Learning to Rank for Semantic Search

Lorand Dali
Jožef Stefan Institute
Jamova 39, Ljubljana
Slovenia
+386 1 477 3933
Lorand.Dali@ijs.si

Blaž Fortuna
Jožef Stefan Institute
Jamova 39, Ljubljana
Slovenia
+386 1 477 3934
Blaz.Fortuna@ijs.si

ABSTRACT

PageRank, one of the most important algorithms in information retrieval, was developed to give an estimate of how many users are likely to be visiting a given page on the web at a given moment. This way PageRank gives an approximation of the popularity of a page. Because semantic web resources can be represented in a RDF graph it is easy to adapt the original PageRank algorithm to rank these resources; and indeed much previous work has been done in this direction. However there are important differences between an RDF graph and a graph composed by web pages and hyperlinks. This paper explores to what extent PageRank and other domain and query-independent features can approximate the popularity of a resource in the semantic web. Moreover we study how these features can be combined to derive more robust ranking models. We did our study on DBpedia and Yago, two knowledge bases where the resources have one-to-one mappings to Wikipedia articles. For popularity of a resource we used the Wikipedia access logs as a golden standard.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval *–retrieval models.*

General Terms

Algorithms, Experimentation,

Keywords

semantic search, learning to rank, ranking models, page rank

1. INTRODUCTION

With the development of the semantic web and the increase in number of resources available in it, the very important application of object-level information retrieval has become practical. As opposed to web information retrieval, where search engines give pages in response to the user's queries, results in object-level information retrieval are finer grained. Here entities such as persons, countries, buildings, phone numbers, movies, athletes etc. are returned. Query languages such as SPARQL make the search for specific objects in the semantic web possible. However, SPARQL has two important disadvantages: it is schema specific so the user has to be familiar with the structure of the ontology he is querying, and the results are not ranked. This paper is related to the second problem, the lack of ranking in the results returned for a SPARQL query.

In the early days of web search it became clear that just giving the users all the documents which match the search query does not work because usually a very high number of documents are matching, and the user cannot process them all. Even more, studies have shown that users typically scan the search results from top to bottom and usually focus only on the top three or four search results [1]. This is why big efforts were made to develop ranking models which place the most relevant documents as high as possible in the list of search results. One of the most influential ranking models which was developed for web information retrieval is PageRank [2]. The aim of PageRank is to give a global, query independent ranking of a page which would accurately approximate the popularity of that page. The score computed by PageRank for a given page can be looked at intuitively as the number of random web surfers which are visiting that page at a given moment. To achieve this, PageRank leverages the hyperlink structure of the web and looks at the internet as a big graph where pages are nodes and hyperlinks are edges.

RDF datasets in the semantic web can also be represented as graphs where resources such as *Tennessee*, *Arthur Golden* and *Memoires of a Geisha* are nodes, and relations such as *writtenBy* and *bornIn* are edges connecting them. Moreover, a recent study of the object link structure in the semantic web [3] has confirmed that the RDF graph structure of the linked open data is very similar to the structure of the graph of web pages on the internet. This suggests that PageRank is applicable as a ranking model in object-level information retrieval as well.

A lot of previous work takes advantage of the RDF graph representation of linked open data, and Entity-Relation graphs in databases, to adapt PageRank for ranking the results of SPARQL or SQL queries. The approach of [4] uses a two layered version of PageRank where a resource gets a high rank if it has a high PageRank within its own knowledge base and if the knowledge base has a high PageRank in the linked open data cloud. Other papers try to make the ranking query specific by adapting the Personalized PageRank algorithm [5]. [6] introduces ObjectRank where the teleport vector is nonzero only for nodes matched by the query, and HubRank [7] deals with the scalability issues of computing Personalized PageRank vectors at runtime by computing them for a carefully chosen subset of hub nodes beforehand. Most of these papers point out the difficulty in adapting PageRank to RDF graphs, which as opposed to the web graph have heterogeneous nodes and edges. The solution is most often manually assigning weights to different relations, but this approach is only applicable in a restricted domain such as experiments over paper-author-conference collections [6],[8]. [8] and [9] try to learn popularity propagations as different weights for edges representing different relations.

The contribution of this paper is determining to what extent the PageRank algorithm is applicable in the RDF graph scenario. We do the experiments on two of the arguably most important and diverse datasets in the semantic web: DBpedia [10] and Yago [11]. The resources in DBpedia and Yago have one-to-one mappings to Wikipedia articles. The number of visits of an article in the Wikipedia access logs[1] from June 2010 to January 2011 is taken as golden standard for the popularity of the corresponding resource. We show that directly using PageRank without taking the heterogeneity of the RDF graph into account weakly correlates with the popularity of the resources. Moreover we look into simple, domain-independent features which have higher correlations with the popularity of the resources and try to combine them to get better results.

2. DATA DESCRIPTION

2.1 Wikipedia

Wikipedia[2] is a web-based, collaborative, multilingual encyclopedia written by volunteers around the world. We used the English version which currently contains about 3.4 million articles. Every article describes one concept or entity, and many of the articles contain infoboxes which summarize the most important facts about the entity in a table.

2.2 DBPedia

The DBpedia knowledge base is a community effort to extract structured information from Wikipedia. The facts are extracted from Wikipedia infoboxes and stored as (subject, property, object) triples. The subject is a resource such as *Lyon* or *Lionel Messi*, the property is a relation like *hasPopulation*, *bornIn* or *playsFor*, and the object can be either a resource or a string literal. Each resource may belong to one or more categories which are organized in a hierarchy. Moreover, each DBpedia resource has a corresponding Wikipedia page. The topic coverage of the facts and categories is diverse. DBpedia contains around 7.8 million resources, 1282 properties, 590 000 categories, 295 classes and 10 million relations. Linking with other ontologies from the data cloud has made DBpedia to develop into the central interlinking hub for the web of data.

2.3 Yago

Similarly to DBpedia, Yago is an ontology whose facts were extracted from Wikipedia infoboxes. Yago facts are also represented as (subject, property, object) triples and have the same characteristics like the facts of DBpedia. Resources have corresponding Wikipedia articles. The category hierarchy for Yago is derived from Wikipedia categories and overlaps with the DBpedia category hierarchy. However Yago also has categories which correspond to the WordNet [12] taxonomy. At the moment Yago has around 2 million resources, 180 000 Wikipedia categories, 66000 WordNet categories, 74 properties and about 6 million relations.

3. LEARNING TO RANK

Learning to rank [17] is a machine learning technique used to induce a ranking model from training data. Depending on the form of the training data and the learning algorithm used, three learning to rank settings can be distinguished: the pointwise, the pairwise, and the listwise setting.

[1] http://dammit.lt/wikistats

[2] http://wikipedia.org

In the pointwise setting, the training data comes in form of relevance judgments for each document, the pairwise setting gives relative preferences between pairs of documents, and in the listwise setting a ranking model is induced from ranked lists of documents.

In our case the training data is made of lists of answers ordered according to a golden standard ranking. The answers are RDF resources. As the learning to rank algorithm we use RankSVM, which is a pairwise learning to rank algorithm. To bring the lists of ranked answers into the pairwise form we can take all pairs and assert that the higher ranked answer is more relevant than the lower ranked one. So if we have a list of n answers, we can obtain $\frac{n(n-1)}{2}$ preference pairs. To avoid too large a number of training examples we did not take all possible pairs, but we went with a window of length k over the ranked list and took all pairs in the window, thus obtaining $d = \frac{(n-k+1)(k-1)k}{2}$ training pairs (in our experiments we have set $k = 3$). Hence the training data for a question with n answers consists of a list of pairs:

$$\{(x_R^i, x_N^i)\}_{1 \le i \le d}, \quad x_R^i, x_N^i \in \mathbb{R}^P$$

where:

$$x = (f_1, f_2, \cdots, f_P)$$

$f_i$are features of the answer x, and x_R is considered more relevant than x_N. The goal is to learn a weight vector $w \in \mathbb{R}^P$, of the same dimensions as the training vectors x. Then given a new vector y we can compute the score of this vector, which is equal to the inner product between the weight vector w and the vector y.

$$score = w \cdot y$$

The ranking then consists in ordering answers by their scores decreasingly.

3.1 Rank SVM

Linear SVM [13] is a popular way of learning the weight vector w. Originally SVM is formulated as a binary classification problem, where w is the separating hyperplane with maximum margin. The linear soft margin SVM for classification can be adapted to the pairwise ranking problem[14]. The objective is to make the inner product $w \cdot y$ greater than $w \cdot y$ by the margin 1 and allowing for some errors ξ. We have:

$$w \cdot (x_R^i - x_N^i) \ge 1 - \xi_i, \forall i$$

We recall that the maximum margin separating hyperplane is the one which minimizes:

$$\frac{1}{2}\|w\|^2 + C \sum \xi_i$$

This is called the primal problem, and it is the one which we shall solve as described in[15]. By substituting ξ_i we get the hinge loss

$$\frac{1}{2}\|w\|^2 + C \sum (1 - w \cdot (x_R^i - x_N^i))_+$$

Where the function $(\cdot)_+$ is defined as $(\cdot)_+ := \max(0, \cdot)$. We minimize the hinge loss by using the subgradient method, which gives an approximate solution but converges very fast.

3.2 Evaluation Measures

3.2.1 Pairwise Accuracy

The pairwise accuracy evaluation measure is the percentage of answer pairs (x_R^i, x_N^i) which are correctly ordered by the model, in other words, how many times it is true that:

$$x_R^i \cdot w > x_N^i \cdot w$$

3.2.2 *Spearman's correlation coefficient*

Spearman's rank correlation coefficient is a measure used to evaluate the extent to which one random variable is a monotonic function of another one. In our case, let $\boldsymbol{V}$ be the vector of scores of the answers to a question, and let $\boldsymbol{W}$ be the golden standard scores of the answers to the question. The raw scores in $\boldsymbol{V}$ and $\boldsymbol{W}$ are converted to rank positions $\boldsymbol{v}$ and $\boldsymbol{w}$. We compute the difference between ranks $\forall i, d_i = v_i - w_i$. Hence, Spearman's correlation coefficient is defined as:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2-1)}, \quad 1 \le i \le n, \rho \in [-1,+1]$$

where n is the number of answers to the query. ρ is +1 if $\boldsymbol{W}$ can be expressed as a monotonic increasing function of $\boldsymbol{V}$, -1 if $\boldsymbol{W}$ can be expressed as a monotonic increasing function of $\boldsymbol{V}$, and 0 if there is no correlation between the two.

3.2.3 *Discounted Cumulative Gain*

Discounted Cumulative Gain (DCG) is an evaluation measure used in information retrieval which assigns a gain to each answer based on its position. The gain is summed starting from the top result, and is logarithmically decreased as one advances to the lower ranked results. The DCG at position p for the result list of a query is defined as:

$$DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{log_2 i}$$

The problem with DCG is that queries with higher number of answers will accumulate a higher DCG score, making it impossible to compare two queries with different number of answers. This is why we use a normalized version, nDCG, which divides the DCG score with the DCG score of the ideal ranking.

$$nDCG_p = \frac{DCG_p}{IDCG_p}$$

4. RANKING FEATURES

4.1 Importance of Wikipedia Pages

Approximations of the importance of a page/resource can be used as golden standard features in learning to rank experiments. In this paper we focus mostly on page popularity as an approximation of page importance.

We measure the popularity of a Wikipedia page by the amount of attention it receives from users. The attention from the users is given by the number of visits of a page during a time period (we shall refer to this feature in tables and graphs as **PgPop**). The number of visits of each page is obtained from the Wikipedia access logs.

From another point of view, the popularity of a Wikipedia page can be seen as the amount of attention the page receives from the authors. We consider that if a page has more content, then it is likely to be more popular or important since the author has taken the time to write more. So we consider the length of a Wikipedia page as another indicator for popularity (we shall refer to this feature in tables and graphs as **PgLen**).

We also consider the amount of editing a Wikipedia page undergoes as a measure for the popularity of that page. The intuition is that a page is likely to be important because often someone took the time to find and fix a mistake on the page or to add or improve some information on the page. (we shall refer to this feature in tables and graphs as **PgEdit**)

Because the resources in DBpedia and Yago have corresponding Wikipedia pages we can make the assumption that the popularity of a resource in the knowledge base is the same as the popularity of the corresponding Wikipedia page.

4.2 PageRank

This section briefly describes the PageRank algorithm and how it applies to our case.

PageRank was introduced in the early days of web search out of a need for a global, query independent ranking of the web pages. PageRank assumes a directed graph as input and will give a score to each of the nodes as a result. PageRank is based on the random walk model, which assumes that a very large number of users walk the graph choosing at each step a random neighbor of the current node or jumping to any node in the graph. The score of a node is given by the expected number of users being at the given node at a moment in time. The scores are computed recursively from the following equation:

$$\boldsymbol{p} = d \cdot \boldsymbol{M} \cdot \boldsymbol{p} + (1-d) \cdot \boldsymbol{u}, \qquad \boldsymbol{p}, \boldsymbol{u} \in \mathbb{R}^n, \boldsymbol{M} \in \boldsymbol{\mathcal{M}}(n)$$

Where n is the number of nodes in the graph, $\boldsymbol{p}$ is the PageRank vector containing the score for each node and is initialized with 0, $\boldsymbol{M}$ is the transition matrix constructed such that $\boldsymbol{M}[i,j] = 1$ if there is an edge from node i to node j and 0 otherwise. Moreover, to eliminate nodes which do not link to any other node we consider a sink node k such that $\boldsymbol{M}[i,k] = 1$, $\forall\, i$ and $\boldsymbol{M}[k,i] = 0$, $\forall\, i$. Finally the columns of M are normalized to sum up to 1. $\boldsymbol{u}$ is the jump vector and its entries are $\boldsymbol{u}[i] = \frac{1}{n}, \forall i$. d is the damping parameter, and represents the probability of walking to a neighboring node versus jumping. In our experiments we have set the value of d to 0.85 and the number of iterations to 20.

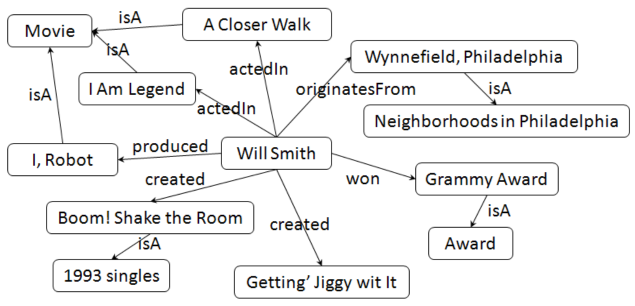

Figure 1 Graph representation of the knowledge base

In case of the web, the graph is made of the web pages as nodes and the hyperlinks as edges. In our case the nodes are DBpedia or Yago resources or categories, and the edges are properties. For illustration, Figure 1 shows a subgraph from the Yago knowledge base.
In tables and graphs we shall refer to PageRank short as **PR.**

4.3 Hubs and Authorities

Hubs and authorities, also known as the Hyperlink-Induced Topic Search (HITS) algorithm, is an iterative algorithm which takes as input a directed graph and assigns two scores to each of its nodes. The two scores, the hub score and the authority score, are defined recursively in terms of each other such that a node gets a high hub

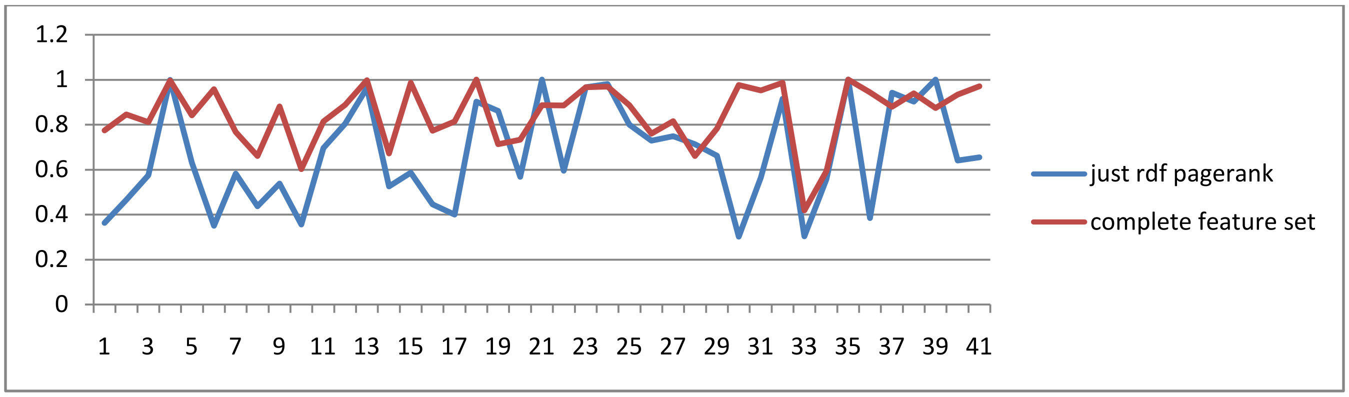

Figure 2 Comparison between nDCG obtained by ranking with just PageRank vs ranking with the model learned from all features

score if it points to nodes with high authority scores, and a node gets a high authority score if it is pointed to by nodes with high hub scores.

$$\forall p, \quad auth(p) = \sum_{i=1}^{n} hub(i)$$

$$\forall p, \quad hub(p) = \sum_{i=1}^{n} auth(i)$$

where p is a node in the graph, n is the total number of nodes connected to p, and i is a node connected to p. $auth(p)$ and $hub(p)$ are initialized to 1. In tables and graphs In tables and graphs we shall refer to the hub score as **Hub**, and to the authority score as **Auth**.

4.4 RDF Features

For every resource in DBpedia and Yago we have extracted simple domain-independent global features using various characteristics from the RDF graph. Table 1 shows the list of features which can be extracted for each of the resources.

Table 1 Description of RDF features

Feature	Description
nRSubj	number of relations where this resource appears as the subject
nRObj	number of relations where this resource appears as the object
nLiteral	number of relations where this resource appears as the subject and the object is a literal
nCat	number of Wikipedia categories this resource is in
sBgCat	size of the biggest category in which this resource is
sSmCat	size of the smallest category in which this resource is
sMdCat	size of the median category in which this resource is

4.5 Search Engine Based

We have used the search services provided by Yahoo! BOSS[3] to measure how many times the label of a resource from DBpedia or Yago appear on the internet. We do this by searching the web with the resource's label as a query and taking the number of hits as a feature (we shall refer to this feature in tables and graphs as **YnHits**). Moreover, we also extracted the top 5 keywords from the abstracts of Wikipedia pages in order to be used as queries for searching the web. This way we get 5 additional features which we shall call **Ykw1Hits**, **Ykw2Hits**, **Ykw3Hits**, **Ykw4Hits** and **Ykw5Hits**. Another feature which we compute using is by searching for the resource label and the question text together in a query. This feature is abbreviated as **YQHits**.

4.6 Google N-grams

Google released a dataset of all n-grams[4] (1-grams up to 5-grams) which appear on the internet at least 40 times,.together with their frequency counts. We consider as a feature of a resource the frequency count of its label in the n-gram dataset. In graphs and tables we shall refer to this feature as **NGram**.

4.7 Question-Answer Similarity

We constructed a text corpus from the descriptions of DbPedia and Yago resources. An extended text representation for each question was also compiled by doing web search with the question as the query and taking the text in the abstracts of the top 20 search results. The TFIDF [16] cosine similarity between the text of the question and the description of the answer resource was considered as a feature for ranking the answer. We shall refer to this feature as **qaSim**.

5. EXPERIMENTS

In this section we will focus on evaluation of ranking on sets of resources, which are results of SPARQL queries. This scenario closely matches a typical retrieval use case on top of, for example, Linked Open Data datasets. In the experiments we focused on ranking answers to 14 example queries for DBpedia and 27 example queries for Yago using features described in Section 4, and measuring the performance with the page popularity (PgPop) as golden standard.

The queries used in the experiments were SPARQL queries corresponding to questions such as: Which athlete was born in Philadelphia?, List of Schalke 04 players, Which countries have French as an official language?, Which objects are heavier than the Iosif Stalin tank? etc.

The point of the experiments is to see how effective each feature is in ranking the query results. Also, we look into ways of combining several features to learn a more accurate and more robust ranking model. Another question which we attempt to answer is to what extent a ranking model thus learned is

[3] http://developer.yahoo.com/search/boss/

[4] http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

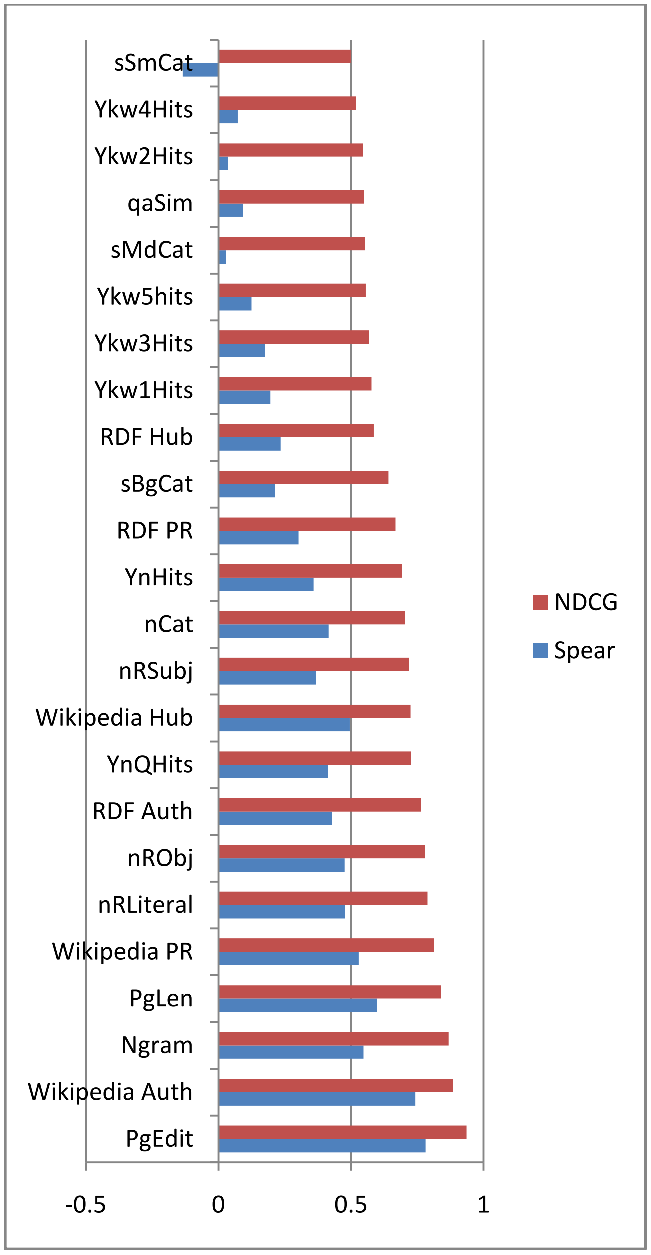

Figure 3 Performance obtained by ranking with one feature

Transferrable to other datasets. For instance, does a ranking model learned on DBPedia questions still perform well in ranking questions to YAGO? shows for each feature the performance of ranking the query results according to that feature only. The features are sorted descending by their nDCG score. We can observe that the features which we believe to approximate the importance of a Wikipedia page (PgEdit and PgLen) correlate very well with the page popularity. Other good features are the ones which are computed from Wikipedia pages (page rank, and hub and authority scores of Wikipedia pages). However, since these features are extracted from the same Wikipedia pages whose popularity we try to estimate, we consider it questionable to include them into our ranking model, and will leave them out in the rest of the experiments. The Google n-gram count appears to be a very good feature as well. The PageRank score on the RDF graph does not correlate particularly well with the page popularity, and even more, it is outperformed by simple

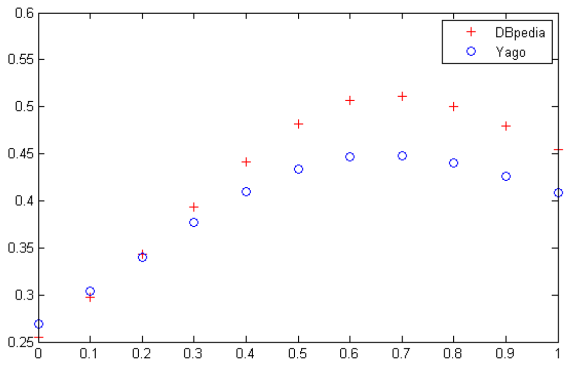

Figure 4 Correlations between number of visits and combined feature. The x-axis depicts different values of λ and y-axis depicts the correlation. Extreme left and right sides correspond to taking either only PageRank (left) or number of web hits (right).

features extracted from the RDF graph, like nRSubj, nRObj and nRLiteral. This is an important result because it seems that directly adapting the PageRank algorithm to RDF graphs is not necessary for a good ranking like it was assumed by a large part of the previous work in the semantic search field.

5.1 Combining Features

In what follows we evaluate the perspective of combining several features to arrive to a more robust ranking. To this end we analyzed combinations of two features, PageRank, f_{PR}, and number of web hits, f_{YnHits}, into a new feature defined as a convex combination of the two

$$f_{comb}(r) = \lambda f_{PR}(r) + (1-\lambda) f_{YnHits}(r).$$

Both features were normalized to values between 0.0 and 1.0. Figure 4 shows the correlation of new combined feature, parameterized by λ, with the golden standard. It can be seen that for both DBpedia and Yago, combination achieves a maximum around 0.7. There, the correlation of combined feature with golden standard is significantly higher than when taking any of the two features alone. The fact that such a simple combination of two features significantly outperforms the isolated features serves as a motivation for further investigation in building ranking models based on a collection of features.

Table 2 Feature Sets

FtrSet	Description
A	Complete feature set
B	Combined graph features (C + D + E)
C	Combination of nRSubj, nRObj and nLiteral
D	Hub and Authority scores
E	PageRank
F	NGram count
G	Combined search engine based features
H	YnHits alone
I	qaSim alone

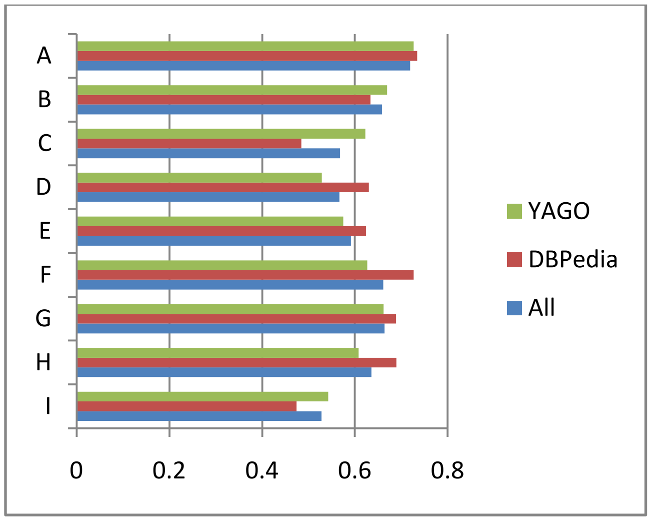

Figure 5 Pairwise Accuracy

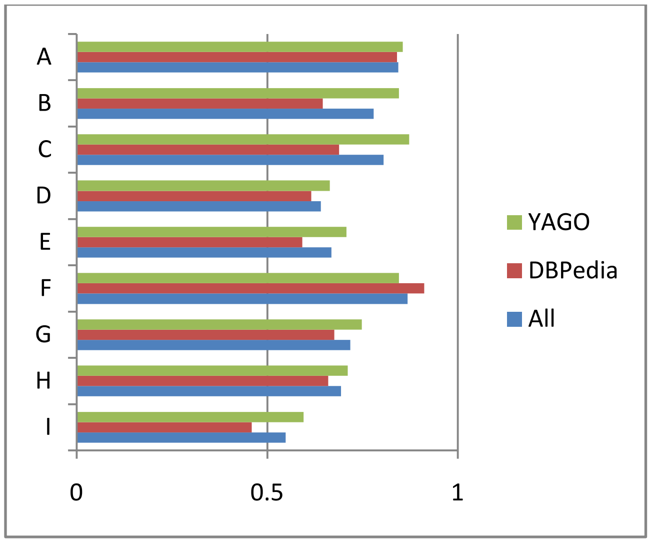

Figure 6 Normalized Discounted Cumulative Gain

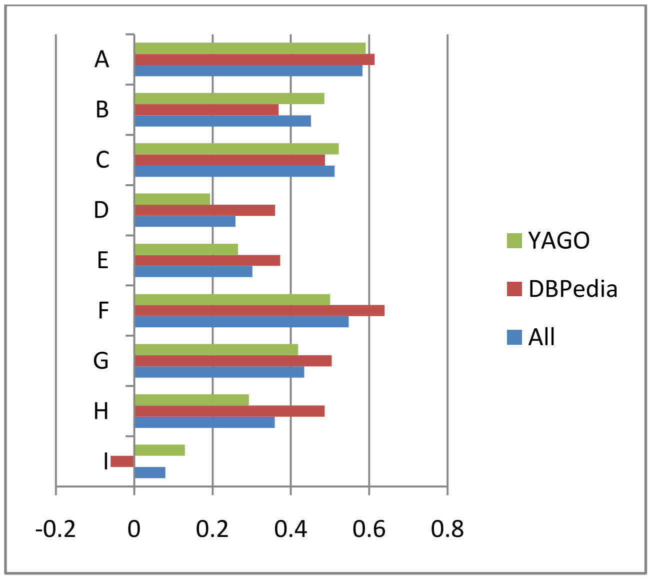

Figure 7 Spearman's correlation coefficient

In Table 2 we define nine feature sets (labeled A through I) on which we train ranking models by using the RankSVM algorithm described in Section 3.1. As the performance measures we use pairwise accuracy, normalized discounted cumulative gain, and Spearman's rank correlation coefficient. For the evaluation process we use leave one out cross-validation, meaning that each query in turn is taken aside as the test query, a ranking model is learned on the other queries and then the answers to the test query are ranked according to the ranking model. Figure 2 shows the nDCG score obtained for each question by ranking according to a model trained on the complete feature set A, as opposed to ranking according to PageRank only. It can be seen clearly that the performance of the model trained on the complete feature set dominates the performance of PageRank alone. More detailed experiments were performed, and the results are presented in Figure 5 (pairwise accuracy), Figure 6 (normalized discounted cumulative gain) and Figure 7 (Spearman's correlation coefficient). For each feature set the training was done once only on questions with answers from YAGO, once on questions from DBPedia and once on all questions. The scores for the separate datasets are shown as bars of different colors in the pictures. While there appear to be several strong feature sets, the complete feature set A is certainly one of the best, and even more, it has the important property of being consistent across different data sets. Feature set F, which consists of a single feature has a good score, but this score depends on which dataset the experiment was run on. The lesson we have learned from these experiments is that a combination of various features can improve not only the performance, but also the robustness of the ranking model.

The reslts also show that the n-gram feature performs better than the search engine based features. This is an important finding because extracting features from n-gram bases is cheaper and more scalable that issuing queries to search engines.

5.2 Domain Independence of Ranking Models

Ideally a ranking model for semantic search should be easily applied across different domains. A model trained on one data set should still have a good performance when used on a data set with a different RDF graph. In this section we explore in how far this is possible by training a ranking model on DBPedia questions and then evaluating it on ranking answers to YAGO questions. Similarly, we train a ranking model on YAGO questions and evaluate it on DBPedia questions. We use the feature sets defined in Table 2, except feature se I which was omitted due to poor performance. The evaluation measures are normalized discounted cumulative gain, shown in Figure 8, and Spearman's correlation coefficient, shown inFigure 9. It can be seen that the scores, although being comparable to the ones obtained in the previous experiment, are very much different depending on which data set was used as the training data. The only feature set which is consistent and robust is feature set A (a feature set combining all features). This experiment shows that in general a model learned on one dataset is not transferrable to another one unless it has been trained on a combination of many features such as feature set A. Our analysis also reveals that the biggest inconsistencies occur on the models trained on feature sets containing graph based features (i.e. feature sets B, C, D and E). The text based feature sets (F, G and H) also have inconsistent performance, but this inconsistency is smaller. This behavior confirms the intuition that RDF graphs preserve particularities reflecting the ontology designer's decisions. This makes it difficult to develop general ranking models, but combining text based features with graph based features can improve generality and domain independence.

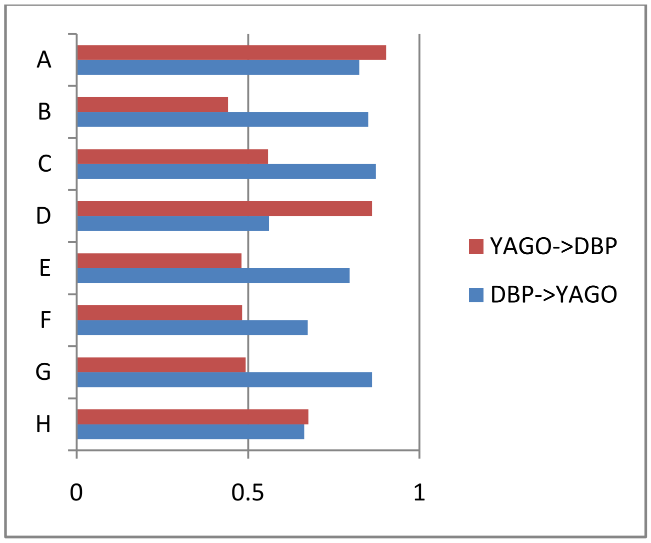

Figure 8 Normalized Discounted Cumulative Gain

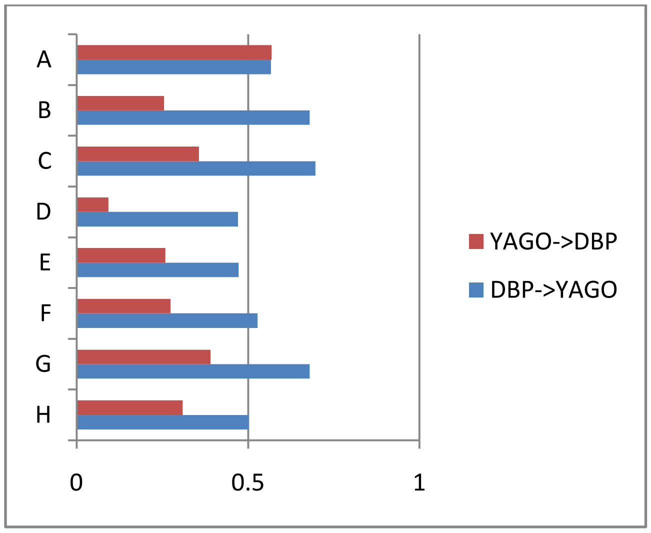

Figure 9 Spearman's correlation coefficient

6. CONCLUSIONS

In this paper we have analyzed the utility of individual and combined features to rank resources in Linked Open Data datasets. We have focused on two datasets derived from Wikipedia, namely DBpedia and Yago, and defined golden standard for retrieval by taking popularity scores from Wikipedia (number of visits, article length and number of edits). Such setup enabled us extensive experimentation of ranking within LOD datasets, which was until now constrained by expensive user studies and biased towards user groups evaluating the results.

In the experiments we have shown that PageRank alone does not correlate with popularity scores in a high and stable manner. Furthermore, we have shown that there are several other easy to compute features, which can be used for ranking and can achieve as high as performance as PageRank (e.g. RDF based features) or even significantly higher (web search hits). Nonetheless, all features were show to have a high variance in their performance, depending on the query and dataset.

We have also shown that a trained a statistical ranking model, which can combine several features into one final score, can lead to a better and more stable performance. Most importantly, the experiments show that the model, trained on one dataset, can be successfully transferred not only across queries but also across datasets.

As part of future work we plan to further investigate transferability of models across LOD datasets. Another potential field of work is towards using scores from ranking models to better guide reasoning engine in large scale reasoning scenarios.

7. REFERENCES

[1] Cutrell E., Guan, Z., What are you looking for? An eye-tracking study of information usage in Web search, SIGCHI 2007

[2] Page, L., Brin, S., Motowani, R., Winograd, T., The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Libraries, 1998

[3] Ge, W., Chen, J., Hu, W., Qu, Y., Object Link Structure in the Semantic Web, ESWC 2010

[4] Delbru, R., Toupikov, N., Castata, M., Tummarello, G., Decker, S., Hierarchical Link Analysis for Ranking Web Data, ESWC 2010

[5] Jeh, G., Widom, J., Scaling personalized web search, WWW 2003

[6] Hristidis V., Hwang, H., Papakonstantinou, Y., Authority-Based Keyword Search in Databases, ACM Transactions on database Systems, 2008

[7] Chakrabarti, S., Dynamic Personalized Pagerank in Entity-Relation Graphs, WWW 2007

[8] Nie, Z., Zhang, Y., Wen, J., Ma, W., Object-Level ranking: Bringing Order to Web Objects, WWW 2005

[9] Xi, W., Zhang, B., Chen, Z., Lu, Y., Yan, S., Ma, W., Fox, E., Link Fusion: A unified Link Analysis Framework for Multi-Type Interrelated Data Objects, WWW 2004

[10] Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S., DBpedia – A Crystallization Point for the Web of Data. Journal of Web Semantics: Science, Services and Agents on the World Wide Web, Issue 7, Pages 154–165, 2009.

[11] Suchanek, F., Kasenci, G., Weikum, G., Yago: A Large Ontology from Wikipedia and WordNet, Web Semantics: Science, Services and Agents on the World Wide Web, 2008

[12] Fellbaum, C., Wordnet: An Electronic Lexical Database, 1998

[13] Joachims, T.,Making Large-Scale SVM Learning Practical. Advances in Kernel-Methods - Support Vector Learning, B. Scholkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

[14] Joachims, T., Optimizing search engines using clickthrough data. In Proceedings of the eight ACM SIGKDD international Knowledge discovery and data mining, 2002

[15] Rupnik, J. Stochastic subgradient approach for solving linear support vector machines, SiKDD, 2008.

[16] Salton, G. Developments in Automatic Text Retrieval. Science, Vol 253, 974-979, 1991

[17] Tie-Yan Liu. Learning to Rank for Information Retrieval. Foundations and Trends in Information Retrieval. Vol. 3: No 3, pp. 225–331, 2009