ListCreator: Entity Ranking on the Web

Alexandros Komninos
Department of Electrical and Computer Engineering
Democritus University of Thrace
Xanthi 67100, Greece
alexkomn@ee.duth.gr

Avi Arampatzis
Department of Electrical and Computer Engineering
Democritus University of Thrace
Xanthi 67100, Greece
avi@ee.duth.gr

Abstract— In this paper, we present a web application for entity ranking. The application accepts as input a query in natural language and outputs a list of the most relevant entities according to the query. The system uses web documents as data and performs extraction, formatting and ranking of entities in real time. An experiment is conducted to determine the most efficient ranking method among six alternatives. The experiment suggests that the total frequency of an entity in a retrieved set of documents has less to say on the entity's relevance than the number of retrieved documents it occurs in. Furthermore, for small retrieved sets such as the top-10, document rank information seems to play a little role.

Keywords-web entity ranking; entity search; information retrieval

I. INTRODUCTION

Search engines answer user queries by returning ordered lists of documents. In many occasions though, users are not searching for documents but for some more specific information. This information is often *named entities*. The term named entity is used for anything that has a distinct existence and can be characterized by a name, so it can refer to people, companies, products, etc. The need for retrieving named entities as query answers has led to research for systems that can recognize and return this type of information instead of whole documents.

ListCreator [1] is a web application that can answer user queries for entities of three categories: persons, locations and organizations. The application uses as data web documents that match to the submitted query. The ranking of the entities found in these documents is achieved by statistical information retrieval methods, taking advantage of the common information among them. The results are returned to the user as a ranked list of all the relevant entities that the application managed to extract.

The contribution of this paper is twofold. First, we build an online prototype as proof-of-concept for entity ranking using information retrieval methods. Such methods are simple and fast, and therefore suited for an online application. They are also less-limited than ontology-based methods since web documents are used as data. Second, we evaluate several entity ranking methods based on several combinations of statistical quantities corresponding to different hypotheses on language use of document authors and search engine document ranks.

The rest of this paper is organized as follows. In Section II, we review related work. In Section III, we give a detailed description of ListCreator's methods and architecture. In Section IV, we perform a small experiment comparing different methods for ranking entities. Conclusions are drawn in Section V together with directions for further research and improvements.

II. RELATED WORK

Entity ranking has a lot in common with automatic question answering, since the answer to a question is often a named entity or in some cases a list of named entities. An approach that led to good results is using many different text snippets that are expected to contain the desired answer, and using the common information among them to accurately locate it [2]. INEX (INitiative for the Evaluation of XML retrieval) started in 2007 an entity ranking track which was run until 2009. The purpose of this track was the creation of entity ranking systems that could rank relevant entities that had a Wikipedia page. A common approach among many teams was to find a relevant document for each candidate entity and then rank these entities according to the relevance of the document to the query, using document retrieval methods [3][4]. TREC (Text REtrieval Conference) run from 2009 to 2011 a track for related entity finding in the web. The purpose was finding relevant entities to a query that engage in a given relationship with a source entity. The relevance of the candidate entities was determined by many participants by the co-occurrence with the source entity in web documents [5][6].

A different approach is using information extraction techniques to construct structured date from text by extracting facts about entities [7][8]. This requires natural language processing, for example a syntactic parser, and is achieved using machine learning methods. Since applying machine learning to large volumes of text has great computational cost, the above systems constructed a database of relations between entities offline. The database is then queried for relevant entities by the user at runtime. An alternative is using data sets of existing ontologies constructed either manually or automatically using information extraction [9][10].

ISBN: 978-1-61208-227-1

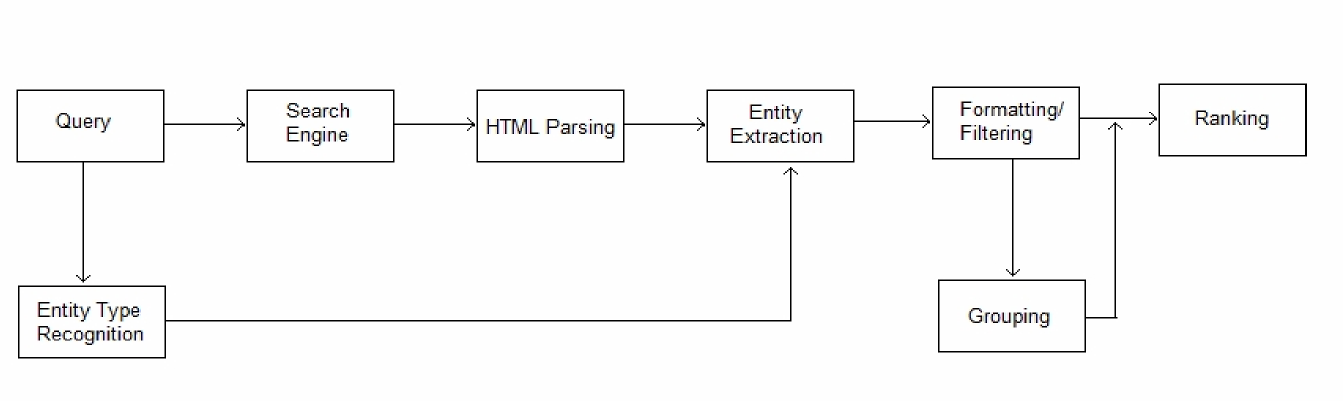


Figure 1 . The System’s Components and Dataflow

Our approach is based on information retrieval methods leveraging the large data volume of the web. The difference from INEX and TREC approaches is that a descriptive document for the entities, like Wikipedia pages or personal homepages, or a source entity is not required. The information retrieval methods use statistical measures based on a bag of words model. These methods cannot identify complex relations in sentences like the methods using machine learning, but can process large amounts of text very efficiently and are proven effective by traditional search engines. The advantage over methods using machine learning is that the data can be processed in real time so the results are not limited by the relations recorded in a database. The question answering systems that use common information between different documents are closer to our approach, but they only use term frequency as a measure since their goal is not ranking but verifying the correctness of results produced by an extraction process. We evaluated several methods for ranking, and the results suggest, in contrast to question answering systems, that term frequency is not a strong indication of an entity’s relevance, as we will see in Section IV.

III. SYSTEM DESCRIPTION

The system’s architecture is depicted in Figure 1. The components for formatting, filtering, grouping and ranking of entities are all coded in JAVA [11]. The user web interface is coded in HTML [12], JavaScript [13], and PHP [14].

A. *The Application Website*

The central webpage consists of an input form for the user’s query and gives the option to determine the type of entity (person, location, organization) that she is searching for. The default option is “auto” which corresponds to automatic recognition of the entity type.

The automatic recognition feature uses a list of about 100 keywords for the location type and about 50 keywords for the organization type. The collection of keywords is based on WordNet categories [15]. The system checks for the appearance of any of those keywords in the submitted query and if they exist it is assumes the user is searching for the corresponding entity type. If none of the keywords appear the system assumes that the user is searching for persons.

The submission of a query calls the main application and the output is presented in the results webpage with the use of PHP. Each result is linked to a corresponding Wikipedia page (if it exists) so that the user can get more information. The results webpage also gives as references links to the web documents that the entities were extracted from. A results page is presented in Figure 2.

B. *The Search Engine*

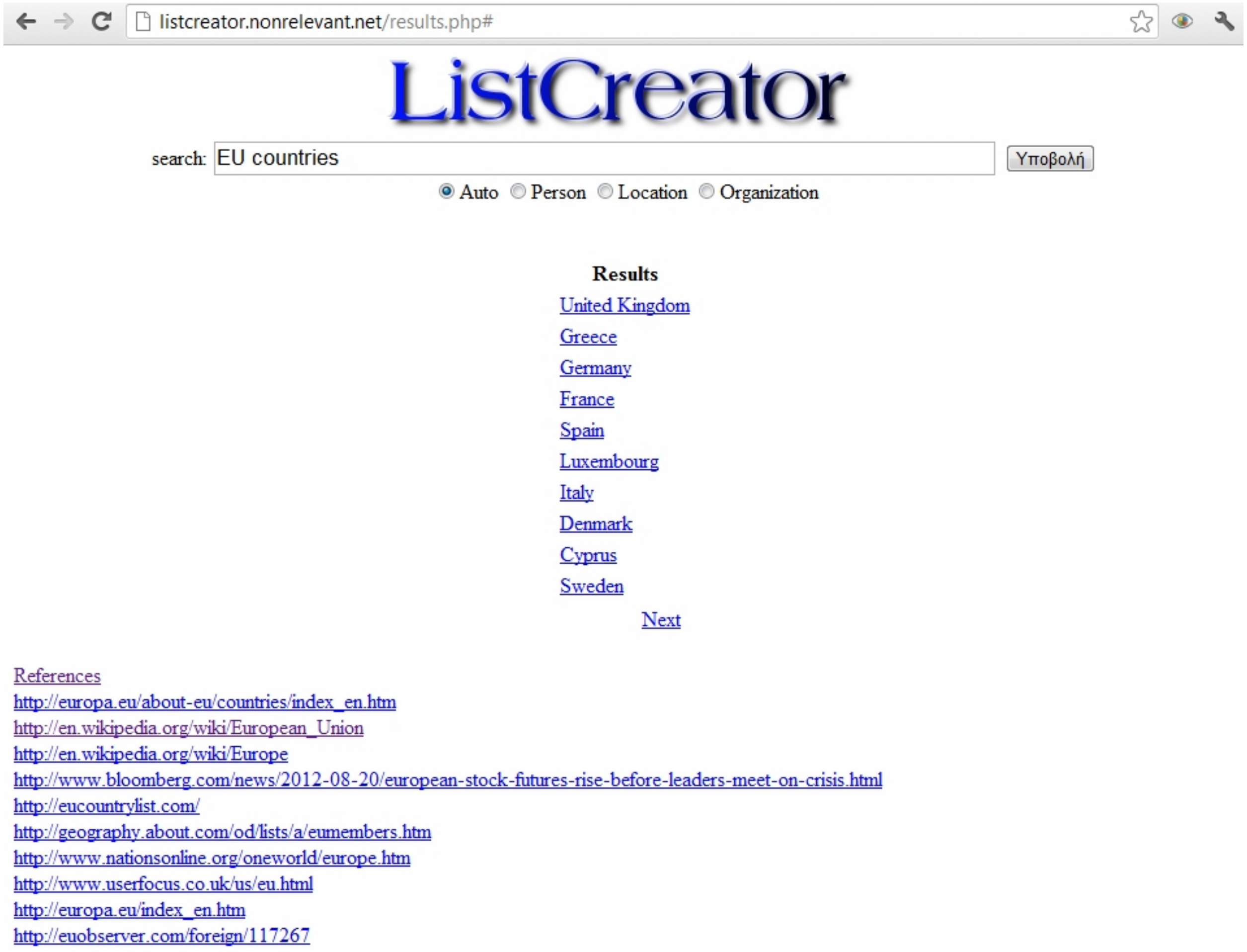
The search engine is a very important component of the system since it provides all the data in the form of documents for extracting and ranking the entities. The application essentially functions as a front-end in a search engine. In the current version the search engine used is the Yahoo! BOSS API [16]. Google and Bing were also tested with similar results but Yahoo was chosen because it combines good results with an easy to use API.

The user’s query is sent to Yahoo! API without being changed and the results are returned in JSON (JavaScript Object Notation) format. The system asks for only the top-N results. Through some testing we empirically determined that N=10 retrieves enough information while, at the same time, keeps the computational cost low enough for a real time application.

C. *Entity Extraction*

In this stage, the system recognizes the entities in the documents and determines their type. For this purpose the Stanford NER (Named Entity Recognizer) is used [17]. Stanford NER is a system for entity extraction from text coded in JAVA and distributed with GNU general public license [18] for research and education purposes. The entity recognition is done with a classifier, an algorithm that assigns words in specific categories. The categories supported by the classifier are person, location and organization.

Classification is a supervised machine learning technique. The algorithm uses hand-annotated text to construct statistical rules that can find and determine the category of names in documents. The Stanford NER classifier [19] is based on the statistical model CRF



search: EU countries

● Auto ○ Person ○ Location ○ Organization

Results

United Kingdom
Greece
Germany
France
Spain
Luxembourg
Italy
Denmark
Cyprus
Sweden

Next

References
http://europa.eu/about-eu/countries/index_en.htm
http://en.wikipedia.org/wiki/European_Union
http://en.wikipedia.org/wiki/Europe
http://www.bloomberg.com/news/2012-08-20/european-stock-futures-rise-before-leaders-meet-on-crisis.html
http://eucountrylist.com/
http://geography.about.com/od/lists/a/eumembers.htm
http://www.nationsonline.org/oneworld/europe.htm
http://www.userfocus.co.uk/us/eu.html
http://europa.eu/index_en.htm
http://euobserver.com/foreign/117267

Figure 2. A results Page of the Application

(Conditional Random Field) [20] and comes trained on American and British news articles. The classification process offers some very useful filtering of the entities. The usage of a NER system was considered more suitable for unknown data since it identifies entities by their context in documents, in contrast with a dictionary based approach. It is limited though in the three general entity categories.

In order to extract entities from a web document, the HTML tags have to be removed. For the HTML parsing the JSOUP HTML Parser is used [21]. JSOUP is an open source parser also coded in JAVA that can handle html code with errors

D. *Formatting and Filtering*

Each entity can appear in a document in many different ways. A person's name for example can first appear with its full name and later be referred with just the last name. In order to achieve a cleaner better ranking in the next stage, the system must recognize which names correspond to the same entity, a task called *co-reference resolution*, and then assign to all of them the same canonical name. The results of this stage are also important for the final presentation since names should appear with all details and avoid listing the same names more than once. The processing of names comes in two steps. In the first step, each entry is formatted and in the second step the names referring to the same entity are grouped taking in consideration the whole set of extracted names.

The basic processing of the first step is converting the names to proper case, i.e. converting the first letter in uppercase and the rest in lowercase. For organization names with less than four letters, all of them are converted to uppercase. Furthermore, the candidate entities are filtered using an exception list. The exception list consists of about 20 entries that correspond to certain names that are often misclassified by Stanford NER. These names are popular websites (Wikipedia, Facebook, Twitter, etc.) that are classified as locations and some acronyms like FAQ, ISBN that are classified as organization. Using this exception list the results from the extraction stage are improved. Another exception list used contains all the country names. This list is checked for search of location type entities because country names appear in large numbers in documents about locations and they can have negative influence on ranking. This exception list is not used when the user is searching for country names.

The grouping of entities that happens in the second step is rule-based and is achieved by comparing each entry with all others. The system checks if an entry forms part of another in word level, and then it is substituted by its complete name. For example, the entries John Kennedy, Kennedy, John F. Kennedy and John Fitzgerald Kennedy are all grouped and substituted by the last form. In order to avoid grouping into names that may be misspelled, or into a concatenation of two names, the substitution takes place when an entry appears more than once. The grouping step is not applied for queries asking for names of countries, cities and organizations. Country and city names usually do not appear in different forms, while organization names have lots of variance to be grouped with simple rules that often lead to errors.

The above method of grouping gives good results and greatly improves performance, but in some cases the correct grouping of entries cannot be determined. Such is the case of two different candidate entities with the same last name and an entry containing this last name alone. A possible improvement could be the usage of a system that accomplishes co-reference resolution utilizing machine learning but such an approach would increase computational cost.

E. *Entity Ranking*

The ranking algorithm makes usage of statistical methods of information retrieval. The input in this stage is 10 lists of candidate entities, each one corresponding to the names extracted from each document the search engine provides. The entities are then ranked according to the formula:

$$score = \sum_{j=1}^{df} (N + 1 - r_j)$$

where j is the document an entity appears in, df is the number of the top-N documents that mention an entity, N is the total number of retrieved documents and in the current version is always equal to 10, r is the rank of the retrieved document according to the search engine and has a value from 1 to 10. The formula is based on the preferential voting method *Borda Count*. According to the formula, an entity that appears only in the first document gets 10 points, if it appears on the first and second document, it will get 10 plus 9 points, etc. Entities with higher score are considered more relevant to the query. This ranking formula was chosen after the small experiment that will be described in the next section.

IV. EXPERIMENT

The proposed ranking method tries to solve a problem that resembles the reverse procedure of finding relevant documents to a query. Instead of searching for documents relevant to some terms, it utilizes a small collection of documents (10 in our case) with a common subject and searches for terms (in this case named entities) that are important for this collection. The quantities that were considered useful for the ranking according to the above line of thinking are:

- The total number of occurrences of each entity in the collection of documents (*f*). The higher the frequency of an entity, the more confidence we have in its correctness and importance.
- Document frequency (*df*), which corresponds to the number of distinct documents where each entity occurs. This quantity shows the common information between documents. Assuming that all documents are equally relevant to the submitted query, the names that occur in most documents would also be the most relevant.
- The rank of documents that an entity appears in, according to the search engine (*r*). Taking into account this quantity the documents are no longer treated as equally relevant.

In order to find which of these quantities or which combination of them is more accurate for ranking entities, the following six ranking formulae were compared in the experiment:

$$score = \log(df) \tag{1}$$

$$score = \log(f) \times \log(df) \tag{2}$$

$$score = f \times \log(df) \tag{3}$$

$$score = \sum_{j=1}^{df} (N + 1 - r_j) \tag{4}$$

$$score = \sum_{j=1}^{df} \log(1 + f_j)(N + 1 - r_j) \tag{5}$$

$$score = \sum_{j=1}^{df} f_j (N + 1 - r_j) \tag{6}$$

In all formulae above, j is the document, N is equal to 10, f_j is the number of occurrences of an entity in document j.

There are two opposite hypothesis regarding the frequency of a term and the importance that has for a document [22]. According to the *verbosity hypothesis*, multiple occurrences of a term are not really important because the document is more verbose: the author just used more words to express the same meaning. According to the *scope hypothesis* though, a document's author uses a specific term more times because she has more information to share on this subject.

Equations (1), (2) and (3) do not take into account the ranking of documents, while equations (4), (5) and (6) do. The other difference between the above equations is the weight given to the term frequency of each entity. Equations (1) and (4) are based on the verbosity hypothesis, while (3) and (6) are based on the scope hypothesis. In equations (2) and (5) the logarithm of the term frequency is used. The logarithm in these equations acts as a dampening factor so that the equations represent a middle ground between the two hypotheses.

The evaluation of information retrieval systems is done with some specific measures. For evaluating the performance of the various ranking formulae the measures Precision-at-10 (P@10) and R-Precision were used. P@10 shows the number of relevant answers within the top-10 results. While it does not take into account the ranking of the correct answers, it offers an easy interpretation of results and does not require knowledge of the total of correct answers (recall)

TABLE I. P@10 AND R-PRECISION MEASURES FOR THE SIX RANKING EQUATIONS AVERAGED OVER THE 30 EVALUATION QUERIES.

Ranking Equations	P@10	R-Precision
$\log(df)$	0.4733	0.4209
$\log(f_{tot}) \times \log(df)$	0.4633	0.4306
$f_{tot} \times \log(df)$	0.4433	0.4294
$\sum_{j=1}^{df}(N+1-r_j)$	**0.49**	0.4216
$\sum_{j=1}^{df}\log(1+f_j)(N+1-r_j)$	0.4767	**0.4463**
$\sum_{j=1}^{df} f_j(N+1-r_j)$	0.41	0.4024

to be computed. Furthermore, the p@10 measure is suitable for web retrieval since most users usually check only the top-10 results. A problem with P@10 is that it does not average well across queries, since the number of correct answers has great variance. R-Precision shows the number of relevant answers within the top-R results, where R is the total number of relevant answers in the set. R-precision overcomes the problem of variance in the number of correct answers [23].

Each ranking formula was tested on 30 queries based on the evaluation topics for entity ranking systems from INEX 2009 and TREC 2010. The usage of these topics was not intended to compare the results of this system to those participating on these tracks, but to evaluate on a set of queries with several degrees of difficulty. The queries were slightly modified to be more specific, since they originally were followed by a narrative for more details. Most of them ask for entities that satisfy more than one condition. In order to accept an entity as relevant, it had to satisfy all the conditions of the query. The correctness of the results was manually checked. The experimental results can be seen on Table 1. The query set is on Table 2.

The six ranking methods achieved similar results, so it is not clear which one is better. The P@10 measure indicates that term frequency does not improve ranking results. As the influence of term frequency increases, P@10 decreases, suggesting that verbosity hypothesis works better for entity ranking. Equations (2) and (5) that represent the middle ground, achieve a higher R-Precision. Assuming the user wants to find all relevant results this method will work better. The reason that (4) is used in the prototype is we expect users to be mostly interested in the first 10 results. Further increase of term frequency influence on ranking, as the scope hypothesis suggests, does not offer any improvement. The ranking of documents does not have a great impact, as expected with a small set of 10 documents, but offers some small improvement except for the case of (6).

TABLE II. THE 30 EVALUATION QUERIES USED IN THE EXPERIMENT

Evaluation Queries
Pacific navigators Australia explorers
List of countries in World War Two
Nordic authors known for children's literature
Makers of lawn tennis rackets
National capitals situated on islands
Poets winners of Nobel prize in literature
Formula 1 drivers that won the Monaco Grand Prix
Formula One World Constructors' Champions
Italian Nobel prize winners
Musicians who appeared in the Blues Brothers movies
Swiss cantons where they speak German
US Presidents since 1960
Countries which have won the FIFA world cup
Toy train manufacturers that are still in business
German female politicians
Actresses in Bond movies
Star Trek Captains characters
EU countries
Record-breaking sprinters in male 100-meter sprints
Professional baseball team in Japan
Japanese players in Major League Baseball
Airports in Germany
Universities in Catalunya
German cities that have been part of the hanseatic league
Chess world champions
Recording companies that now sell the Kingston Trio songs
Schools the Supreme Court justices received their undergraduate degrees
Axis powers of World War Two
State capitals of the United States of America
National Parks East Coast Canada US

The experiment also provided some insight in the system's function. First, we noticed the dependency of performance on the quality of retrieved documents. As expected, queries that resulted in many relevant documents had much more precision in results than others where they had fewer relevant documents. Another problem comes with queries that have a small amount of correct answers (e.g., Axis Powers of World War Two). Determining a cut-off threshold on result scoring so that only relevant results may appear on the list is a difficult task [24].

V. CONCLUSIONS AND FUTURE WORK

We presented a prototype of an online application for entity ranking that uses web documents as data and ranks the entities using information retrieval methods. The application uses various components for recognizing the query topic, retrieving documents, extracting entities and performing co-reference resolution before the ranking takes place. We experimented with and evaluated several combinations of statistical quantities for ranking entities.

The experiments showed that the combination of rank position for source documents along with a measure of the common information among them yields the best results for ranking. The total frequency of entities did not work very well, verifying the verbosity hypothesis. Furthermore, the experiments showed that using the large data volume of the Web along with a search engine for retrieving them, the system has almost no limitations in query handling.

The application currently supports search for persons, locations and organization. The search can be easily expanded to other types of entities like products, books and movie titles by incorporating them to the extraction stage. The ranking method is very fast but the overall speed of the application is currently confined by the extraction stage which uses machine learning. The necessary processing of this stage though could be done in advance by crawling for documents and extracting information in a similar way that search engines create their indices. With this modification the speed of the ranking method will be fully utilized.

REFERENCES

[1] ListCreator. [Online]. Available: http://listcreator.nonrelevant.net [retrieved: September 2012].

[2] J. Lin. “The Web as a Resource for Question Answering: Perspectives and Challenges”. *Proceedings of the third International Conference on Language Resources and Evaluation (LREC 2002)*, 2002.

[3] G. Dermatini, T. Iofciu, and A. P. de Vries. “Overview of the INEX 2008 Entity Ranking Track”. *Lecture Notes in Computer Science, Volume 5631/2009*, 2009, pp. 243-252.

[4] G. Dermatini, T. Iofciu, and A. P. de Vries. “Overview of the INEX 2008 Entity Ranking Track”. *Lecture Notes in Computer Science, Volume 6203/2010*, 2010, pp. 254-264.

[5] K. Balog, A. P. de Vries, P. Serdyukov, P. Thomas, and T. Westerveld. “Overview of the TREC 2009 Entity Track”. *Proceedings of the Eighteenth Text RErieval Conference (TREC 2009)*,.2009.

[6] K. Balog, A. P. de Vries, and P. Serdyukov. “Overview of the Trec 2010 Entity Track”. *Proceedings of the Nineteenth Text RErieval Conference (TREC 2010)*,.2010.

[7] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. “Open Information Extraction: the Second Generation”. *Proceedings of the Twenty-second International Joint Conference in Artificial Intelligence (IJCAI’11)*, 2011, pp 3-10..

[8] M.J. Cafarella, C. Re, D. Suciu, and O. Etzioni. “Structured Querying of Web Text Data: A Technical Challenge”. *Proceedings of the Third Conference on Innovative Data Systems Research (CIDR 2007)*, 2007, pp.225-234.

[9] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. “NAGA: Searching and Ranking Knowledge”. *Proceedings of the Twenty-fourth International Conference on Data Engineering (ICDE 2008)*, 2008, pp. 953-962.

[10] E. Meij, M. Bron, L. Hollink, B. Huurnink, and M.D. Rijke,. “Mapping queries to the Linking Open Data cloud: A case study using DBpedia”. In J. Web Semantics. December 2011, pp.418-433.

[11] Java. [Online]. Available: http://www.java.com/en/ [retrieved: September 2012].

[12] HTML 4.01 Specification. [Online]. Available: http://www.w3.org/TR/1999/REC-html401-19991224/ [retrieved: September 2012].

[13] JavaScript. [Online]. Available: https://developer.mozilla.org/en-US/docs/JavaScript [retrieved: September 2012].

[14] PHP. [Online]. Available: http://www.php.net/ [retrieved: September 2012].

[15] Princeton University (2010). WordNet. [Online]. Available: http://wordnet.princeton.edu [retrieved: September 2012].

[16] Yahoo BOSS API. [Online]. Available: http://developer.yahoo.com/search/boss [retrieved: September 2012].

[17] J. R. Finkel, T. Grenager, and C. Manning. “Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling”. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, 2005, pp 363-370.

[18] GNU General Public License. [Online]. Available: http://www.gnu.org/licenses/gpl.html [retrieved: September 2012].

[19] Named Entity Recognition and Information Extraction [Online] http://nlp.stanford.edu/ner/index.shtml [retrieved: September 2012].

[20] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data”. *Proceedings of the Eighteenth International Conference on Machine Learning (ICML’01)*, 2001, pp.282-289.

[21] Jsoup: Java HTML Parser. [Online]. Available: http://jsoup.org [retrieved: September 2012].

[22] S. E. Robertson and S. Walker. “Some Simple Effective Approximations to the 2–Poisson Model for Probabilistic Weighted Retrieval”. *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994, pp. 345–354.

[23] C. Buckley and E. M. Voorhees. “Retrieval Evaluation with Incomplete Information”. *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2004, pp. 25-32.

[24] A. Arampatzis, J. Kamps, and Stephen Robertson “Where to Stop Reading a Ranked List? Threshold Optimization using Truncated Score Distributions.”. *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2009, pp. 524–531.

ISBN: 978-1-61208-227-1