Entity Ranking as a Search Engine Front-End

Alexandros Komninos
Department of Computer Science
University of York
York, YO10 5GH, UK
ak1153@york.ac.uk

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 eleven 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. Four algorithms are tested for estimating the correct amount of results in the ranked list and provide a threshold. The best results are achieved by the maximum entropy algorithm applied to the distribution of scores provided by a multiplicative combination of logarithmic entity frequency and document frequency.

Keywords - Web entity ranking, entity search, information retrieval, threshold optimization.

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, such as “German female politicians” or “Swiss Cantons where they speak French”. In this type of queries, users are looking for an answer consisting of semantically important units called *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 identify and return this type of information instead of whole documents.

Entity ranking is the process of finding and sorting entities according to their relevance to an information need. The difference from document retrieval is that it gives direct answers to a user’s query; therefore, it is an approach for data oriented search. Another important difference is that results can come from combining information from multiple sources instead of a single one.

Different methods have been used, either separately or in combination, for the purpose of entity ranking. Most of them come from the fields of information retrieval (IR) and natural language processing, especially information extraction. The problem has been also studied from the perspective of Semantic Web technologies. Each approach offers some distinct advantages. Natural language processing techniques can capture complex relations between entities but with a computational cost that is often difficult to scale up to the volume of Web data. Semantic Web methods rely on ontologies that can also describe complex relations, but are limited to a predefined number of them. For the purpose of entity ranking on the Web, an IR approach is being presented in this paper. IR methods have been proven by Web search engines to be effective in dealing with large volumes of data and the heterogeneity of information found on the Web. They also allow free text querying and can provide fresh information that is found in Web documents.

In [1], we presented an application for entity ranking called ListCreator. Six ranking methods were formulated and their performance was evaluated. In this extended version, we perform a systematic and exhaustive analysis of possible ranking methods based on the same hypotheses and statistical measures. Therefore, we evaluate eleven ranking formulae that take into account all measure combinations, excluding ranking-wise equivalent ones. Furthermore, we address a serious limitation of this approach and ranking methods for retrieval in general. While using an effective ranking method, we expect the relevant results to be ranked higher than the non-relevant ones, but there is no further indication for how many are correct. This poses a serious problem for entity ranking when increasing the number of source documents. The ranking may get better, but the number of incorrect results on the lower part of the list also increases. A way to mitigate this problem is estimating a threshold in the results list, from which point on, the relevancy is rapidly degrading. We investigate solutions to this problem in Section V.

ListCreator can answer user queries for entities of the three major categories: persons, locations, and organizations. The application uses a search engine to obtain a small collection of Web documents that are related to the submitted query. The entities found in the documents are extracted using a named entity recogniser. The ranking is achieved by statistical information retrieval methods, taking advantage of the common information among the source documents. The results are returned to the user as a ranked list of all the relevant entities that the application managed to extract.

The above ranking method is based on two assumptions. First, that a Web search engine will be able to retrieve documents that contain the relevant entities. The connection between the query and the entities will take place using document retrieval methods. Second, given that the Web is a collection of documents from independent authors, the desired information or some part of it will be found on several different documents. In order to obtain a ranking of entities according to relevance to the query, we find how important each entity is for the retrieved collection. The work on this paper mainly addresses the problem of measuring this importance with a statistical model.

The system relies on the technologies of Web search and named entity recognition for acquiring data in order to perform the ranking. Therefore, it can be used as a front-end to a commercial Web search engine utilizing its state-of-the-art search functionality. Our motivation is to build an entity ranking system that can use effectively the information in Web documents, and can produce results without relying on external sources. An online demo of the application can be found in [2].

The contribution of this paper is threefold. First, we build an online prototype as proof-of-concept for entity ranking as a search engine front-end, using IR methods. Such methods are simple and fast, and therefore suited for an online application, also scaling well to large amounts of data. Second, we formulate and experimentally evaluate several ranking methods that can be used in the particular system. Third, we evaluate the performance of four algorithms for threshold estimation.

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 describe the different methods for ranking entities and perform experiments to compare their effectiveness. The experiments for threshold estimation are presented in Section V, followed by a discussion in Section VI. Conclusions are drawn in Section VII, together with directions for further research and improvements.

II. Related Work

Entity ranking has a lot in common with automatic question answering, where the answer to a query is often a name or a list of names. Research for question answering systems took place during the TREC (Text REtrieval Conference) QA track. A method used for extracting answers from raw text is checking document snippets that are relevant to the query, and counting the frequency of each possible answer [3][4]. As the frequency of a candidate answer gets higher, so does the probability of it being the correct one. This approach is similar to ListCreator's, with the difference being that it is not a model build to produce a ranking, but only focuses on the top result, in order to provide a single answer.

Another related task to entity ranking is expert finding. For this task, a system has to automatically find an expert that meets the criteria determined in a user's query, so it is an entity retrieval problem limited to the person category. In [5], two models for expert finding were formalized. In Standard Model 1, candidate experts are described with representative documents and document retrieval methods are used to obtain the relevance of an expert according to a query. Standard Model 2 uses documents relevant to the query as latent variables for calculating the desired relevance of experts to queries. The documents are retrieved using standard document retrieval methods and each expert is assigned a total score that corresponds to the sum of the scores of all the documents that his name appears in. In [6], a combination of the two methods is proposed, creating profiles that incorporate parts of many different documents according to probability distributions.

IR methods for the purpose of entity ranking were demonstrated and evaluated during the INEX (INitiative for the Evaluation of XML retrieval) and TREC entity ranking tracks. INEX evaluated the performance of many systems from 2007 to 2009, for entity ranking in Wikipedia in two tasks [7][8]. For the entity ranking task, the requirement was retrieving entities that satisfy a topic described in natural language, while for the list completion task the objective was creating a list of entities given some examples and a description. For these tasks, an entity is anything that has a Wikipedia article dedicated to it. Participating teams used Wikipedia's semi structured format, specifically the categories and the links between articles for determining entity relations, and the infoboxes for retrieving information in a machine readable way.

TREC evaluated systems for entity ranking in the Web from 2009 to 2011 [9][10]. A name was considered to correspond to an entity, if it had its own Webpage. TREC runs a related entity search task, where the goal was to retrieve relevant entities that satisfy conditions related to another entity, and a list completion task similar to INEX. Typical approaches used the given entity to determine relation with candidate entities through co-occurrence frequency and link analysis. In [11], the structure of HTML is used to find entities in lists and tables assuming that entities found in the same format will also belong to same category, along with specific templates and filtering rules. In [12], a profile document is constructed from different parts of the corpus that mention a candidate entity, and then document retrieval is used for ranking. In [13], a document language model for estimating the probability of generating an entity from a query, a supervised and an unsupervised learning to rank approaches using SVMs are tested. In [14], Wikipedia was used as an information source for Web entity ranking, providing descriptive documents, category and link information.

A typical feature used for entity ranking from document sources is proximity measures of candidate entities and query terms. Proximity measures estimate the relevance of an entity to a query by taking into account the quantity and distance of query terms to an entity in a predefined window, aggregated over many documents. In [15], a model for ranking with proximity measures is built using non-uniform kernel functions, while in [16] and [17] proximity measures are enhanced by patterns that consider the order of the keywords. In [18], the proximity, profile and voting methods were integrated in a single probabilistic model using a Markov Random Field.

A different approach to entity ranking is using information extraction techniques to construct structured data from text by extracting facts about entities [19][20]. This requires natural language processing, for example part of speech tagging, and is typically achieved by 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 to obtain RDF data like in [21]. The database method adopts a data retrieval approach for entity ranking, where the system accepts structured queries in a query language like SPARQL or more sophisticated extensions like [22], instead of imprecise free text queries. Recent research addressing the problem of transforming keyword queries to structured ones can be found in [23] and [24].

Entity retrieval by keyword queries from datasets of ontologies was the subject of the Semantic Search Challenge (2010, 2011) [25][26] and the JIWES 2012 (Joint International Workshop on Entity-oriented and Semantic search) [27], where a related entity finding and a list search task similar to TREC took place. The objective was to rank entities belonging to the Linked Open Data according to a free text query. The participants used IR methods specifically modified for retrieving RDF data. A model for efficient usage of the structured information presented in a knowledge base is explored in [28]. Approaches that combine IR models for keyword search and the structured information presented in knowledge bases to return a ranking according to relevance can be found in [29] and [30].

The database and ontology approaches have currently certain limitations compared to the IR methods that deal directly with unstructured data (text). The relations and attributes defined in an ontology are limited in comparison to the relations found in documents and can be interesting to a user. Furthermore, a system relying in a database can only accept structured queries that impose restrictions to the user. Finally, a lot of effort is required to keep a database up to date with recent information, something that an approach dealing directly with Web documents can easily achieve.

The idea of using statistical evidence from multiple sources to certify the correctness of results has been used in the similar tasks of question answering and expert finding. However, previous work does not investigate suitable ranking methods to better utilize evidence, instead they make an arbitrary choice of ranking method without considering alternatives, e.g., term frequency [3] or document score aggregation [5]. Research on the document retrieval task has shown that different ranking methods can play an important role in the quality of retrieval. In this paper, we investigate an optimal ranking method for the entity retrieval task by using a systematic approach. In order to formulate ranking methods, we make no assumptions about the underlying distribution of relevancy for entities. Instead, we begin with two generic hypotheses about use of language from Web document authors and construct several ranking algorithms for each one, using different combinations of statistical measures. We proceed to measure effectiveness based on empirical evaluation.

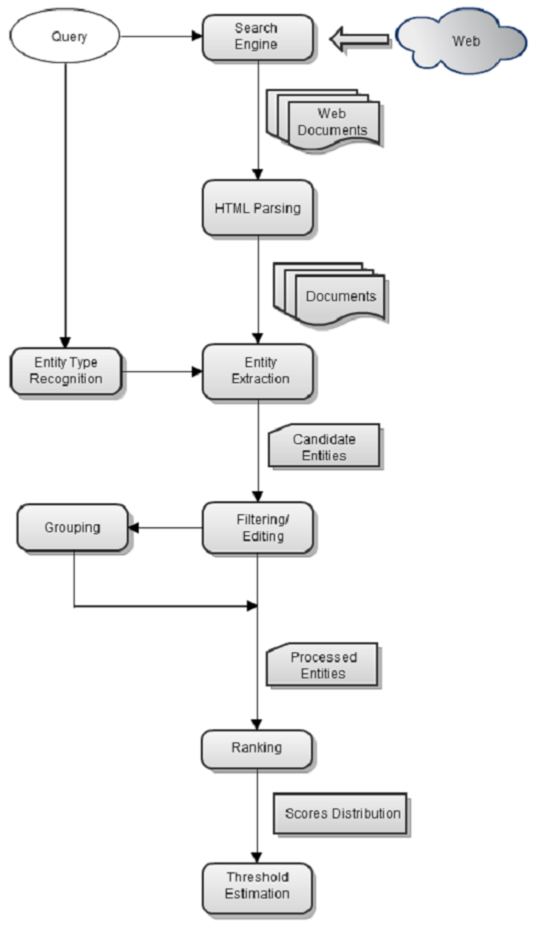


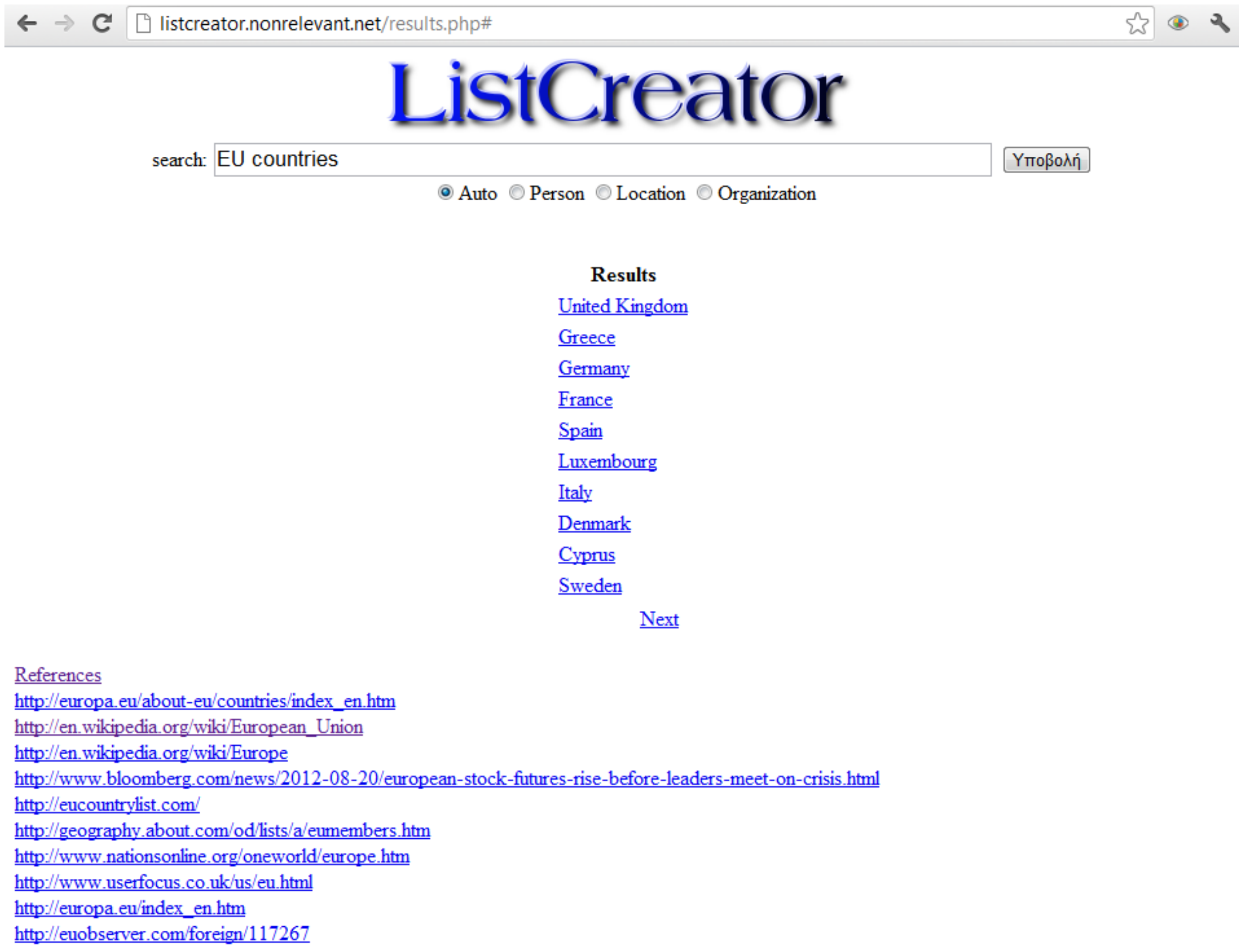
Figure 1. The system's components and dataflow.

III. System Description

The data flow that takes place in the system is depicted in Figure 1. The components for formatting, filtering, grouping and ranking of entities are all coded in JAVA [31]. The user Web interface is coded in HTML [32], JavaScript [33], and PHP [34].

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 he is searching



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.

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 [35]. 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 where 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 [36]. 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 [37]. Stanford NER is a system for entity extraction from text coded in JAVA and distributed with GNU general public license [38] 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 task. 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 [39] is based on the CRF (Conditional Random Field) probabilistic model [40] 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 [41]. 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 better ranking in the next stage, the system must recognize which names correspond to the same entity, a task called *coreference 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 falsely 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. The list is not used when the user is searching for country names. The effect of this method is splitting the location category into two, countries and other locations, providing a better filtering. The described exception lists are used solely for the purpose of improving the entity recognition task. Since Stanford NER is not trained on Web documents, its accuracy is lower when dealing with them. Some common mistakes are handled by the first list we accumulated. An alternative method would be to retrain the classifier on Web documents. The location entity category is too broad, and an approach for obtaining finer grained entities is splitting it into a geopolitical entities category and other locations. By using a list of known countries we take a step towards that direction.

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 coreference resolution utilizing machine learning, but such an approach would increase computational cost.

E. Entity Ranking

The ranking algorithm makes usage of statistical methods of IR. 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:

$$score = df \sum_{i=1}^{df} (N + 1 - r_i)$$

where i 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 Borda Count preferential voting method, multiplied by the document frequency of the entity. According to the formula, an entity that appears only in the first document will get 10 points, if an entity appears on the first and second document, it will get 10 plus 9 points multiplied by 2, etc. Entities with higher score are considered more relevant to the query. This ranking formula was chosen after the 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

TABLE 1. SUMMARY OF RANKING EQUATIONS.

			Frequency Weighting			
			None	Verbosity	in-between	scope
			1	df	$\log(1+f_i)$	f_i
Document Frequency Weighting	None	1	no ranking	(1)	(3)	(2)
	Linear	df	(1)	equivalent to (1)	(4)	(6)
	Logarithmic	log(1+df)	equivalent to (1)	equivalent to (1)	(5)	(7)
	rank-based	Borda Count	(8)	(9)	(10)	(11)

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 each document (f_i). The higher the frequency of an entity, the more confidence we have in its importance for a particular document.
- 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*). By taking into account this quantity the documents are no longer treated as equally relevant.

There are two opposite hypotheses regarding the frequency of a term and the importance that it has for a document [42]. According to the *verbosity hypothesis*, multiple occurrences of a term are not really important, because the document's author 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 he has more information to share on this subject.

Using the above statistical measures and hypotheses we formulate 11 ranking equations. The measures are used by itself and in multiplicative combination. We try linear and sublinear scoring, where for the latter case we use the logarithmic function as a damping factor. The logarithmic scoring for entity frequency gives an in-between approach for the two hypotheses. In all the following formulae, *i* is the document, *N* is the total number of documents and equals 10, f_i is the number of occurrences of an entity in document *i*, and *df* is document frequency.

A. Frequency-only scoring

Under the verbosity hypothesis an entity will not get extra credit for appearing more than once in a document so:

$$score = \sum_{i=1}^{df} 1 = df \tag{1}$$

The scope hypothesis suggests that each entity appearance contributes linearly to relevance:

$$score = \sum_{i=1}^{df} f_i = f \tag{2}$$

The logarithmic approach corresponds to an in-between approach and gives:

$$score = \sum_{i=1}^{df} \log(1+f_i) \tag{3}$$

This means that we get diminishing returns on entity occurrences, so only the first few of them can contribute significantly to the score.

B. Document Frequency-only scoring

For the document frequency scoring, both the linear (*df*) and logarithmic approach (logarithm of *df*) result in ranking that is equivalent to that of (1).

C. Combination of scoring measures

Combining the two measures multiplicatively and excluding combinations that give equivalent ranking to the above scoring equations we get:
In-between frequency weighting and linear *df*:

$$score = \sum_{i=1}^{df} \log(1+f_i) df \tag{4}$$

In-between frequency weighting and logarithmic *df*:

$$score = \sum_{i=1}^{df} \log(1+f_i) \log(1+df) \tag{5}$$

Scope hypothesis frequency weighting and linear *df*:

$$score = f \times df \tag{6}$$

Scope hypothesis frequency weighting and logarithmic *df*:

TABLE 2. EVALUATION RESULTS FOR THE 11 RANKING FORMULAE AVERAGED OVER THE 30 QUERIES.

Ranking Equations	P@10	R-Precision
(1)	0.4733	0.4209
(2)	0.3900	0.3675
(3)	0.4400	0.4200
(4)	0.4700	0.4314
(5)	0.4500	0.4267
(6)	0.4367	0.4178
(7)	0.4333	0.4061
(8)	0.4900	0.4216
(9)	**0.4933**	0.4200
(10)	0.4766	**0.4463**
(11)	0.4100	0.4025

$$score = f \times \log(1 + df) \tag{7}$$

Multiplicative combinations between the verbosity hypothesis weighting for entity frequency and document frequency result in the same ranking as the one provided by (1).

D. *Document Rank scoring*

In previous equations, the documents were seen as equivalent. To weight each document according to its rank we use the Borda Count:

$$score = \sum_{i=1}^{df} (N + 1 - r_i) \tag{8}$$

Equation (8) is equivalent to the ranking method proposed in [5], with unknown scores for the document retrieval part.

Combining the document rank with frequency weighting according to the three hypotheses we get:

$$score = df \sum_{i=1}^{df} (N + 1 - r_i) \tag{9}$$

$$score = \sum_{i=1}^{df} \log(1 + f_i)(N + 1 - r_i) \tag{10}$$

$$score = \sum_{i=1}^{df} f_i (N + 1 - r_i) \tag{11}$$

The summary of all the ranking equations according to different weightings is on Table 1.

E. *Evaluation*

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 correct answers, it offers an easy interpretation of results and does not require knowledge of the total number of correct answers to be computed. Furthermore, the P@10 measure is suitable for Web retrieval evaluation, 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 can have great variance. R-Precision shows the number of relevant answers within the top-R results, where as R we use the total number of relevant answers in the set. R-precision overcomes the problem of variance in the number of correct answers [43].

TABLE 3. THE 30 EVALUATION QUERIES WITH THE NUMBER OF CORRECT RESULTS RETRIEVED (R).

Evaluation Queries	R
Pacific navigators Australia explorers	23
List of countries in World War Two	105
Nordic authors known for children's literature	6
Makers of lawn tennis rackets	3
National capitals situated on islands	46
Poets winners of Nobel prize in literature	16
Formula 1 drivers that won the Monaco Grand Prix	32
Formula One World Constructors' Champions	11
Italian Nobel prize winners	9
Musicians who appeared in the Blues Brothers movies	29
Swiss cantons where they speak German	15
US Presidents since 1960	11
Countries which have won the FIFA world cup	8
Toy train manufacturers that are still in business	9
German female politicians	108
Actresses in Bond movies	67
Star Trek Captains characters	10
EU countries	27
Record-breaking sprinters in male 100-meter sprints	14
Professional baseball team in Japan	19
Japanese players in Major League Baseball	46
Airports in Germany	52
Universities in Catalunya	8
German cities that have been part of the hanseatic league	18
Chess world champions	20
Recording companies that now sell the Kingston Trio songs	5
Schools the Supreme Court justices received their undergraduate degrees	37
Axis powers of World War Two	6
State capitals of the United States of America	36
National Parks East Coast Canada US	10

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, in order to determine the most effective ranking method. Chosen queries deal with entity types of the three categories that are supported in the system. 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 2. The query set along with the total number of relevant entities that were retrieved by the system (R) for each one is on Table 3.

The 11 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 the verbosity hypothesis works better for entity ranking. However, (4) and (10) that represent the middle ground, achieve a higher R-Precision. 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 (11).

V. THRESHOLD ESTIMATION

A large set of possible queries for entity ranking problem have answers that take a binary value of relevance, i.e., they can be described as either relevant or non-relevant. All the queries used in the experiment are of this type (factoids), in contrast with queries that ask for opinion and, therefore, their answers can rarely take binary values of relevance. We investigated ways to estimate the total number of correct results (R) for a query. The threshold can then be R. This is the breakeven-point of precision and recall, meaning that at this point the precision of the system is equal to its recall. As a result, the harmonic mean of precision and recall (the F_1 measure) is maximized. This threshold choice strikes a balance between precision and recall.

The problem of threshold estimation for document retrieval was addressed in [44], where the score distributions were used to cluster the results. The scores of relevant and non-relevant results were treated as belonging to different probability distributions and the expectation maximization algorithm was used to determine their corresponding distribution. For the problem of entity retrieval, the distributions of scores are generally unimodal, so we applied nonparametric approaches to estimate the threshold.

A great variety of algorithms are used for threshold estimation in image segmentation, where the problem is to find a threshold of the grey-level value of pixels, in order to separate the foreground and background of an image. A lot of these algorithms are nonparametric and take as input only the histogram of the values making them suitable for threshold estimation in problems unrelated to image segmentation. We applied four algorithms: Otsu's algorithm [45], Kapur et al. maximum entropy [46], Rosin's algorithm [47], and the t-point algorithm [48], and tested their accuracy to the problem of threshold estimation for each ranking method. We give a short description of each algorithm.

TABLE 4 . RELATIVE ERROR OF THE THRSHOLD ESTIMATION FOR EACH THRESHOLD ALGORITHM APPLIED TO EACH RANKING EQUATION.

Equatio n	**Otsu**	**Entropy**	**Rosin**	**T-point**
(1)	0.7876	0.7256	0.8062	0.6451
(2)	0.9461	0.7625	0.7588	0.7742
(3)	1.2953	0.7783	1.0039	0.6860
(4)	0.7301	0.7840	0.6092	0.7045
(5)	0.8791	0.7091	0.9455	0.6666
(6)	0.8334	0.7674	0.719	0.7791
(7)	0.8341	**0.5984**	0.7594	0.7255
(8)	4.7738	0.9976	2.3284	1.7352
(9)	1.2242	0.7203	1.3532	1.088
(10)	2.8295	0.6114	1.2869	0.8739
(11)	1.0267	0.7341	0.6630	0.7248

A. *Otsu's algorithm*

For each value of the threshold in the histogram, the algorithm computes the variance of the two resulting classes and their sum. The optimum value for the threshold is the value that gives the minimum sum of variance for the two classes. This optimization criterion is equivalent to maximizing the between class variance.

$$T_{opt} = \arg\max \left\{ \frac{P(T)[1-P(T)][\mu_r T - \mu_{nr} T]^2}{P(T)\sigma_r^2(T) + [1-P(T)]\sigma_{nr}^2(T)} \right\} \tag{12}$$

where *P(T)* is the probability of a relevant result for a threshold value *T*, μ is the mean and σ is the variance of each class, relevant (*r*) and non-relevant (*nr*).

B. *Kapur's entropy based algorithm*

Every possible value for the threshold is tested and the algorithm calculates the sum of the entropy for the two resulting classes. The algorithm chooses as optimum threshold the value that maximizes the sum of the entropy of the two classes.

$$T_{opt} = \arg\max\{H_r(T) + H_{nr}(T)\} \tag{13}$$

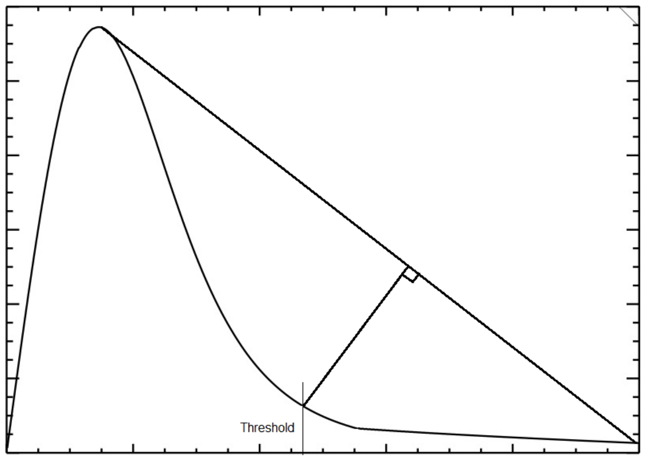


Figure 3. Rosin's algorithm. For threshold estimation in the entity ranking context, values of the x-axis represent the ranking scores and values of y-axis represent the number of entities that were assigned that score.

$$H_r(T) = -\sum_{i=0}^{T} \frac{P(i)}{P(T)} \log \frac{P(i)}{P(T)}$$

$$H_{nr}(T) = -\sum_{i=T+1}^{S} \frac{P(i)}{1-P(T)} \log \frac{P(i)}{1-P(T)}$$

where H_r and H_{nr} are the entropies of the relevant and non-relevant class, S is the number of values of the histogram of the scores.

C. *Rosin's algorithm*

The algorithm considers the line that crosses the maximum value of the histogram and the last value. The threshold is determined as the point in the histogram between the aforementioned points that has the greatest Euclidian distance from the line (Figure 3).

D. *T-point algorithm*

For every value of the histogram between the maximum and the last value, two lines are fitted to the data using linear regression. The first is between the maximum value and the threshold, and the second between the threshold and the last value. The goodness of fit of these two lines against the points of the histogram is calculated by checking the sum of residuals. The algorithm determines the optimum threshold as the point that the two best fitted lines intersect (Figure 4).

Otsu's and Kapur's are largely cited algorithms for determining threshold values in a multimodal histogram, while Rosin's and the T-point algorithm are designed for unimodal histograms. For each algorithm and each ranking method we calculated the relative error of the estimated threshold compared to the real value and averaged over the 30 queries. The results of the experiment are in Table 4. The histogram of the scores distribution was created by grouping the values into 10 different bins. This resulted in better accuracy than using all the distinct score values, while further increase of the number of bins did not affect the results significantly.

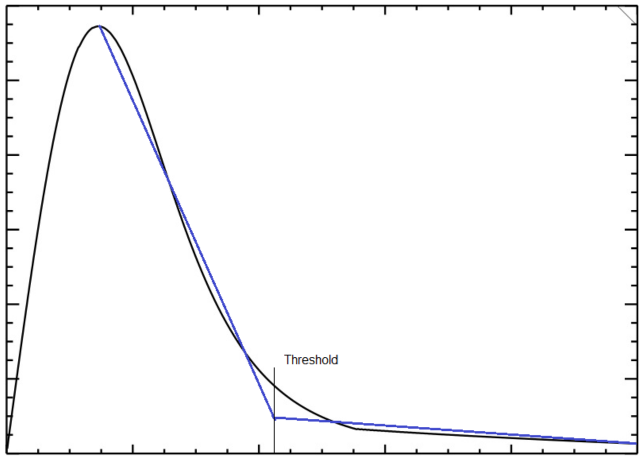


Figure 4. The T-point algorithm. Axis values are the same as in Figure 3.

The results suggest that each combination of ranking equation and thresholding algorithm performs differently. Although the histograms of the score distributions were mostly unimodal, the algorithms that specialize in unimodal thresholding did not perform better. The single best result is obtained with Kapur's maximum entropy algorithm when applied to the score distribution of (7).

VI. Discussion

In order to decide for a better ranking method, a user model has to be taken into account. Assuming the user wants to find all relevant results, methods with higher R-Precision will work better. In case a user is interested in only a few characteristic results, then a method with higher P@10 performance will be more useful. The reason that (9) is used in the prototype is that we expect most Web users to belong in the second category.

The experiment also provided some insight in the overall system's functionality. First, we noticed the dependency of performance on the quality of retrieved documents. For queries that even one strictly relevant document was retrieved, like a Wikipedia "list of" page, the ranking was nearly optimal. In cases where partially relevant documents were retrieved, for example lists of entities according to one attribute requested by the query, the system managed to produce a combination but with reduced accuracy. The most problematic queries proved to be ones with a complex relation between attributes that cannot be well defined by simple keywords, such as "toy train companies that are still in business". A query like this would require some extra pre-processing, perhaps combined with a model for reasoning. Another problem comes with queries that have a small amount of correct answers (e.g., Axis Powers of World War Two). The threshold estimation can give valuable information in such a case so that a user can consider only a few top results as relevant. Even though only 10 documents were used, a large number of relevant entities were retrieved for each query.

The discussed approach has the benefit of scaling well to large amounts of data. In the current implementation, we are forced to perform the named entity recognition task in real time because we do not have access to the search engine's index. Given an integrated approach, the entity extraction part can be completed offline during the preprocessing stage of indexing. The only part that always has to be computed in real time is ranking, which is accomplished by a simple formula, and is therefore not affected by the amount of data. Named entity recognition has been effectively applied in large document collections [49]. Given that all the necessary data preparation has been completed offline, we can expect that increasing the amount of data can only have a positive effect. The entity categories depend exclusively on the capabilities of the NER system. Systems for more and finer-grained categories have been described in the NER literature, e.g., in [50]. The extension of entity types can provide better filtering, but could prove problematic for the automatic type detection from the query. A more sophisticated method rather than relying on keyword detection may be needed.

VII. 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 IR methods. The application uses various components for recognizing the query topic, retrieving documents, extracting entities and performing coreference resolution before the ranking takes place. We formulated and evaluated several combinations of statistical quantities for ranking entities and algorithms for estimating the number of relevant results.

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 within-document frequency of entities did not work very well, supporting the verbosity hypothesis. Furthermore, the experiments showed that using the large data volume of the Web along with a state-of- the-art Web search engine for retrieving them, the system has little limitations in query handling. The threshold estimation experiment suggests that Kapur's maximum entropy algorithm applied to the score distribution of a multiplicative combination of entity frequency and document frequency gives the best results for estimating the number of relevant answers returned by the system.

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 entity extraction stage which uses machine learning methods. By integrating the application with a search engine the required processing for this stage could be done in advance along with the indexing stage. With this modification, the speed of the ranking method will be fully utilized.

Acknowledgment

This work was done when the first author was with the Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, Greece.

References

[1] A. Komninos and A. Arampatzis, "ListCreator: Entity Ranking on the Web," in *Proceedings of The Second International Conference on Advances in Information Mining and Management*, 2012, pp. 141-146.

[2] ListCreator. [Online]. Available: http://listcreator.nonrelevant.net [retrieved: May 2013].

[3] B. Sabine, "Using Grammatical Relations, Answer Frequencies and the World Wide Web for TREC Question Answering," in *Proceedings of the Tenth Text REtrieval Conference*, 2001.

[4] J. Lin, "The Web as a Resource for Question Answering: Perspectives and Challenges," in *Proceedings of the third International Conference on Language Resources and Evaluation*, 2002.

[5] B. Krisztian, L. Azzopardi, and M. De Rijke, "Formal models for expert finding in enterprise corpora," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieva*, 2006, pp. 43-50.

[6] P. Desislava and W. B. Croft, "Hierarchical language models for expert finding in enterprise corpora," in *International Journal on Artificial Intelligence Tools,* 2008, pp. 5-18.

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

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

[9] K. Balog, A. P. de Vries, P. Serdyukov, P. Thomas, and T. Westerveld, "Overview of the TREC 2009 Entity Track," in *Proceedings of the Eighteenth Text RErieval Conference*, 2009.

[10] K. Balog, A. P. de Vries, and P. Serdyukov, "Overview of the Trec 2010 Entity Track," in *Proceedings of the Nineteenth Text RErieval Conference*, 2010.

[11] Q. Yang, P. Jiang, C. Zhang, and Z. Niu, "Reconstruct Logical Hierarchical Sitemap for Related Entity Finding," in *Proceedings of the Nineteenth Text REtrieval Conference*, 2011.

[12] J. Guo, H. Xu, and Y. Liu, "A Novel Framework for Related Entities Finding: ICTNET at TREC 2009 Entity Track," in *Proceedings of the Eighteenth Text REtrieval Conference*, 2009.

[13] Y. Wu and H. Kashioka, "NiCT at TREC 2009: Employing Three Models for Entity Ranking Track" in *Proceedings of the Eighteenth Text REtrieval Conference*, 2009.

[14] J. Kamps, R. Kaptein, and M. Koolen, "Using Anchor Text, Spam Filtering and Wikipedia for Web Search and Entity Ranking," in *Proceedings of the Nineteenth Text REtrieval Conference*, 2011.

[15] D. Petkova and W. B. Croft, "Proximity-based document representation for named entity retrieval," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, 2007, pp. 731-740.

[16] Cheng, Tao, Xifeng Yan, and Kevin Chen-Chuan Chang, "EntityRank: searching entities directly and holistically," in *Proceedings of the 33rd international conference on Very large data bases*, VLDB Endowment, 2007, pp. 387-398.

[17] X. Li, C. Li, and C. Yu, "Entity-relationship queries over wikipedia," in *ACM transactions on Intelligent Systems and Technology*, volume 3, issue 4, ACM, 2012.

[18] R. Hadas, D. Carmel, and O. Kurland, "A Ranking Framework for Entity Oriented Search using Markov Random Fields," in *Proceedings of the 1st Joint International Workshop on Entit-Oriented and Semantic Search,* ACM, 2012.

[19] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam, "Open Information Extraction: the Second Generation," in *Proceedings of the Twenty-second International Joint Conference in Artificial Intelligence*, 2011, pp. 3-10.

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

[21] J. Hoffart, F.M. Suchanek, K. Berberich, E. Lewis-Kelham, G. DeMelo, and G. Weikum, "Yago2: exploring and querying world knowledge in time, space, context, and many languages," in *Proceedings of the 20th international conference companion on world wide Web, ACM,* 2011, pp. 229-232.

[22] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum, "NAGA: Searching and Ranking Knowledge," in *Proceedings of the Twenty- fourth International Conference on Data Engineering*, 2008, pp. 953-962.

[23] 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.

[24] J. Pound, I.F. Ilyas, and G. Weddell, "Expressive and flexible access to Web-extracted data: a keyword-based structured query language," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data,* 2010, pp. 423-434.

[25] M. Grobelnik, P. Mika, D. T. Tran, and H. Wang. *Proceedings of the 3rd International Semantic Search Workshop,* ACM, 2010.

[26] M. Grobelnik, P. Mika, D. T. Tran, H. Wang, "SemSearch'11: the 4th semantic search workshop," in *Proceedings of the 20th international conference companion on World Wide Web,* ACM, 2011, pp. 315-316.

[27] K. Balog, D. Carmel, A.P. de Vries, D. M. Herzig, P. Mika, H. Roitman, and T. T. Duc, "The first joint international workshop on entity-oriented and semantic search (JIWES)," *ACM SIGIR Forum*, Volume 46. No. 2. ACM, 2012, pp. 87-94.

[28] K. Balog, R. Neumayer, and K. Norvag, "On the modeling of entities for ad-hoc entity search in the Web of data," in *Advances in Information Retrieval*, 2012, pp. 133-145.

[29] R. Delbru, S. Campinas, and G. Tummarello, "Searching Web data: An entity retrieval and high-performance indexing model," in *Web Semantics: Science, Services and Agents on the World Wide Web*, Volume 10, 2012, pp. 33-58.

[30] H. Wang, Q. Liu, T. Penin, L. Fu, L. Zhang, T. Tran, Y. Yu, and Y. Pan, "Semplore: A scalable IR approach to search the Web of Data," in *Web Semantics: Science, Services and Agents on the World Wide Web*, Volume 7, Issue 3, 2009, pp. 177-188.

[31] Java. [Online]. Available: http://www.java.com/en/ [retrieved: May 2013].

[32] HTML 4.01 Specification. [Online]. Available: http://www.w3.org/TR/1999/REC-html401-19991224/ [retrieved: May 2013].

[33] JavaScript. [Online]. Available: https://developer.mozilla.org/en-US/docs/JavaScript [retrieved: May 2013].

[34] PHP. [Online]. Available: http://www.php.net/ [retrieved: May 2013].

[35] Princeton University, 2010, WordNet. [Online]. Available: http://wordnet.princeton.edu [retrieved: May 2013].

[36] Yahoo BOSS API. [Online]. Available: http://developer.yahoo.com/search/boss [retrieved: May 2013].

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

[38] GNU General Public License. [Online]. Available: http://www.gnu.org/licenses/gpl.html [retrieved: May 2013].

[39] Named Entity Recognition and Information Extraction [Online] http://nlp.stanford.edu/ner/index.shtml [retrieved: May 2013].

[40] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001, pp. 282-289.

[41] Jsoup: Java HTML Parser. [Online]. Available: http://jsoup.org [retrieved: May 2013].

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

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

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

[45] N. Otsu, ''A threshold selection method from gray level histograms,'' in *Automatica*, Volume 11, no. 285-296, 1975, pp. 23-27.

[46] J. N. Kapur, P. K. Sahoo, and A. K. C. Wong, ''A new method for gray-level picture thresholding using the entropy of the histogram,'' in *Computer Vision, Grapic, and Image Processing*, Volume 29, issue 3, 1985 ,pp. 273-285.

[47] P.L. Rosin, "Unimodal thresholding," in *Pattern Recognition*, Volume 34/11, 2001, pp. 2083-2096.

[48] N. Coudray, J.L. Buessler, J.P. Urban, "Robust threshold estimation for images with unimodal histograms," in *Pattern Recognition Letters*, Volume 31, Issue 9, 2010, pp. 1010-1019.

[49] C. Whitelaw, A. Kehlenbeck, N. Petrovic, and L. Ungar, "Web-Scale Named Entity Recognition," in *Proceedings of the 17th ACM conference on Information and knowledge management*, 2008, pp. 123-132.

[50] S. Sekine and C. Nobata, "Definition, Dictionaries and Tagger for Extended Named EntityHierarchy," in *Proceeding of Itnernational Conference on Language Resources and Evaluation*, 2004.