An Architecture for Finding Entities on the Web

Gianluca Demartini, Claudiu S. Firan, Mihai Georgescu, Tereza Iofciu, Ralf Krestel, and Wolfgang Nejdl
L3S Research Center, University of Hanover
Appelstr. 9a, 30167 Hanover, Germany
{*demartini,firan,georgescu,iofciu,krestel,nejdl*}*@L3S.de*

***Abstract*—Recent progress in research fields such as Information Extraction and Information Retrieval enables the creation of systems providing better search experiences to web users. For example, systems that retrieve entities instead of just documents have been built. In this paper we present an approach for large-scale Entity Retrieval using web collections as underlying corpus. We propose an architecture for entity extraction and entity ranking starting from web documents. This is obtained (1) using an existing web document index and (2) creating an entity centric index. We describe advantages and feasibility of our approach using state-of-the-art tools.**

***Keywords*-entity retrieval; web search; natural language processing;**

I. Introduction

In recent years web search technologies have been moving from the concept of web page retrieval to more challenging tasks such as retrieving images, videos, news, etc. The current focus of Information Retrieval (IR) research and of advances in commercial search systems is to provide the user with something more than just a ranked list of web pages. One such possibility is to address *Entity Retrieval* (ER) queries, that is, queries where the user is looking for a list of entities as, for example, "American countries" where the expected result is a list containing "Mexico", "United States of America", etc.

Following the trend initiated by commercial systems such as Yahoo! Correlator[1] and Google Search Options, it is possible to see how web search engines are going in the direction of providing the user with an overview of all the information available about the query, and not just the top 10 documents. In this context, it is also important to provide the user with a list of entities relevant to her query.

Many years of research on how to rank documents have made current web search engines good in such task. The same techniques can not be directly applied to these new tasks but they need to be adapted. Some work have been done, for example, on how to rank experts in an enterprise context. When we want to rank people or, generally, entities we need to apply techniques from different research fields such as Information Extraction (IE), Natural Language Processing (NLP), and IR where effective approaches for specific tasks have been developed. The availability of out-of-the-box tools developed in these different fields gives now the opportunity to combine them in order to provide the web user with new ways of accessing information. Previous approaches mainly focus on few specific entity types (e.g., people and phone numbers [7]) or on the cleaner context of Wikipedia [19] while our architecture is designed for finding any type of entity on the web.

In this paper we present a possible architecture for such a system aiming at combining approaches from different research fields with the goal of providing the user with tools for mining the huge amount of web pages relevant to her keyword query. Our approach uses standard NLP tools in order to identify named entities in top retrieved web pages. We also propose baseline algorithms for ranking such set of retrieved entities. We show results of an experimental study on how current search engines can be the basis of such approach.

The rest of the paper is structured as follows. In Section II we briefly describe previous work in the area of Entity Retrieval. In Section III we analyse in detail two existing systems for finding entities in Wikipedia. In Section IV we define the proposed architecture for an entity web search engine. In Section V we detail the steps of the ER process. In Section VI we analyse how the output of current web search engines can be used as starting point for the proposed system. Then, in Section VII we describe two possible datasets to be used for evaluating ER algorithms. Finally, Section VIII concludes the paper.

II. Related Work

Finding entities instead of documents on the web is a recent topic in the field of IR. The earliest approaches [2], [6], [7] mainly focus on scaling efficiently on web dimension datasets but not on retrieval effectiveness. The goal of this paper is to provide an scalable architecture for finding typed entities on the web enabling the inclusion of effective ranking algorithms.

Approaches for finding entities have also been developed in the Wikipedia context. Previous approaches to rank entities in Wikipedia exploited the link structure between Wikipedia pages [19] or its category structure using graph based algorithms [22]. Other approaches used semantic and NLP techniques to improve effectiveness of ER systems [12], [13]. In [13] the authors improve the effectiveness of ER leveraging on a highly accurate ontology for refining

[1] http://correlator.sandbox.yahoo.net/

the search on the Wikipedia category hierarchy. Compared to these approaches, we aim at building a system for ER on the entire web making the development of ranking algorithms possible also in domains different from Wikipedia. Our next step will be to design, apply, and evaluate algorithms, using as document collection a web crawl.

With respect to previous approaches we based our algorithms on a structured representation of entities at indexing level – we annotate web pages identifing entities which enable the possibility of constructing an entity centric structured index. For this reason, relevant to our work are projects aiming at extracting and annotating entities and structure in Wikipedia. For example, versions of Wikipedia annotated with state of the art NLP tools are available [20], [17]. It is clear that performing the same task on a general web crawl is more challenging given the lack of regularities, presence of spam and of not well-formatted pages.

Another relevant work is [23] which also aims at retrieving entities in Wikipedia but without the assumption that an entity is represented by a Wikipedia page. They rather annotate and retrieve any passage of a Wikipedia article that could represent an entity. This is similar to our approach where an entity is modelled as a passage in a web document. A foundation for an effective ER can also be the automatic identification of instances and classes in the Wikipedia category hierarchy [24]. Knowing which categories describe instances can help the system in finding entities relevant to the query. Such approaches could be adapted to improve entity type classification when performed on the web.

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 afterwords navigate in the entity descriptions. A strong discussion already started in the web research community [3], [5] and solutions for entity identity resolution on the web have been proposed [4] also when identities are evolving over time [16]. Our solution for finding entities relies on these infrastructures able to globally identify entities on the web.

III. Existing Systems

In this section we describe how existing sytems perform the ER task. We describe two systems: PowerSet, an entity search engine build on top of Wikipedia, and DBpedia, a structured collection of facts about Wikipedia entities.

PowerSet is a commercial system aiming at finding entities and answer questions based on Wikiepdia. It is

> ... first applying its natural language processing to search, aiming to improve the way we find information by unlocking the meaning encoded in ordinary human language.[2]

[2]http://www.powerset.com/about

	P@10	MAP
Powerset-XER07	0.1261	0.0791
Powerset-XER07 (only XER07 results)	0.2196	0.1377

Table I
PowerSet Effectiveness on XER07 topics

Similarly to our proposal, this system exploits NLP techniques to answer queries and questions about entities. In order to understand its search quality, we run a set of queries from a standard ER benchmark such as INEX XER 2007 [11] on PowerSet. In this benchmark 25 ER topics and their relevance assessments using the Wikipedia 2006 corpus are available. In Table I Precision at 10 retrieved entities and Mean Average Precision (see [1] for a definition of these measures) for these topics are presented. As PowerSet is using the current Wikipedia while the used benchmark has been constructed on a snapshot from 2006, we also computed PowerSet effectiveness by ignoring results which are not present in the benchmark. It is possible to see that posing ER queries to this system, the quality of the results is not very high.

A different approach for finding entities is to create a structure repository of knowledge about entities and search on top of it. One such repository is DBpedia[3] which is a structured representation of entities in Wikipedia. It is

> a community effort to extract structured information from Wikipedia and to make this information available on the web. DBpedia allows you to ask sophisticated queries against Wikipedia[4]

One problem of such system is that for effectively posing queries to it, a client has to create a structured (e.g., SPARQL) query in order to get the result set. This is, of course, not very desiderable for a standard web user used to pose simple keyword queries and get back a list of results. For example, the query "Books written by Friedrich Nietzsche" would be translated in

```
SELECT ?title
WHERE {?title
<http://dbpedia.org/property/author>
<http://dbpedia.org/resource/Friedrich_Nietzsche> .

 ?title rdf:type
 <http://dbpedia.org/class/yago/Book106410904> .
 }
```

which, when posed to the SPARQL endpoint of DBpedia[5], returns as result set:

- http://dbpedia.org/resource/Nietzsche_contra_Wagner
- http://dbpedia.org/resource/On_the_Genealogy_of_Morality
- http://dbpedia.org/resource/The_Birth_of_Tragedy
- http://dbpedia.org/resource/The_Gay_Science

[3]http://dbpedia.org/
[4]http://wiki.dbpedia.org/About
[5]http://dbpedia.org/sparql

- http://dbpedia.org/resource/Twilight_of_the_Idols

all being relevant results.

We have described two possible approaches for finding entities on the web. The first enables users to pose natural language queries about the entities they want to retrieve which currently shows a low search quality. The second, similarly to a database, requires users to spend some effort in formulating their queries with the benefit of providing results which are relevant.

IV. Search Engine Architecture

In this section we present two possible architectures of an ER system: (1) Having a standard document centric index like available in current web search engines where we process the resulting documents at query time to extract relevant entities – Section IV-A – and (2) Building an entity centric index where all information about entities is compiled at indexing time – Section IV-B.

A. Entity Extraction at Query Time

Figure 1 presents the architecture of an ER system which uses as data source an existing document index like in current web search engines. This index was created by crawling web pages and each document in the index corresponds to a web page. The documents are indexed as terms and no entity specific information is extracted. Thus, any web search index (e.g. Yahoo!, Google, etc.) can be used by the ER system as underlying information. The modules inside the ER system are used for bridging ER specific queries and the document index, and are:

- A *Query Preprocessor* which parses the user query given as text with no additional annotations and parses it into a ER suited query. More about the query preprocessing can be found later in Section V-B.
- An *Entity Annotator* which can take a textual document as input and outputs a list of entities and their attributes contained in that document. This process is described in Section V-A.
- The *Ranking* mechanism creates then a suited ranking of entities given the output of the Query Preprocessor and the Entity Annotator. Section V-C discusses possible ranking approaches.

The entire ER process of using an existing document index is presented as Algorithm 1. The input of the system is a plain textual query given by the user and the output will be a ranked list of entities, i.e. not entire web pages. First the system uses the Query Preprocessor to transform the textual user query into a ER query suited for the system (Step 1). The class of the sought entity and the desired entity attributes are extracted from the textual query and compiled into an ER query (see Section V-B for more information). The query is then run against the existing (possibly external) document index taking into account the differences between entity class and its attributes (Step 2). The document index will return a ranked list of documents (Step 3). This list is then processed by the Entity Annotator (Step 4) and entities in the pages are extracted (see also Section V-A). This resulting list of entities is then ranked by the Ranking mechanism (Step 5, more in Section V-C) and the final ranked list of entities is presented to the user.

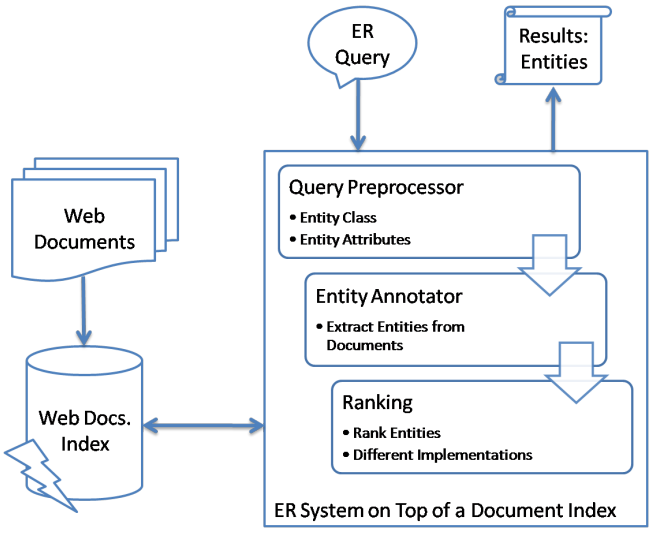

Figure 1. Architecture of entity ranking system using an existing document index.

Alg. 1. Entity Ranking Using an Existing Document Index
Input: User ER query q_t (Web-style textual query)
Output: Ranked list of entities R_e

1: Transform user textual query q_t into proper ER query q_e
1a: Annotate entity class $q_{e_{cls}}$ and entity attributes $q_{e_{attr}}$ in q_t
1b: Create ER query as $q_e = q_{e_{cls}} \cup q_{e_{attr}}$
2: Run user ER query q_e on web document index
3: Retrieve top n results R_d^n from the index
4: Extract and annotate entities R_e in R_d^n
5: Rank R_e according to relevancy

The advantage of such a system is that it can reuse an already existing web document index and only needs to add pre- and post-processing steps. Inconveniences are a short latency due to the processing overhead as well as the fixed document index which is seen as a black box and cannot be tuned for ER specific purposes.

B. Entity Centric Index

The second proposed architecture is the creation of an entity centric index and then the usage of such an index to directly retrieve entities. Figure 2 shows the entity index creation process and Figure 3 presents the system architecture at query time. The modules used in the architecture

are mainly the same as in Section IV-A but are being used differently.

The entity centric index does not contain web pages as documents but instead has each entity as a separate document in the index. Thus, at query time, the query can be matched directly to the entity class and attributes. The result is directly a ranked list of entities.

The indexing process splits and combines text from web pages in order to attach accurate information to each entity. Each "document" (i.e. entity) will contain text excerpts from different web pages where it is mentioned. Each web page which contains multiple entities will be used as information source for them. For example, the web page at http://www.mexico.com contains entities such as *Acapulco*, *Cancun*, *Cozumel*, *Guadalajara*, where each entity can have textual descriptions like "*Guadalajara* is the birthplace of some of *Mexico*'s best-known cultural exports" – in this case *Guadalajara* and *Mexico* will share parts of the description.

Entities are found in the text by the Entity Annotator, as described in Section V-A. The information extracted from a web page which is attached to an entity can be implemented in several ways, e.g., the sentence containing the entity, a window of w words around the entity, by syntactic or semantic boundaries, etc. The entity class can also be extracted by detecting several forms of the verb "to be", e.g., "Mexico City is the capital of Mexico" will result in the system perceiving *Mexico City* as a *capital*. The entity attributes will then be defined as the rest of the analyzed text excerpt.

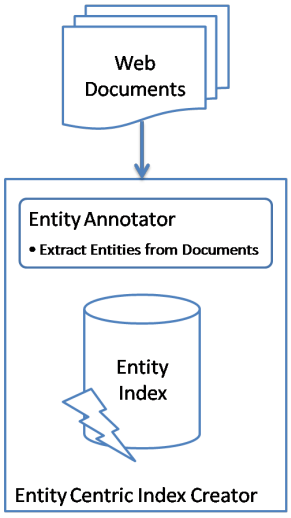

Figure 2. Architecture for an entity index creation system

As shown in Algorithm 2, at indexing time a web corpus is processed and an entity centric index is created. Entities in each web document are annotated by the Entity Annotator (Step 1a, see Section V-A). Text around the annotated entity, $e_{j_{text}}$, is extracted from the original web document as, e.g., the sentence in which e_j appears or a window of w words around e_j. The string $e_{j_{text}}$ is then split into information about the entity class $e_{j_{cls}}$ and entity attributes $e_{j_{attr}}$. Having all this information we can update the index using separate fields for $e_{j_{cls}}$ and $e_{j_{attr}}$.

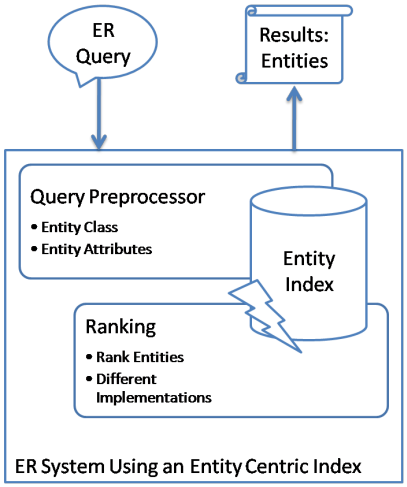

Figure 3. Architecture of entity ranking system using an entity centric index

Alg. 2. Creation of an Entity Centric Index
Input: List of web documents W
Output: Entity centric index

1: For each document $w_i \in W$
1a: Annotate entities E in w_i
1b: For each $e_j \in E$
1b1: Extract textual information $e_{j_{text}}$ about e_j
1b2: Annotate entity class $e_{j_{cls}}$ and entity attributes $e_{j_{attr}}$ from $e_{j_{text}}$
1b3: Update index with e_j

Algorithm 3 is run at query time using and transforming the user plain textual query on the entity centric index. The query preprocessing step (Step 1) is identical to Algorithm 1: the user query is transformed into a proper ER query q_e. The difference to Algorithm 1 is that by running q_e directly on the entity index we skip the entity extraction step and can incorporate the ranking scheme into the similarity function of the index. Thus, we are directly presented with the resulting ranked list of entities.

Alg. 3. Entity Ranking Using an Entity Centric Index
Input: User ER query q_t (Web-style textual query)
Output: Ranked list of entities R_e

1: Transform user textual query q_t into proper ER query q_e
1a: Annotate entity class $q_{e_{cls}}$ and entity attributes $q_{e_{attr}}$ in q_t
1b: Create ER query as $q_e = q_{e_{cls}} \cup q_{e_{attr}}$

2: Run user ER query q_e on entity index
3: Retrieve ranked entities R_e

This would be the preferred strategy for creating an ER system: it runs faster as no post-processing of the results is required and the ranking is implemented directly in the search engine. Disadvantages lie in the index maintenance: when processing one web document we have to update all index entries of all entities appearing in that page.

V. Entity Retrieval Components

The search process is split up into different phases. In this section we describe the components of the proposed architectures. At query time or during the index generation phase we extract entities and store them in an index for later retrieval. In the second phase we process the query of a user identifying the type of entities he/she is looking for. The final phase consists of retrieving suitable entities from the index and ranking them according to various strategies.

A. Entity Extraction

Entity Extraction, either during index generation time or at query time, facilitates finding appropriate entities. The authors of [21] describe a way to annotate documents with Wikipedia concepts. This, together with an hierarchy of concepts, can be used to find entities corresponding to the information need of a user. Since entities do not have a simply identifiable characteristic, the extraction or annotation becomes very difficult. Consider entities like "movie titles", "words that describe movement" or "alcoholic beverages". There is no straight forward way of identifying them looking only at documents. A very large lexicon, like Wikipedia, can be a starting point to find any kinds of entities. Each title of a Wikipedia page can be interpreted as an entity.

To reduce the complexity and scale of the problem we limited the entity extraction to finding Named Entities like Persons, Locations, or Organizations. Automatic systems are rather good finding Named Entities. In [8] an overview of results for the Named Entity Recognition (NER) task of the Message Understanding Contest (MUC) is given. The best systems are nearly as good as humans performing the task (93.39% f-measure vs. 97.60% and 96.95%). These systems use either rule-based linguistic approaches or statistical models and machine learning techniques. The difference to our setting is that they work on "clean" newspaper articles whereas we work on web pages. Web pages usually contain a lot of noise, like advertisement, navigational links, etc. Also the main content of a page does not necessarily consist of well structured English sentences. Often only lists or links are presented. This makes it difficult for NER systems to find all entities.

For our experiments, we used a Named Entity Recognition system shipped together with GATE [9]. Unfortunately, not all entities found manually in a subset of the corpus were detected by the system.

ID	Query	Detected type
129	Science fiction book written in the 1980	book
138	National Parks East Coast Canada US	park
139	Films directed by Akira Kurosawa	film
140	Airports in Germany	airport
141	Universities in Catalunya	university
142	Prizes named after people	prize

Table II
Sample of INEX entity queries

B. Query Processing

During query processing time we need to find the type of entity the user is interested in. This allows to explicitly search the generated index for this type of entity. It is also possible to incorporate an ontology to map rather general types like "musicians" to more specific types like "guitarists" or vice versa.

To identify the entity type in the query we use a set of rules. As a development set we used the INEX queries [11] for ER in Wikipedia. A few example queries and automatically identified entity types can be seen in Table II. The rules are based on results from various NLP components. Among them are a Part-of-Speech tagger, a Morphological Analyzer, and a Noun Phrase Extractor.

The query processing is also implemented in GATE [9] and the rules are written in JAPE, a grammar rules language [10].

C. Ranking Entities

Once we identified entities in web documents and processed the user query we can focus on ranking the set of identified entities. In the following we describe possible baseline approaches for ranking such set of entities. They are designed to be efficient in order to scale at web dimension datasets.

The first way of ranking the set R_e of retrieved entities is by their *frequency*. We count the number of times an entity $e_i \in R_e$ appears in the top N retrieved documents and we put first the entity which is most mentioned. The assumption is that when querying, for example, for "American Countries" the top retrieved pages will contain many occurrences of the relevant entities (e.g., "Mexico").

A second way is to rank entities by *document frequency*. We count the number of top N retrieved documents in which each entity $e_i \in R_e$ appears at least once. We then put first in the ranking the entity appearing in most documents. This is a measure of entity popularity that solves the problem when a page having many occurrences of the same entities is not a good indicator of its relevance. In this way we find entities which are mentioned in many retrieved documents.

A third approach is to take into consideration the *proximity* of the entities in the web documents. We define the

proximity of an entity e_i as a fixed number of characters before and after its occurrence. It is possible then to check the occurrence of query terms in the proximity of entities in web pages. We define as ranking mechanism that placing first the entity with most query terms in its proximity. The assumption is that the existence of passages of text like, e.g., "Mexico is an American country" will enable us to find "Mexico" as a result for the query "American Countries".

VI. Data Sources Analysis

In this section we describe an analysis on how people search for entities using current web search engines and how it is possible to obtain evidence for finding entities on the web.

A. Query Log Analysis

We computed statistics on the AOL query log [18] which consists of 20M web queries collected from 650k users of the AOL search engine over three months in 2006. This query log contains over 700k (200k unique) queries starting with "list of". Most popular "List of" queries are shown in Table III.

Frequency	"List of" query
12952	southwest airlines
6543	american airlines
5099	united airlines
4643	continental airlines
4369	free music downloads
4331	jokes
4090	delta airlines
3778	screen names
3764	airlines
3593	movies
3550	baby names
3467	music
3363	flowers
2961	hotels

Table III
MOST POPULAR "LIST OF" QUERIES FROM THE AOL QUERY LOG.

While not all of them are ER queries (e.g., "list of free music downloads" is a transactional query) it is possible to see that a big number of user queries are actually looking for a ranked list of entities on the web.

B. Google Coverage

We also performed experiments in order to understand how reliable current search engine results are for the task we intend to perform. We used the top 100, top 300 and all the available search results through the University Research Program for Google Search[6] constructing "or" (disjunctive) and "and" (conjunctive) queries out of the given search terms. We used as queries the titles of "List of" pages from Wikipedia (see Section VII-B). The maximum number of search results returned by Google API for a given query was, on average, 600. Our goal is to check whether relevant entities to an ER query are contained in top N retrieved documents. We computed such coverage for all the pages retrieved by Google, as well as excluding from the retrieved set pages from Wikipedia, and just "List of" pages of Wikipedia as we used as groundtruth Wikipedia entities (see Section VII-B). The percentages of relevant entities present in Google results are shown in the table IV.

Size	Type	"List of" pages	Wikipedia pages	Coverage
top 100	and	yes	yes	0.7925
	and	yes	no	0.7925
	and	no	no	0.6995
	or	yes	yes	0.4207
	or	yes	no	0.4207
	or	no	no	0.3781
top 300	and	yes	yes	0.8412
	and	yes	no	0.8410
	and	no	no	0.7728
	or	yes	yes	0.4869
	or	yes	no	0.4869
	or	no	no	0.4548
all	and	yes	yes	0.8562
	and	yes	no	0.8554
	and	no	no	0.7801
	or	yes	yes	0.5558
	or	yes	no	0.5558
	or	no	no	0.5371

Table IV
GOOGLE COVERAGE OF RELEVANT ENTITIES.

We can see that when posing a conjunctive query to the Google search engine and considering top 100 results (this constraint is necessary for annotation efficiency reasons) we obtain a coverage of 70% when not considering any page from Wikipedia as they are used as groundtruth for checking relevant results. A disjunctive query built out of the user query, while retrieving more results, results in a lower coverage in top N retrieved documents. This result tells us that by using an optimal ranking algorithm the best obtainable recall is 0.7 given the constraints of the underlying system.

VII. Benchmarks

In this section we describe two possible benchmarks that can be used for evaluating the quality of ER results. The first one is a standard set of queries and groundtruth built on top of the Wikipedia corpus. The second one is a set of queries and relevant entities built out of Wikipedia "List of" pages.

A. INEX XER Benchmark

In the context of the INitiative for Evaluation of XML Retrieval (INEX) a benchmark for ER has been built in the 2007 and 2008 editions. A total set of 60 ER queries (25+35) have been defined and relevance assessment performed over the two editions[7]. INEX uses as underlying document col-

[6]http://research.google.com/university/search

[7]http://www.l3s.de/~demartini/XER08/

lection Wikipedia [14] and, for the Entity Ranking Track, the assumptions is that each entity is represented by its Wikipedia article. Thus, ER systems have to provide a ranked list of articles as response to a ER query. This collection can be used for evaluating effectiveness of ER systems on the web by matching title of relevant articles to the retrieved results.

B. Wikipedia Lists as Benchmark

Additionally to INEX, we build a benchmark using as gold standard for the evaluation the "List of" pages from Wikipedia. The title of such a page, after removing the first two terms, is used as an ER query (e.g., "lakes in Arizona"[8]). The titles of the Wikipedia pages that are linked to from such a "List of" page are considered to be relevant results (e.g., "Canyon Lake", "Lake Havasu", ...). In order to use only queries that are more similar to typical web queries in terms of length, we kept only those queries that consisted of 2 or 3 terms apart from "List of". Thus we had 17,110 pages out of the total of 46,867 non-redirect "List of" pages. In order to construct a benchmark that is as realistic as possible, by analysing a search engine query log we filtered queries to only keep those which were posed at least 100 times and had at least 5 clicks on results. We are thus left with 82 queries for ER evaluation.

VIII. Conclusion

In this paper we proposed an architecture for a system able to find entities on the web as response to a user query. We detailed the overall architecture as well as possible solutions for the different steps of the search process. Additionally, we presented results from an initial experimental feasibility study of state-of-the-art tools in the field of Information Extraction and Information Retrieval.

As future work we want to deploy such system and evaluating its overall effectiveness using one of the presented benchmarks for Entity Retrieval.

Acknowledgement

This work is partially supported by the EU Large-scale Integrating Projects OKKAM[9] - Enabling a Web of Entities (contract no. ICT-215032), PHAROS[10] (IST contract no. 045035), and LivingKnowledge[11] (contract no. 231126)

References

[1] R. Baeza-Yates and R. Neto. *Modern Information Retrieval*. ACM Press, 1999.

[2] H. Bast, A. Chitea, F. Suchanek, and I. Weber. Ester: efficient search on text, entities, and relations. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 671–678, New York, NY, USA, 2007. ACM.

[3] P. Bouquet, H. Halpin, H. Stoermer, and G. Tummarello, editors. *Proceedings of the 1st international workshop on Identity and Reference on the Semantic Web (IRSW2008) at the 5th European Semantic Web Conference (ESWC 2008), Tenerife, Spain, June 2nd, 2008*, CEUR Workshop Proceedings. CEUR-WS.org, 2008.

[4] P. Bouquet, H. Stoermer, and B. Bazzanella. An entity name system (ens) for the semantic web. In S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, editors, *ESWC*, volume 5021 of *Lecture Notes in Computer Science*, pages 258–272. Springer, 2008.

[5] P. Bouquet, H. Stoermer, G. Tummarello, and H. Halpin, editors. *Proceedings of the WWW2007 Workshop I^3: Identity, Identifiers, Identification, Entity-Centric Approaches to Information and Knowledge Management on the Web, Banff, Canada, May 8, 2007*, volume 249 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[6] T. Cheng and K. C.-C. Chang. Entity search engine: Towards agile best-effort information integration over the web. In *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 7-10, 2007, Online Proceedings*, pages 108–113. www.crdrdb.org, 2007.

[7] T. Cheng, X. Yan, and K. C.-C. Chang. Entityrank: searching entities directly and holistically. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 387–398. VLDB Endowment, 2007.

[8] N. Chinchor. Overview of MUC-7. In *Proceedings of the Seventh Message Understanding Contest*, Fairfax, VA, USA, 1997.

[9] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.

[10] H. Cunningham, D. Maynard, and V. Tablan. JAPE: a Java Annotation Patterns Engine (Second Edition). Research Memorandum CS–00–10, Department of Computer Science, University of Sheffield, Nov. 2000.

[11] A. P. de Vries, A.-M. Vercoustre, J. A. Thom, N. Craswell, and M. Lalmas. Overview of the inex 2007 entity ranking track. In Fuhr et al. [15], pages 245–251.

[12] G. Demartini, C. S. Firan, T. Iofciu, R. Krestel, and W. Nejdl. A model for ranking entities and its application to wikipedia. *Web Congress, Latin American*, 0:29–38, 2008.

[13] G. Demartini, C. S. Firan, T. Iofciu, and W. Nejdl. Semantically enhanced entity ranking. In J. Bailey, D. Maier, K.-D. Schewe, B. Thalheim, and X. S. Wang, editors, *WISE*, volume 5175 of *Lecture Notes in Computer Science*, pages 176–188. Springer, 2008.

[8] http://en.wikipedia.org/wiki/List_of_Arizona_lakes
[9] http://fp7.okkam.org/
[10] http://www.pharos-audiovisual-search.eu/
[11] http://livingknowledge-project.eu/

[14] L. Denoyer and P. Gallinari. The Wikipedia XML corpus. *ACM SIGIR Forum*, 40(1):64–69, 2006.

[15] N. Fuhr, J. Kamps, M. Lalmas, and A. Trotman, editors. *Focused Access to XML Documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007, Dagstuhl Castle, Germany, December 17-19, 2007. Selected Papers*, volume 4862 of *Lecture Notes in Computer Science*. Springer, 2008.

[16] J. Gaugaz, J. Zakrzewski, G. Demartini, and W. Nejdl. How to trace and revise identities. In L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou, and E. P. B. Simperl, editors, *ESWC*, volume 5554 of *Lecture Notes in Computer Science*, pages 414–428. Springer, 2009.

[17] M. C. Jordi Atserias, Hugo Zaragoza and G. Attardi. Semantically Annotated Snapshot of the English Wikipedia. In E. L. R. A. (ELRA), editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may 2008.

[18] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *InfoScale '06: Proceedings of the 1st international conference on Scalable information systems*, page 1, New York, NY, USA, 2006. ACM.

[19] J. Pehcevski, A.-M. Vercoustre, and J. A. Thom. Exploiting locality of wikipedia links in entity ranking. In C. Macdonald, I. Ounis, V. Plachouras, I. Ruthven, and R. W. White, editors, *Advances in Information Retrieval , 30th European Conference on IR Research, ECIR 2008, Glasgow, UK, March 30-April 3, 2008. Proceedings*, volume 4956 of *Lecture Notes in Computer Science*, pages 258–269. Springer, 2008.

[20] R. Schenkel, F. M. Suchanek, and G. Kasneci. YAWN: A Semantically Annotated Wikipedia XML Corpus. In A. Kemper, H. Schöning, T. Rose, M. Jarke, T. Seidl, C. Quix, and C. Brochhaus, editors, *BTW*, volume 103 of *LNI*, pages 277–291. GI, 2007.

[21] P. Schönhofen. Annotating documents by wikipedia concepts. *Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*, 1:461–467, 2008.

[22] T. Tsikrika, P. Serdyukov, H. Rode, T. Westerveld, R. Aly, D. Hiemstra, and A. P. de Vries. Structured Document Retrieval, Multimedia Retrieval, and Entity Ranking Using PF/Tijah. In Fuhr et al. [15], pages 306–320.

[23] H. Zaragoza, H. Rode, P. Mika, J. Atserias, M. Ciaramita, and G. Attardi. Ranking Very Many Typed Entities on Wikipedia. In M. J. Silva, A. H. F. Laender, R. A. Baeza-Yates, D. L. McGuinness, B. Olstad, Ø. H. Olsen, and A. O. Falcão, editors, *CIKM*, pages 1015–1018. ACM, 2007.

[24] C. Zirn, V. Nastase, and M. Strube. Distinguishing between instances and classes in the wikipedia taxonomy. In S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, editors, *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*, volume 5021 of *Lecture Notes in Computer Science*, pages 376–387. Springer, 2008.