books written by Friedrich Nietzsche") together with their containing Wikipedia categories.

and adjectives. Observations showed that nouns and adjectives are especially helpful to describe entities, whereas verbs mostly introduce noise to the results due to their generality. The formal notation for this approach is:

\[ w_i \in W^T_{Adj} \cup (W^T_{Nouns} \setminus (W^C \cup \text{Synonyms}(W^C))) \]

5.5 Named Entity Recognition Approach

Another well known concept in Information Extraction is Named Entity Recognition. The knowledge about named entities in the query can be a valuable hint to identify what kind of entity is expected in the answer. We use Named Entity (NE) Recognition provided by LingPipe. Finding named entities can be done using dictionary matching, regular expressions, or statistical approaches. We used a machine learning approach with a model gained from supervised training on a large news article corpus. We identified different named entities like organizations, locations, and persons. The found named entities were then used to perform a keyword search using the following terms:

\[ w_i \in W^T \cap \{\text{NamedEntities}\} \]

Table 3 shows an example of the different Approaches.

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom Hanks movies</td>
<td>Films where he plays a leading role.</td>
</tr>
<tr>
<td>Synonyms</td>
<td>Tom &quot;Uncle Tom&quot; Hanks &quot;Thomas J. Hanks&quot; movies film flick &quot;motion picture&quot; &quot;motion-picture show&quot; &quot;moving picture&quot; &quot;picture show&quot; &quot;moving-picture show&quot; where he plays a leading role</td>
</tr>
<tr>
<td>Related Words</td>
<td>Synonyms plus 50 additional concepts related mainly to motion pictures</td>
</tr>
<tr>
<td>Core Characteristics</td>
<td>Tom Hanks leading role</td>
</tr>
<tr>
<td>Named Entities</td>
<td>Tom Hanks</td>
</tr>
</tbody>
</table>

Table 3. Query after applying different strategies.

6 Experimental Results on Wikipedia

For evaluation of our algorithms we used the Wikipedia collection provided by INEX. We used the approaches presented in Section 5 and combination of those, with the same notations as used previously, and some additional notations introduced here. Thus, a query is of the form:

\[ q = \{(\text{field}_i,\text{terms}_j)\} \]

where \( \text{field}_i \) is one of the fields in the Lucene index:

- \( \text{text} \) – the Wikipedia page text;
- \( \text{category} \) – Wiki categories of the pages;
- \( \text{outLinks} \) – outgoing links of the Wiki pages;

and \( \text{terms}_j \) is a list of terms which should be searched in \( \text{field}_i \):

- \( W^X \) – a list of words given in the Topic (see Section 5 for the complete notations);
- \( \text{SY}(X) \) – apply the synonyms approach on the list of words \( X \) (e.g., \( \text{SY}(W^T) \));
- \( \text{RW}(X) \) – apply the related words approach on \( X \);
- \( \text{NE}(X) \) – extract only the named entities from \( X \);
- \( \text{CC}(X) \) – apply the core characteristics approach on \( X \);
- \( X \cup Y \) – union of all terms in \( X \) and \( Y \).

We can combine terms from different approaches: e.g. \( q = \{\text{text}, W^T \cup \text{NE}(W^T)\}, \{\text{category}, W^C\} \) would duplicate the named entities appearing in the Topic Title (thus putting a double weight on the named entities only) and search this in the Wiki page text. Additionally the Topic Category is searched in the Wikipedia categories.

Table 4 presents the Mean Average Precision (MAP) and Precision for the first ten retrieved results (P@10) of our
methods. Additional to the query presented for each approach, the Category given with the Topic was also searched in the category field of the index. The baseline used is approach #1 with a MAP and P@10 values of 0.20 and 0.19.

We evaluated our algorithms both independently and as combinations of several approaches. All our approaches improved in terms of both MAP and P@10 over the baseline, and the combination of all approaches proved to improve most.

**Outgoing Links.** Approaches #2, #3, #4 from Table 4 show the results for searching with the terms from the Topic Title, with the Core Characteristics, and with the Named Entities approaches in the outgoing links text of Wikipedia pages, respectively. While the simple (#2) approach already shows 21% improvement in P@10, the CC approach reaches 37% statistically significant improvement in P@10 and 15% improvement in MAP over the baseline. This proves extracting concept names (done as outgoing links in Wikipedia) from entity descriptions to be a valuable additional information for raising early precision values.

**Synonyms and Related Words.** Adding only synonyms of nouns (#5) results in better performance than adding all related words of the nouns (#6). SY improved MAP at a statistically significant level by 15% and P@10 by 21%. This is due to the vast amount of noise added by RW. Also SY adds some noise as although Word Sense Disambiguation was performed prior to adding the synonyms, still some synonyms are misleading and might need a further filtering step.

**Core Characteristics.** Approach #7, when used for searching in the whole page text shows the same level of improvement as RW. But when using CC for searching in the outgoing links (#3) we see a high improvement of 37% in P@10. This shows extracting the key concepts from both the page text and the query text as being extremely efficient for improving early precision.

**Named Entity Recognition.** Similar to SY and CC, NE (#8) shows improvements of 15% and 21%, both statistically significant, for MAP and P@10, respectively. We see that searching with more weight on the named entities helps improving the ranking.

**Combining the approaches.** All approaches improve but ranked entities differently. This leaves room for improvement by combining the single approaches. We performed several combinations and present only the best performing ones. When searching in the page text, we found that including all methods in the query (#9) improves MAP by 35% and P@10 by 47%. When also adding the best performing search method on outgoing links we have the same improvement of 35% in MAP with a very high statistical significance and a maximum improvement of 53% in P@10.

### Table 4. Mean Average Precision and Precision for the first 10 results of the different techniques on the Wikipedia corpus. The results marked with * are statistically significant (two-tailed t-test, p < 0.05) as compared to the baseline (#1).

<table>
<thead>
<tr>
<th>Nr</th>
<th>Method: $q = {\text{category}, W^t} \cup \ldots$</th>
<th>MAP</th>
<th>P@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{text, $W^t$}</td>
<td>0.20</td>
<td>0.19</td>
</tr>
<tr>
<td>2</td>
<td>{text, $W^t$, {outLinks, $W^t$}}</td>
<td>0.22</td>
<td>0.23</td>
</tr>
<tr>
<td>3</td>
<td>{text, $W^t$, {outLinks, CC($W^t$)}}</td>
<td>0.23</td>
<td>0.24</td>
</tr>
<tr>
<td>4</td>
<td>{text, $W^t$, {outLinks, NE($W^t$)}}</td>
<td>0.23</td>
<td>0.24</td>
</tr>
<tr>
<td>5</td>
<td>{text, $W^t$, {outLinks, SY($W^t$)}}</td>
<td>0.23</td>
<td>0.24</td>
</tr>
<tr>
<td>6</td>
<td>{text, $W^t$, {outLinks, RW($W^t$)}}</td>
<td>0.23</td>
<td>0.24</td>
</tr>
<tr>
<td>7</td>
<td>{text, $W^t$, {outLinks, CC($W^t$)}}</td>
<td>0.23</td>
<td>0.24</td>
</tr>
<tr>
<td>8</td>
<td>{text, $W^t$, {outLinks, NE($W^t$)}}</td>
<td>0.23</td>
<td>0.24</td>
</tr>
<tr>
<td>9</td>
<td>{text, $W^t$, {outLinks, SY($W^t$)} \cup RW($W^t$) \cup CC($W^t$) \cup NE($W^t$)}</td>
<td>0.27</td>
<td>0.29</td>
</tr>
<tr>
<td>10</td>
<td>{text, $W^t$, {outLinks, CC($W^t$)}}</td>
<td>0.27</td>
<td>0.29</td>
</tr>
</tbody>
</table>

7 Related Work

Finding entities on the Web is a recent topic in the IR field. The first proposed approaches [3, 8, 9] mainly focus on scaling efficiently on Web dimension datasets but not on the effectiveness of search. The goal of this paper is to improve the precision in the ER task.

A formal model for entities has been presented in [13]. This entity representation is, similarly to our proposal, based on (<attribute>, <value>) pairs and on a “Category of reference” that describes the entity type which can be taken from an ontology. In our paper we propose a model for the entire ER process where the entity representation is just a sub-part. A framework for modelling the IR process has been presented in [15] where the authors present a matrix-based framework for modelling possible search
tasks. The model we propose is focused on ER; it is less formal but more intuitive.

Approaches for finding entities have also been developed in the Wikipedia context. For example, Pechevski et al. [14] use the link information for improving the effectiveness of ER in Wikipedia obtaining as best result 0.36 Mean Average Precision. In [10] the authors improve the effectiveness of ER leveraging on a highly accurate ontology for refining the search on the Wikipedia category hierarchy obtaining as best result 0.15 Mean Average Precision. Compared to these approaches, we start first designing a model for ER making the development of algorithms possible also in domains different from Wikipedia. Our next step will be to apply the algorithms, evaluated on the Wikipedia corpus, on the entire Web, as done in [3, 8, 9], aiming to find the best compromise between efficiency and effectiveness of search. Another work which can be a foundation for an effective ER is the automatic identification of instances and classes in the Wikipedia category hierarchy [18]. Knowing which categories describe instances can help the ERS in finding entities relevant to the query because not all the articles in Wikipedia are entity descriptions.

An important related area of research is entity identity on the Web. It is crucial for the ER task being able to uniquely and globally identify entities on the Web so that the search engine can return a list of identifiers to the user who can afterwards navigate in the entity descriptions. A strong discussion already started in the Web research community [4, 6] and solutions for entity identity resolution on the Web have been proposed [5]. Our solution for finding entities relies on these infrastructures able to globally identify entities on the Web.

8 Conclusions and Further Work

In this paper we presented a general model for ranking entities and we shown how the model can be applied to different real world scenarios. We described in detail a possible instantiation of the model and a set of algorithms for the Wikipedia dataset. The experimental evaluation of the ER algorithms has shown that by combining our approaches we achieve an improvement of 35% in terms of MAP and of 53% in terms of P@10. In this paper and in related work (see Section 7) it is possible to notice that Precision values are low overall. This fact indicates that the entity ranking research field still needs more work focusing on high precision algorithms in order to provide the users with a satisfying search experience.

As continuation of this work, based on the proposed model, we will design ER algorithms for the entire Web of Entities. The first step will be to identify entities in Web pages. After this we will build entity description which can be indexed by a search engine allowing the end user to query for entities. Moreover, on the Wikipedia corpus, we will adapt our ER algorithms for the scenario where the user provides the ERS with a natural language query as, for example, in the Description part of the INEX Entity Ranking Topics.

9 Acknowledgements

This work is partially supported by the EU Large-scale Integrating Projects OKKAM\(^3\) - Enabling a Web of Entities (contract no. ICT-215032) and PHAROS\(^4\) (IST contract no. 045035).

References


\(^3\)http://fp7.okkam.org/
\(^4\)http://www.pharos-audiovisual-search.eu/


