Relevance Feedback Between Hypertext and Semantic Web Search: Frameworks and Evaluation

Harry Halpin [a,1,*], Victor Lavrenko [a],

[a] *University of Edinburgh, 10 Crichton St., Edinburgh EH8 9AB UK*

Abstract

We investigate the possibility of using Semantic Web data to improve hypertext Web search. In particular, we use relevance feedback to create a 'virtuous cycle' between data gathered from the Semantic Web of Linked Data and web-pages gathered from the hypertext Web. Previous approaches have generally considered the searching over the Semantic Web and hypertext Web to be entirely disparate, indexing, and searching over different domains. While relevance feedback has traditionally improved information retrieval performance, relevance feedback is normally used to improve rankings over a single data-set. Our novel approach is to use relevance feedback from hypertext Web results to improve Semantic Web search, and results from the Semantic Web to improve the retrieval of hypertext Web data. In both cases, an evaluation is performed based on certain kinds of informational queries (abstract concepts, people, and places) selected from a real-life query log and checked by human judges. We evaluate our work over a wide range of algorithms and options, and show it improves baseline performance on these queries for deployed systems as well, such as the Semantic Web Search engine FALCON-S and Yahoo! Web search. We further show that the use of Semantic Web inference seems to hurt performance, while the pseudo-relevance feedback increases performance in both cases, although not as much as actual relevance feedback. Lastly, our evaluation is the first rigorous 'Cranfield' evaluation of Semantic Web search.

Key words: semantic search, query logs, information retrieval, relevance feedback, evaluation, Linked Data

1. Introduction

There has recently been a return of interest in semantic search. This seems inspired in part by the Semantic Web, in particular by the Linked Data initiative's releasing of a massive amount of public structured data on the Web from a diverse range of sources in common Semantic Web formats like RDF (Resource Description Format) [15]. This has in turn led to the rise of specialized Semantic Web search engines. Semantic Web search engines are search engines that specifically index and return ranked Linked Data in RDF in response to keyword queries, but their rankings are much less well-studied than hypertext Web rankings, and so are thought likely to be sub-optimal.

One hypothesis put forward by Baeza-Yates is that the search on the Semantic Web can be used to improve traditional ad-hoc information retrieval for hypertext Web search engines and vice-versa [2]. While we realize the amount and sources of structured data on the Web are huge, to restrict and test the hypothesis of Baeza-Yates, from hereon we will assume that 'semantic search' refers to indexing and retrieving of Linked Data by search engines like Sindice and FALCON-S [6], and hypertext search

* Corresponding author. Tel: +44 131 650 4421
Email addresses: H.Halpin@ed.ac.uk (Harry Halpin), vlavrenk@inf.ed.ac.uk (Victor Lavrenko).

[1] Support provided by Microsoft "Beyond Search" award.

refers to the indexing and retrieval of hypertext documents on the World Wide Web by search engines like Google and Yahoo! Search.

We realize that our reduction of 'semantic search' to keyword-based information retrieval over the Semantic Web is very restrictive, as many people use 'semantic search' to mean simply search that relies on anything beyond surface syntax, including the categorization of complex queries [3] and entity-recognition using Semantic Web ontologies [11]. We will not delve into an extended explanation of the diverse kinds of semantic search, as surveys of this kind already exist [19]. Yet given the relative paucity of publicly accessible data-sets about the wider notion of semantics and the need to start with a simple rather than complex paradigm, we will restrict ourselves to the Semantic Web and assume a traditional, keyword-based ad-hoc information retrieval paradigm for both kinds of search, leaving issues like complex queries and natural language semantics for future research. Keyword search consisting of 1-2 terms should also be explored as it is the most common kind of query in today's Web search regardless of whether any results from this experiment can generalize to other kinds of semantic search [27].

Until recently semantic search suffered from a lack of a thorough and neutral Cranfield-style evaluation, and so we carefully explain and employ the traditional information retrieval evaluation frameworks in our experiment to evaluate semantic search. At the time of the experiment, our evaluation was the first Cranfield-style evaluation for searching on the Semantic Web. This evaluation later generalized into the annual 'Semantic Search' competition, [2] which has since become a standard evaluation for search over RDF data [4]. However, our particular evaluation presented here is still the only evaluation to determine relevance judgments over both hypertext and RDF using the same set of queries.

2. Overview

Our hypothesis is that relevance feedback can improve the ranking of relevant results for both hypertext Web search and Semantic Web search. Previous approaches have assumed that the Semantic Web and the hypertext Web to be entirely disparate, indexing and searching them differently [6]. Our novel approach is to use relevance feedback from hypertext Web search to improve the retrieval of semantic search. Then more interestingly, we attempt to run the relevance feedback in reverse: Assuming we have Semantic Web data, can it be used to improve hypertext search? This is not unfeasible, as one could consider the consumption of Semantic Web data by a program to be a judgment of relevance.

Relevance feedback is the *use of explicit relevance judgments from users of a query in order to expand the query.* By 'expand the query,' we mean that the usually rather short query is expanded into a much larger query by adding words from known relevant documents. For example, a query on the hypertext Web for the Eiffel Tower given as 'eiffel' might be expanded into 'paris france eiffel tour.' If the relevant pages instead were about an Eiffel Tower replica in Texas, the same results query could be expanded into 'paris texas eiffel replica.' The same principle applies to the Semantic Web, except that the natural language terms may include Semantic Web URIs and terms resulting from inference or URI processing. The hypothesis of relevance feedback, as pioneered by Rocchio in the SMART retrieval system, is that the relevant documents will disambiguate and in general give a better description of the information need of the query than the query itself [26]. Relevance feedback has been shown in certain cases to improve retrieval performance significantly, like relevance modeling (as formalized by Lavrenko et al. [16]) that creates relevance models directly from the indexed documents rather than explicitly waiting for the user to make a relevance judgment. Relevance feedback has never been used with semantic search, and furthermore, it is unknown whether or not it helps. In many cases, relevance feedback, particularly over noisy or sparse data, actually decreases performance of search engines, and this could be especially true on the relatively messy data of the Web.

In Section 3 we first elucidate the general nature of semantic search in comparison to hypertext search by running a general open-domain collection of user queries from a real hypertext query-log against the Semantic Web and then have human judges construct a 'gold-standard' collection of queries and results judged for relevance, from both the Semantic and hypertext Web. Then in Section 4 we give a brief overview of information retrieval frameworks and ranking algorithms. While this section may be of interest to Semantic Web researchers unfamiliar with such techniques, information retrieval researchers may wish to proceed immediately past this

[2] Sponsored by Yahoo! Research for both 2010 and 2011.

section. Our system is described in Section 5. In Section 6, these techniques are applied to the 'gold standard' collection created in Section 3 so that the best parameters and algorithms for relevance feedback for both hypertext and semantic search can be determined. In Section 7 and Section 8 the effects of using pseudo-feedback and Semantic Web inference are evaluated. The system is evaluated against 'real-world' deployed systems in Section 9. Finally, in Section 10 future work is detailed, and conclusions are given in Section 11.

3. Is There Anything Worth Finding on the Semantic Web?

In this section we demonstrate that the Semantic Web does indeed contain information relevant to ordinary users by sampling the Semantic Web according to a real-world queries referring to entities and concepts from the query log of a major search engine. The main problem confronting of any study of the Semantic Web is one of *sampling*. As almost any large-data database can easily be exported to RDF, statistics demonstrating the actual deployment of the Semantic Web can be biased by the automated release of large, if useless, data-sets, the equivalent of 'Semantic Web' spam. Also, large specialized databases like Bio2RDF can easily dwarf the rest of the Semantic Web in size. A more appropriate strategy would be to try to answer the question: What information is available on the Semantic Web that users are actually interested in? The first large-scale analysis of the Semantic Web was done via an inspection of the index of Swoogle by Ding and Finin [9]. The primary limitation of that study was that the large majority of the Semantic Web resources sampled did not contain rich information that many people would find interesting. For example, the vast majority of data on the Semantic Web in 2006 was Livejournal exporting every user's profile as FOAF and RSS 1.0 data that used Semantic Web techniques to structure the syntax of news feeds. Yet with information-rich and interlinked databases like Wikipedia being exported to the Semantic Web, today the Semantic Web may contain information needed by actual users. As there is no agreed-upon fashion to sample the Semantic Web (and the entire Web) in a fair manner, we will for our evaluation create a sample driven by queries from real-users using easily-accessible search engines that claim to have a Web-scale index, although independent verification of this is difficult if not impossible.

3.1. *Inspecting the Semantic Web*

In order to select real queries from users for our experiment, we used the query log of a popular hypertext search engine, the Web search query log of approximately 15 million distinct queries from Microsoft Live Search. This query log contained 6,623,635 unique queries corrected for capitalization. The main issue in using a query log is to get rid of navigational and transactional queries. A straightforward gazetteer-based and rule-based named entity recognizer was employed to discover the names of people and places [20], based off a list of names maintained by the Social Security Administration and a place name database provided by the Alexandria Digital Library Project. From the query log a total of 509,659 queries were identified as either people or places by the named-entity recognizer, and we call these queries *entity queries*. Employing WordNet to represent abstract concepts, we chose queries recognized by WordNet that have *both* a hyponym and hypernym in WordNet. This resulted in a more restricted 16,698 queries that are supposed to be about abstract concepts, which we call *concept queries*.

A sample entity query from our list would be 'charles darwin,' while a sample concept query would be 'violin.' In our data-set using hypertext search, both queries return almost all relevant results. The query 'charles darwin' gives results that are entirely encyclopedia pages (Wikipedia, eHow, `darwin-online.org.uk`) and other factual sources of information, while 'violin' returns 8 out of 10 factual pages, with 2 results just being advertisements for violin makers. On the contrary for the Semantic Web, the query 'charles darwin' had 6 relevant results, with the rest being for places such as the city of Darwin and books or products mentioning Darwin. For 'violin,' only 3 contain relevant factual data, with the rest being the names of albums called 'Violin' and movies such as 'The Violin Maker.' From inspection of entities with relevant results, it appears the usual case for semantic search is that DBpedia and WordNet have a substantial amount of overlap in the concepts to which they give URIs. For example, they have distinct URIs for such concepts as 'violin' (`http://dbpedia.org/resource/Violin` vs. W3C WordNet's `synset-violin-noun-1`). Likewise, most repetition of entity URIs comes from

WordNet and DBpedia, both of which have distinct URIs for famous people like Charles Darwin. In many cases, these URIs do not always appear at the top, but in the second or third position, with often an irrelevant URI at top. Lastly, much of the RDF that is retrieved seems to have little information in it, with DBPedia and WordNet being the most rich sources of information.

The results of running the selected queries against a Semantic Web search engine, FALCON-S's Object Search [6], were surprisingly fruitful. For entity queries, there was an average of 1,339 URIs (S.D. 8,000) returned for each query. On the other hand, for concept queries, there were an average of 26,294 URIs (S.D. 14,1580) returned per query, with no queries returning zero documents. Such a high standard deviation in comparison to the average is a sure sign of a non-normal distribution such as a power-law distribution, and normal statistics such as average and standard deviation are not good characteristic measures of such distributions. As shown in Figure 1, when plotted in logarithmic space, both entity queries and concept queries show a distribution that is heavily skewed towards a very large number of high-frequency results, with a steep drop-off to almost zero results instead of the characteristic long tail of a power law. For the vast majority of queries, far from having no information, the Semantic Web of Linked Data appears to have *too much data*, but for a minority of queries there is just *no data*. This is likely the result of the releasing of Linked Data in large 'chunks' from data-silos about specific topics rather than the more organic development of the hypertext Web that typically results in power-law distributions. Also, note that hypertext web-pages are updated as regards trends and current events much more quickly than the relatively slow-moving world of Linked Data.

Another question is whether or not there is any correlation between the amount of URIs returned from the Semantic Web and the popularity of the query. As shown by Figure 2, there is *no* correlation between the amount of URIs returned from the Semantic Web and the popularity of the query. For entity queries, the correlation coefficient was 0.0077, while for concept queries, the correlation coefficient was still insignificant, at 0.0125. The popularity of query is not related to how much information the Semantic Web possesses on the information need expressed by the query: Popular queries may have little data, while infrequent queries may have a lot. This is likely due to the rapidly changing and event-

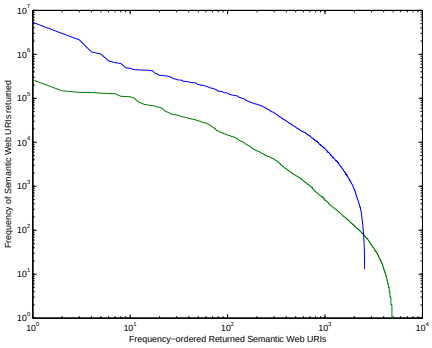

Fig. 1. The rank-ordered frequency distribution of the number of URIs returned from entity and concept queries, with the entity queries given by green and the concept queries by blue.

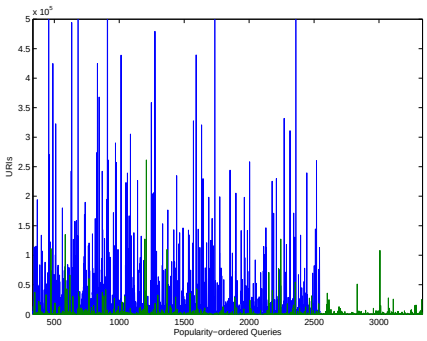

Fig. 2. The rank-ordered popularity of the queries is on the x-axis, with the y axis displaying the number of Semantic Web URIs returned, with the entity queries given by green and the concept queries by blue.

Entity	Concept
ashville north carolina	sociology
harry potter	clutch
orlando florida	telephone
ellis college	ale
university of phoenix	pillar
keith urban	sequoia
carolina	aster
el salvador	bedroom
san antonio	tent
earl may	cinch

Table 1
10 Selected Entity and Concept Queries

dependent nature of hypertext Web queries versus the Semantic Web's preference for more permanent and less temporally-dependent data. For a more full exploration of the data-set used in this experiment, including types of URIs, see the paper on 'A Query-Driven Characterization of Linked Data' [12]. Since this data was collected in spring of 2009 it may not be currently accurate as a characterization of either FALCON-S or the state of Linked Data currently, but for evaluation purposes this sample should suffice, and using random selections from a real human query log is a definite advance, as randomly sampling all of Linked Data would result in an easily biased evaluation, away from what human users are interested in and towards what happens to be available as Linked Data.

Surprisingly, there is a large amount of information that may be of interest to ordinary hypertext users on the Semantic Web, although there is no correlation between the popularity of queries and the availability of that information on the Semantic Web. The Semantic Web is not irrelevant to ordinary users as there is data on the Semantic Web ordinary users are interested in, even if it is distributed unevenly and does not correlate with the popularity of their queries.

3.2. *Selecting Queries for Evaluation*

In order to select a subset of informational queries for evaluation, we randomly selected 100 queries identified as abstract concepts by WordNet and then 100 queries identified as either people or places by the named entity recognizer, for a total of 200 queries to be used in evaluation. Constraints were placed on the URIs resulting from semantic search, such that at least 10 Semantic Web documents (a file containing a valid RDF graph) had to be retrieved from the URI returned by the Semantic Web search engine. This was necessary as some queries returned zero or less than 10 URIs, as explained in Section 3.1. For each query, hypertext search always returned more than 10 URIs. So for each query, 10 Semantic Web documents were retrieved using the FALCON-S Object Search engine [6], leading to a total of 1,000 Semantic Web documents about entities and 1,000 Semantic Web documents about concepts, for a total of 2,000 Semantic Web documents for relevance judgments. Then, the same experimental query log was used to retrieve pages from the hypertext Web using Yahoo! Web search, resulting in the same number of web-pages about concepts and entities (2,000 total) for relevance judgments. The total number of all Semantic Web documents and hypertext web-pages gathered from the queries is 4,000.

The queries about entities and concepts are spread across quite diverse domains, ranging from entities about both locations (El Salvador) and people (both fictional such as Harry Potter and non-fictional such as Earl May) to concepts ranging over a large degree of abstraction, from sociology to ale. A random selection of ten queries from the entity and concept queries is given in Table 1. This set of 4,000 hypertext web-pages and Semantic Web documents are then used to evaluate our results in Section 6.

3.3. *Relevance Judgments*

For each of the 200 queries selected in Section 3.2, 10 hypertext web-pages and 10 Semantic Web documents need to be judged for relevance by three human judges, leading to a total of 12,000 judgments for relevance for our entire experiment, with the correct relevance determined by 'voting' amongst the three judges per document. Human judges each judged 25 queries presented in a randomized order, and were given a total of 3 hours to judge the entire sample for relevancy. No researchers were part of the rating. The judges were each presented first with ten hypertext web-pages and then with ten Semantic documents that could be about the same query. Before starting judging, the judges were given instructions and trained on 10 sample results (5 web-pages and 5 Semantic Web documents). The human judges were forced to make binary judgments of relevance, so each result must be either relevant or irrelevant to the query. They were given the web-page selected by the human user from the query log as a 'gold standard' to determine the meaning of the keyword.

The standard TREC definition for relevance is "If you were writing a report on the subject of the topic and would use the information contained in the document in the report, then the document is relevant" [13]. As semantic search is supposed to be about entities and concepts rather than documents, semantic search needs an definition of relevance based around information about entities or concepts independent of documents. In one sense, this entity-centric relevance should have both a wider remit than document-centric relevance definition, as any information about the entity that could be relevant should be included. Yet in another sense, this definition is more restrictive, as if one considers the world (perhaps fuzzily) partitioned into distinct entities and concepts, then merely related information would not count. In the instructions, relevance was defined *as whether or not a result is about the same thing as the query, which can be determined by whether or not accurate information about the information need is expressed by the result.* The following example was given to the judges: "Given a query for 'Eiffel Tower,' a result entitled 'Monuments in Paris' would likely be relevant if there was information about the Eiffel Tower in the page, but a result entitled 'The Restaurant in the Eiffel Tower' containing only the address and menus of the restaurant would not be relevant."

Kinds of Web results that would ordinarily be considered relevant are therefore excluded. In particular, there is a restriction that the relevant information must be present in the result itself. This excludes possibly relevant information that is accessible via outbound links, even a single link. All manner of results that are collections of links are thus excluded from relevancy, including both 'link farms' purposely designed to be highly ranked by page-rank based search engines, as well as legitimate directories of high-quality links to relevant information. These hubs are excluded precisely because the information, even if it is only a link transversal away, is still not directly present in the retrieved result. By this same principle, results that merely redirect to another resource via some method besides the standardized HTTP methods are excluded, since a redirection can be considered a kind of link. They would be considered relevant only if additional information was included in the result besides the redirection itself.

In order to aid the judges, a Web-based interface was created to present the queries and results to the judges. Although an interface that presented the queries and the search interface in a manner similar to search engines was created, human judges preferred an interface that presented them the results for judgments one-at-a-time, forcing them to view a rendering of the web-page associated with each URI originally offered by the search engine. For each hypertext web-page, the web-page was rendered using the Firefox Web Browser and PageSaver Pro 2.0. For each Semantic Web document, the result was rendered (i.e. the triples and any associated text in the subject) by using the open-source Disco Hyperdata Browser with Firefox.[3] In both cases, the resulting rendering of the Web representation was saved at 469×631 pixel resolution. The reason that the web-page was rendered instead of a link given directly to the URI is because of the unstable state of the Web, especially the hypertext Web. Even caching the HTML would have risked losing much of the graphic element of the hypertext Web. By creating 'snapshot' renderings, each judge at any given time was guaranteed to be presented with the result in the same visual form. One side-effect of this is that web-pages that heavily depend on non-standardized

[3] The Disco Hyperdata Browser, a browser that renders Semantic Web data to HTML, is available at `http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/`.

technologies or plug-ins would not render and were thus presented as blank screen shots to the user, but this formed a small minority of the data. The user-interface divided the evaluation into two steps:

– *Judging relevant results from a hypertext Web search:* The judge was given the search terms created by an actual human user for a query and an example relevant web-page whose full snapshot could be viewed by clicking on it. A full rendering of the retrieved web-page was presented to the user with its title and summary (as produced by Yahoo! Search) easily viewed by the judge as in Figure 3. The judge clicked on the check-box if the result is considered relevant. Otherwise, the web-page was by default recorded as not relevant. The web-page results were presented to the judge one at a time, ten times for each query.
– *Judging relevant results from a Semantic Web search:* Next, the judge assessed all the Semantic Web results for relevancy. These results were retrieved from the Semantic Web using the same interface displayed to the judge in the first step as shown in Figure 4, and a title was displayed by retrieving any literal values from `rdfs:label` properties and a summary by retrieving any literal values from `rdfs:comment` values. Using the same interface as in the first step, the judge had to determine whether or not the Semantic Web results were relevant.

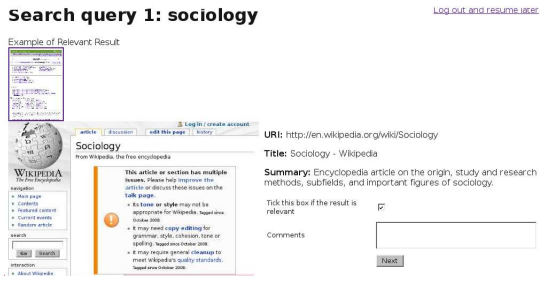

Fig. 3. The interface used to judge web-page results for relevancy.

After the ratings were completed, Fleiss' κ statistic was taken in order to test the reliability of inter-judge agreement on relevancy judgments [10]. Simple percentage agreement is not sufficient, as it does not take into account the likelihood of purely coincidental agreement by the judges. Fleiss' κ both corrects for chance agreement and can be used for more than two judges [10]. The null hypothesis is that the judges cannot distinguish relevant from irrelevant results, and so are judging results randomly. Overall, for both relevance judgments over Semantic Web

Fig. 4. The interface used to judge Semantic Web results for relevancy

results and web-page results, $\kappa = 0.5724$ ($p < .05$, 95% Confidence interval $[0.5678, 0.5771]$), indicating the rejection of the null hypothesis and 'moderate' agreement. For web-page results only, $\kappa = 0.5216$ ($p < .05$, 95% Confidence interval $[.5150, 0.5282]$), also indicating the rejection of the null hypothesis and 'moderate' agreement. Lastly, for only Semantic Web results, $\kappa = 0.5925$ ($p < .05$, 95% Confidence interval $[0.5859, 0.5991]$), also indicating the null hypothesis is to be rejected and 'moderate' agreement. So, in all cases there is 'moderate' agreement, which is sufficient given the general difficulty of producing perfectly reliable relevancy judgments. Interestingly enough, the difference in κ between the web-page results and Semantic Web results show that the judges were actually *slightly* more reliable in their relevancy judgments of information from the Semantic Web rather than the hypertext Web. This is likely due to the more widely varying nature of the hypertext results, as compared to the more consistent informational nature of Semantic Web results.

Were judges more reliable with entities or concepts? Recalculating the κ for all results based on entity queries, $\kappa = 0.5989$ ($p < .05$, 95% Confidence interval $[0.5923, 0.6055]$), while for all results based on concept queries was $\kappa = 0.5447$ ($p < .05$, 95% Confidence interval $[0.5381, 0.5512]$). So it appears that judges are slightly more reliable discovering information about entities rather than concepts, backing the claim made by Hayes and Halpin that there is more agreement in general about 'less' abstract things like people and places rather than abstract concepts [14]. However, agreement is still very similar and 'moderate' for both information about entities and concepts. It is perhaps due to the entity-centric and concept-centric definition of relevance that the agreement was not higher.

For the queries, much of the data is summarized in Table 3. ***Resolved*** queries are *queries that return at least one relevant result* in the top 10 re-

Results:	Hypertext	Semantic Web
Resolved:	197 (98%)	132 (66%)
Unresolved:	3 (2%)	68 (34%)
Top Relevant:	121 (61%)	76 (58%)
Top Non-Relevant:	76 (39%)	56 (42%)

Table 2
Results of Hypertext and Semantic Web Search Relevance Judgments: Raw numbers followed by percentages. The top two row percentages are with respect to all queries, while the latter two columns are with respect to the total of resolved queries.

sults, while ***unresolved*** queries are *queries that return no relevant queries in the top 10 results*. 'Hypertext' means that the result was taken only over the hypertext Web results and 'Semantic Web' indicates the same for the Semantic Web results. The percentages for resolved and unresolved for 'hypertext' and 'Semantic Web' were taken over *all* the hypertext and Semantic Web relevancy corpora in order to allow direct comparison. On the contrary, the percentages for 'Top Relevant' and 'Top Non-Relevant' were computed as percentages over only resolved queries, and so excludes unresolved queries. For ease of reference, a pie-chart for the hypertext relevancy is given in Figure 5 and for the Semantic Web relevancy in Figure 6.

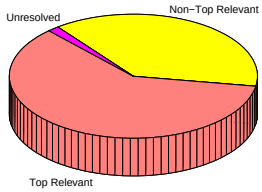

Fig. 5. Results of Querying the Hypertext Web.

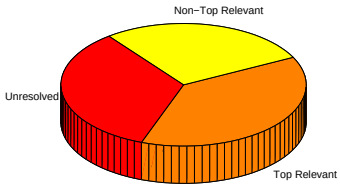

Fig. 6. Results of Querying the Semantic Web.

For both hypertext and Semantic search, there were 71 (18%) unresolved queries that did not have any results. For the hypertext Web search, only 3 (2%) queries were unresolved, while 68 (34%) of the queries were unresolved for the Semantic Web. This simply means that the hypertext search engines almost always returned at least one relevant result in the top 10, but that for the Semantic Web almost a third of all queries did not return any relevant result in the top 10. This only means there is much information that does not yet have a relevant form on the Semantic Web, unless it is hidden by the perhaps sub-optimal ranking by FALCON-S.

Another question is how many queries had a relevant result as their top result? In general, 197 queries (50%) had top-ranked relevant results over both Semantic Web and hypertext search. While the hypertext Web search had 121 (61%) top-ranked relevant results, the Semantic Web only had 76 (58%) top-ranked results. What is more compelling for relevance feedback is the number of relevant results that were *not* the top-ranked result. Again for both kinds of searches, there were 132 (33.0%) queries where a relevant result was *not* in the top position of the returned results. For the hypertext Web, there were 76 (39%) queries with a top non-relevant result. Yet for the Semantic Web there were 56 (42%) queries that had a top non-relevant result. So queries on the Semantic Web are more likely to turn up no relevant results in the top 10. When a relevant query is returned in the top 10 results it is quite likely that a non-relevant result will be in the top position for both the hypertext Web and the Semantic Web.

4. Information Retrieval for Web Search

In our evaluation we tested two general kinds of information retrieval frameworks: vector-space models and language models. In the *vector-space model*, document models are considered to be vectors of terms (usually called 'words' as they are usually, although not exclusively, from natural language, as we transform URIs into 'pseudo-words') where the weighing function and query expansion has no principled basis besides empirical results. Ranking is usually done via a comparison using the cosine distance, a natural comparison metric between vectors. The key to success with vector-space models tends to be the tuning of the parameters of their weighing function. While fine-turning these parameters has led to much practical success in information retrieval, the

parameters have little formally-proven basis but are instead based on common-sense heuristics like document length and average document length.

Another approach, the *language model* approach, takes a formally principled and probabilistic approach to determining the ranking and weighting function. Instead of each document being considered some parametrized word-frequency vector, the documents are each considered to be samples from an underlying probabilistic language model M_D, of which D itself is only a single observation. In this manner, the query Q can itself also be considered a sample from a language model. In early language modeling efforts the probability that the language model of a document would generate the query is given by the ranking function of the document. A more sophisticated approach to language models considers that the query was a sample from an underlying *relevance model* of unknown relevant documents, but that the model could be estimated by computing the co-occurrence of the query terms with every term in the vocabulary. In this way, the query itself was just considered a limited sample that is automatically expanded before the search has even begun by re-sampling the underlying relevance model.

In detail, we will now inspect the various weighting and ranking functions of the two frameworks. A number of different options for the parameters of each weighting function and the appropriate ranking function will be considered.

4.1. *Vector Space Models*

4.1.1. *Representation*

Each vector-space model has as a parameter the factor m, the maximum *window size*, which is the number of words, ranked in descending order of frequency, that are used in the document models. In other words, the size of the vectors in the vector-space model is m. Words with a zero frequency are excluded from the document model.

4.1.2. *Weighting Function: BM25*

The current state of the art weighting function for vector-space models is $BM25$, one of a family of weighting functions explored by Roberson [25] and a descendant of the *tf.idf* weighting scheme pioneered by Spärck Jones and Robertson [24]. In particular, we will use a version of *BM25* with the slight performance-enhancing modifications used in the InQuery system [1]. This weighting scheme has been carefully optimized and routinely shows excellent performance in TREC competitions [7]. The InQuery BM25 function assigns the following weight to a word q occurring in a document D:

$$D_q = \frac{n(q,D)}{n(q,D) + 0.5 + 1.5\frac{dl}{avg(dl)}} \frac{\log\left(0.5 + N/df(q)\right)}{\log\left(1.0 + \log N\right)} \quad (1)$$

The $BM25$ weighting function is summed for every term $q \in Q$. For every q, $BM25$ calculates the number of occurrences of a term q from the query in the document D, $n(q,D)$, and then weighs this by the length of document dl of document D in comparison to the average document length $avg(dl)$. This is in essence the equivalent of term frequency in $tf.idf$. The $BM25$ weighting function then takes into account the total number of documents N and the document frequencies $df(q)$ of the query term. This second component is the idf component of classical $tf.idf$.

4.1.3. *Ranking Function: Cosine and InQuery*

The vector-space models have an intuitive ranking function in the form of cosine measurements. In particular, the cosine ranking function is given by Equation 2, for a document D with query Q, where both D and Q contain q words, iterating over all words.

$$cos(D,Q) = \frac{D \cdot Q}{|D||Q|} = \frac{\sum_q Q_q D_q}{\sqrt{\sum_q Q_q^2}\sqrt{\sum_q D_q^2}} \quad (2)$$

The only question is whether or not the vectors should be normalized to have a Euclidean weight of 1, and whether or not the query terms themselves should be weighted. We investigate both options. The classical cosine is given as *cosine*, which normalizes the vector lengths and then proceeds to weight both the query terms and the vector terms by $BM25$. The version without normalization is called *inquery* after the *InQuery* system [1]. The *inquery* ranking function is the same as *cosine* except without normalization each word in the query can be considered to have uniform weighing.

4.1.4. *Relevance Feedback Algorithms: Okapi, LCA, and Ponte*

There are quite a few options on how to expand queries in a vector-space model. One popular and straightforward method, first proposed by *Rocchio* [26] and at one point used by the *Okapi* system [25],

is to expand the query by taking the average of the j total relevant document models R, with a document $D \in R$, and then simply replacing the query Q with the top m words from averaged relevant document models. This process is given by Equation 3 and is referred to as *okapi*:

$$okapi(Q) = \frac{1}{j} \sum_{D \in R} D \tag{3}$$

Another state of the art query expansion technique is known as *Local Content Analysis* (*lca*) [28]. Given a query Q with query terms $q_1 ... q_k$ and a set of results D and a set of relevant documents R, then *lca* ranks every $w \in V$ by Equation 4, where n is the size of the relevant documents R, idf_w is the inverse document frequency of word w, and D_q and D_w are the frequencies of the words w and $q \in Q$ in relevant document $D \in R$.

$$lca(w; Q) = \prod_{q \in Q} \left(0.1 + \frac{1/\log n}{1/idf_w} \log \sum_{r \in R} D_q D_w \right)^{idf_q} \tag{4}$$

After each word $w \in V$ has been ranked by *lca*, then the query expanded by LCA is just the top m words given by *lca*. Local Content Analysis attempts to select words from relevant documents to expand the query that have limited ambiguity, and so it does extra processing compared to the *okapi* method that simply averages the most frequent words in the relevant documents. In comparison, Local Content Analysis performs an operation similar in effect to $tf.idf$ on the possibly relevant terms, and so attempting by virtue of weighing to select only words w that both appear frequently with terms in query q but have a low overall frequency (idf_w) in all the results.

The final method we will use is the heuristic method developed by Ponte [22], which we call *ponte*. Like *lca*, *ponte* ranks each word $w \in V$, but it does so differently. Instead of taking a heuristic-approach like *Okapi* or LCA, it takes a probabilistic approach. Given a set of relevant documents $R \in D$, Ponte's approach estimates the probability of each word $w \in V$ being in the relevant document, $P(w|D)$, divided by its overall probability of the word to occur in the results $P(w)$. Then the *Ponte* approach gives each $w \in V$ a score as given in Equation 5 and then expands the query by using the m most relevant words as ranked by their scores.

$$Ponte(w; R) = \sum_{D \in R} log \left(\frac{P(w|D)}{P(w)} \right) \tag{5}$$

4.2. *Language Models*

4.2.1. *Representation*

Language modeling frameworks in information retrieval represent each document as a language model given by an underlying multinomial probability distribution of word occurrences. Thus, for each word $w \in V$ there is a value that gives how likely an observation of word w is given D, i.e. $P(w|u_D(v))$. The document model distribution $u_D(v)$ is then estimated using the parameter ϵ_D, which allows a linear interpolation that takes into account the background probability of observing w in the entire collection C. This is given in Equation 6.

$$u_D(w) = \epsilon_D \frac{n(w, D)}{|D|} + (1 - \epsilon_D) \frac{n(w, C)}{\sum_{v \in V} n(v, C)} \tag{6}$$

The parameter ϵ_D just takes into account the relative likelihood of the word as observed in the given document D compared to the word given the entire collection of documents C. $|D|$ is the total number of words in document D, while $n(w, D)$ is the frequency of word d in document D. Further, $n(w, C)$ is the frequency of occurrence of the word w in the entire collection C divided by the occurrence of all words v in collection C.

4.2.2. *Language Modeling Baseline*

When no relevance judgments are available, the language modeling approach ranks documents D by the probability that the query Q could be observed during repeated random sampling from the distribution $u_D(\cdot)$. The typical sampling process assumes that words are drawn independently, with replacement, leading to the following retrieval score being assigned to document D:

$$P(Q|D) = \prod_{q \in Q} u_D(q) \tag{7}$$

The ranking function in Equation 7 is called *query-likelihood* ranking and is used as a baseline for our language-modeling experiments.

4.2.3. *Language Models and Relevance Feedback*

The classical language-modeling approach to IR does not provide a natural mechanism to perform relevance feedback. However, a popular extension of the approach involves estimating a relevance-based model u_R in addition to the document-based model u_D, and comparing the resulting language models using information-theoretic measures. Estimation of

u_D has been described above, so this section will describe two ways of estimating the relevance model u_R, and a way of measuring distance between u_Q and u_D for the purposes of document ranking.

Let $R = r_1 \ldots r_k$ be the set of k relevant documents, identified during the feedback process. One way of constructing a language model of R is to average the document models of each document in the set:

$$u_{R,avg}(w) = \frac{1}{k} \sum_{i=1}^{k} u_{r_i}(w) = \frac{1}{k} \sum_{i=1}^{k} \frac{n(w, r_i)}{|r_i|} \quad (8)$$

Here $n(w, r_i)$ is the number of times the word w occurs in the $i'th$ relevant document, and $|r_i|$ is the length of that document. Another way to estimate the same distribution would be to *concatenate* all relevant documents into one long string of text, and count word frequencies in that string:

$$u_{R,con}(w) = \frac{\sum_{i=1}^{k} n(w, r_i)}{\sum_{i=1}^{k} |r_i|} \quad (9)$$

Here the numerator $\sum_{i=1}^{k} n(w, r_i)$ represents the total number of times the word w occurs in the concatenated string, and the denominator is the length of the concatenated string. The difference between Equations 8 and 9 is that the former treats every document equally, regardless of its length, whereas the latter favors longer documents (they are not individually penalized by dividing their contributing frequencies $n(w, r_i)$ by their length $|r_i|$).

4.2.4. *Ranking Function: Cross Entropy*

We now want to re-compute the retrieval score of document D based on the estimated language model of the relevant class u_R. What is needed is a principled way of comparing a relevance model u_R against a document language model u_D. One way of comparing probability that has shown the best performance in empirical information retrieval research [16] is cross entropy. Intuitively, cross entropy is an information-theoretic measure that measures the average number of bits needed to identify the probability of distribution p being generated if p was encoded using given probability distribution p rather than q itself. For the discrete case this is defined as:

$$H(p, q) = -\sum_{x} p(x) log(q(x)) \quad (10)$$

If one considers that the $u_R = p$ and that document model distribution $u_D = q$, then the two models can be compared directly using cross-entropy, as shown in Equation 11. This use of cross entropy also fulfills the Probability Ranking Principle and so is directly comparable to vector-space ranking via cosine [16].

$$-H(u_R||u_D) = \sum_{w \in V} u_R(w) \log u_D(w) \quad (11)$$

Note that either the *averaged* relevance model $u_{R,avg}$ or the *concatenated* relevance model $u_{R,con}$ can be used in Equation 11. We refer to the former as rm and to the latter as tf in the following experiments.

5. System Description

We present a novel system that uses the same underlying information retrieval system on both hypertext and Semantic Web data so that relevance feedback can be done in a principled manner from both sources of data with language models. In our system, the query is run first against the hypertext Web and relevant hypertext results can then be used to expand a Semantic Web search query with terms from resulting hypertext web-pages. The expanded query is then ran against the Semantic Web, resulting in a different ranking of results than the non-expanded query. We can also then run the process backwards, using relevant Semantic Web data as relevance feedback to improve hypertext Web search.

This process is described using pseudo-code in Figure 7 where the set of all queries to be ran on the system is given by the *QuerySet* parameter. The two different kinds of relevance feedback are given by the *SearchType* parameter, with *SearchType=RDF* for searching over RDF data using HTML documents as data for relevance feedback-based query expansion, and *HTML* for searching over HTML documents with RDF as the data for relevance-feedback query expansion. *Representation* is the internal data model used to represent the documents, either vector-space models or language models. The feedback used to expand the query is given by *Feedback* with the kind of relevance feedback algorithm used to expand the query is given by *Algorithm*, which for relevance models are directly built into the representation. The ranking function (cross-entropy for language models, or some variation of cosine for vector-space models) is given by *Ranking*. The final results for each query are presented to the user in *PresentResults*.

Algorithm
5.1: SEARCH($QuerySet, SearchType$)

if $SearchType = RDF$

$$\begin{cases} Data1 \in Representation(HTMLdata) \\ Data2 \in Representation(RDFdata) \end{cases}$$

else $SearchType = HTML$

$$\begin{cases} FeedbackData \in Representation(RDFdata) \\ ResultData \in Representation(HTMLdata) \end{cases}$$

for each $Query \in QuerySet$

$$\begin{cases} FeedbackResults \leftarrow Feedback(Query, Data1) \\ ExpandedQuery \leftarrow Algorithm(FeedbackResults) \\ FinalResults \leftarrow Ranking(ExpandedQuery, Data2) \\ PresentResults(FinalResults) \end{cases}$$

Fig. 7. Feedback-Driven Semantic Search

We can compare both Semantic Web data and hypertext documents by considering both to be 'bags of words' and using relevance modelling techniques to expand the queries [17]. We consider both to be 'bags of words.' Semantic Web data can be flattened, and URIs can be reduced to 'words' by the following steps:

– Reduce to the rightmost hierarchical component.
– If the rightmost component contains a fragment identifier (#), consider all characters right of the fragment identifier the rightmost hierarchical component.
– Tokenize the rightmost component on space, capitalization, and underscore.

So, `http://www.example.org/hasArchitect` would be reduced to two tokens, 'has' and 'architect.' Using this system, we evaluated both the vector-space and language models described in Section 4 on queries selected in Section 3.2 with relevance judgments on these queries selected in Section 3.3.

6. Feedback Evaluation

In this section we evaluate algorithms and parameters using relevance feedback against the same system without relevance feedback. In Section 9 we evaluate against deployed systems such as FALCON-S and Yahoo! Web Search. To preview our final results in Section 9, relevance feedback from the Semantic Web shows an impressive 25% gain in average precision over Yahoo! Web Search with a 16% gain in precision over FALCON-S without relevance feedback.

6.1. *Hypertext to Semantic Web Feedback*

6.1.1. *Results*

A number of parameters for our system were evaluated to determine which parameters provide the best results. For each of the parameter combinations, we compared the use of relevance feedback to a baseline system which did not use relevance feedback, yet used the same parameters with the exception of any relevance feedback-related parameters. The baseline system without feedback can also be considered an unsupervised algorithm, while a relevance feedback system can be thought of as a supervised algorithm. For example, the relevant hypertext web-pages R can be considered to be training data, while the Semantic Web documents D we wish to re-rank can be considered to be test data. The hypertext web-pages and Semantic Web documents are disjoint sets ($D \cap R = \emptyset$). For evaluation we used mean average precision (MAP) with the standard Wilcoxon sign-test, which we will often just call 'average precision.'

For vector-space models, the *okapi*, *lca*, and *ponte* relevance weighting functions were all run, each trying both the *inquery* and *cosine* ranking functions. The primary parameter to be varied was the *window size* (m), the number of top frequency words to be used in the vectors for both the query model and the document models. Baselines for both *cosine* and *inquery* were run with no relevance feedback. The parameter m was varied over $5, 10, 20, 50, 100, 300, 1000, 3000$. Mean average precision results are given in Figure 8.

Interestingly enough, *okapi* relevance feedback weighting with a window size of 100 and an *inquery* comparison was the best, with a mean average precision of 0.8914 ($p < .05$). It outperformed the baseline of *inquery*, which has an average precision of 0.5595 ($p < .05$). Overall, *lca* did not perform as well, often performing below the baseline, although its performance increased as the window size increased, reaching an average precision of 0.6262 with

$m = 3000$ ($p < .05$). However, given that a window size of 10,000 covered most documents, increasing the window size will not likely result in better performance from *lca*. The *ponte* relevance feedback performed very well, reaching a maximum MAP 0.8756 with a window size of 300 using *inquery* weighing, and so was insignificantly different from *inquery* ($p > .05$). Lastly, both *ponte* and *okapi* experienced a significant decrease in performance as m was increased, so it appears that the window sizes of 300 and 100 are indeed optimal. Also, as regards comparing baselines, *inquery* outperformed *cosine* ($p < .05$).

For language models, both averaged relevance models *rm* and concatenated relevance models tf were investigated, with the primary parameter being m, the number of non-zero probability words used in the relevance model. The parameter m was varied between 100, 300, 1000, 3000,and 10000. Remember that the query model *is* the relevance model for the language model-based frameworks. As is best practice in relevance modeling, the relevance models were not smoothed, but a number of different smoothing parameters for ϵ were investigated for the cross entropy ranking function, ranging from ϵ between .01, .1, .2, .5, .8, .9, and 0.99. The results are given in Figure 9.

The highest performing language model was tf with a cross-entropy ϵ of .2 and a m of 10,000, which produced an average precision of 0.8611, which was significantly higher than the language model baseline of 0.5043 ($p < .05$) using again an m of 10,000 for document models and with a cross entropy ϵ of .99). Rather interestingly, tf always outperformed *rm*, and *rm*'s best performance had a MAP of 0.7223 using an ϵ of .1 and a m of 10,000.

6.1.2. *Discussion*

Of all parameter combinations, the *okapi* relevance feedback works best in combination with a moderate sized word-window ($m = 100$) and with the *inquery* weighting scheme. It should be noted its performance is identical from a statistical standpoint with *ponte*, but as both relevance feedback components are similar and both use *inquery* comparison and $BM25$ weighing, and not surprisingly the algorithms are very similar. Why would *inquery* and $BM25$ be the best performing? The area of optimizing information retrieval is infamously a black art. In fact, $BM25$ and *inquery* combined present the height of heuristic-driven information retrieval algorithms as explored in Robertson and Spärck Jones [24]. While its performance increase over *lca* is well-known and not surprising, it is interesting that $BM25$ and *inquery* perform significantly better than the language model approach.

The answer is rather subtle. Another observation is in order; note that for vector models, *inquery* always outperformed *cosine*, and that for language models tf always outperformed *rm*. Despite the differing frameworks of vector-space models and language models, both *cosine* and *rm* share the common characteristic of normalization. In essence, both *cosine* and *rm* normalize by documents: *cosine* normalizes term frequencies per vector before comparing vectors, while *rm* constructs a relevance model on a per-relevant document basis before creating the average relevance model. In contrast, *inquery* and tf do not normalize: *inquery* compares weighted term frequencies, and tf constructs a relevance model by combining all the relevance documents and then creating the relevance model from the *raw pool* of all relevant document models.

Thus it appears the answer is that any kind of normalization by length of the document hurts performance. The reason for this is likely because the text automatically extracted from hypertext documents is 'messy,' being of low quality and bursty, with highly varying document lengths. As observed informally earlier [9] and more formally later [12], the amount of triples in Semantic Web documents follow a power-law, so there are wildly varying document lengths of both the relevance model and the document models. Due to these factors, it is unwise to normalize the models, as that will almost certainly dampen the effect of valuable features like crucial keywords (such as 'Paris' and 'tourist' in disambiguating various 'eiffel'-related queries).

Then the reason $BM25$-based vector models in particular perform so well is that, due to its heuristics, it is able to effectively keep track of a term's both document frequency and inverse document frequency accurately. Also, unlike most other algorithms, $BM25$ provides a slight amount of rather unprincipled non-linearity in the importance of the various variables [23]. This is important, as it provides a way of extenuating the effect of one particular parameter (in our case, likely term frequency and inverse term frequency) and then massively lowering the power of another parameter (in our case, likely the document length). While $BM25$ can be outperformed normally by language models [16] in TREC competitions featuring high-quality

samples of English, in the non-normal conditions of comparing natural language and pseudo-natural language terms extracted from structured data in RDF, it is not surprising that *okapi*, whose non-linearity allows certain highly relevant terms to have their frequency 'non-linearly' heightened, provides better results than more principled methods that derive their parameters by regarding the messy RDF and HTML-based corpus as a sample from a general underlying language model.

6.2. *Semantic Web to Hypertext Feedback*

In this section, we assume that the user or agent program has accessed or otherwise examined the Semantic Web documents from the URIs resulting from a Semantic Web search, and these Semantic Web documents then be used as relevance feedback to expand a query for the hypertext Web so that the feedback cycle has been reversed.

6.2.1. *Results*

The results for using Semantic Web documents as relevance feedback for hypertext Web search are surprisingly promising. The same parameters as explored in Section 6.1.1 were again explored. The average precision results for vector-space models are given in Figure 10. The general trends from Section 6.1.1 were similar in this new data-set. In particular, *okapi* with a window size of 100 and the *inquery* comparison function again performed best with an average precision of 0.6423 ($p < .05$). Also *ponte* performed almost the same, again an insignificant difference from *okapi*, producing with the same window size of 100 an average precision of 0.6131 ($p > .05$). Utilizing again a large window of 3,000, *lca* had an average precision of 0.5359 ($p < .05$). Similarly, *inquery* consistently outperformed *cosine* in comparison, with *inquery* having a baseline average precision of 0.4643 ($p < .05$) in comparison with the average precision of *cosine* being 0.3470 ($p < .05$).

The results for language modeling were similar to the results in Section 6.1.1 and are given in Figure 11, although a few differences are worth comment. The best performing language model was tf with a m of 10,000 and a cross entropy smoothing factor ϵ to .5, which produced an average precision of .6549 ($p < .05$). In contrast, the best-performing rm, with a m of 3,000 and ϵ=.5, only had an average precision of 0.4858 ($p < .05$). The tf relevance models consistently performed better than rm relevance models ($p < .05$). The baseline for language modeling was also fairly poor with an average performance of 0.4284 ($p < .05$). This was the 'best' baseline using again an m of 10,000 for document models and cross entropy smoothing ϵ of .99. The general trends from the previous experiment then held, except the smoothing factor was more moderate and the difference between tf and rm was even more pronounced. However, the primary difference worth noting was that best performing tf language model outperformed, if barely, the *okapi* ($BM25$ and *inquery*) vector model by a relatively small but still significant margin of .0126. Statistically, the difference was significant ($p < .05$).

6.2.2. *Discussion*

Why is tf relevance modeling better than $BM25$ and *inquery* vector-space models in using relevance feedback from the Semantic Web to hypertext? The high performance of $BM25$ and *inquery* has already been explained, and that explanation about why document-based normalization leads to worse performance still holds. Yet the rise in performance of tf language models seems odd. However, it makes sense if one considers the nature of the data involved. Recalling previous work [12], there are two distinct conditions that separated this data-set from the more typical natural language samples as encountered in TREC [13]. In the case of using relevant hypertext results as feedback for the Semantic Web, the relevant document model was constructed from a very limited amount of messy hypertext data, which had many text fragments, with a large percentage coming from irrelevant textual data to deal with issues like web-page navigation. However, in using the Semantic Web for relevance feedback, these issues are reversed: the relevant document model is constructed out of relatively pristine Semantic Web documents and compared against noisy hypertext documents.

Rather shockingly, as the Semantic Web is mostly manually high-quality curated data from sources like DBpedia, the actual natural language fragments found on the Semantic Web, such as Wikipedia abstracts, are much better samples of natural language than the natural language samples found in hypertext. Furthermore, the distribution of 'natural' language terms extracted from RDF terms (such as 'sub class of' from `rdfs:subClassOf`), while often irregular, will either be repeated very heavily or fall into the sparse long tail. These two conditions can then

be dealt with by the generative tf relevance models, since the long tail of automatically generated words from RDF will blend into the long tail of natural language terms, and the probabilistic model can properly 'dampen' without resorting to heuristic-driven non-linearities. Therefore, it is on some level not surprising that even hypertext Web search results can be improved by Semantic Web search results, because used in combination with the right relevance feedback parameters, in essence the hypertext search engine is being 'seeded' with high-quality structured and accurate descriptions of the information need of the query to be used for query expansion.

7. Pseudo-feedback

In this section we explore a very easy-to-implement and feasible way to take advantage of relevance feedback without manual selection of relevant results by human users. One of the major problems of relevance feedback-based approaches is their dependence on manual selection of relevant results by human users. For example, in our experiments we used judges manually determining if web-pages were relevant using an experimental set-up that forced them to judge every result as relevant or not, which is not feasible for actual search engine use.

A well-known technique within relevance feedback is *pseudo-feedback*, namely simply assuming that the top x documents returned are relevant. Then, one can use this as a corpus of relevance documents to expand the queries in the same manner using language models as described in Section 4. However, in general pseudo-relevance feedback is a more feasible method, as human intervention is not required.

Using the same optimal parameters as discovered in Section 6.1.1, tf with a $m = 10,000$ and $\epsilon = .2$ was again deployed, but this time using pseudo-feedback. Can pseudo-feedback from hypertext Web search help improve the rankings of Semantic Web data? The answer is clearly positive. Employing all ten results as pseudo-relevance feedback and the same previously optimized parameters, the best pseudo-relevance feedback result had an average precision of 0.6240. This was considerably better than the baseline of just using relevance pseudo-feedback from the Semantic Web to itself, which only had an average precision of 0.5251 ($p < .05$), and also clearly above the 'best' baseline of 0.5043 ($p < .05$). However, as shown by Figure 12, the results are still not nearly as good as using hypertext pages judged relevant by humans, which had an average precision of 0.8611 ($p < .05$). This is likely because, not surprisingly, the hypertext Web results contain many irrelevant text fragments that serve as noise, preventing the relevant feedback from boosting the results.

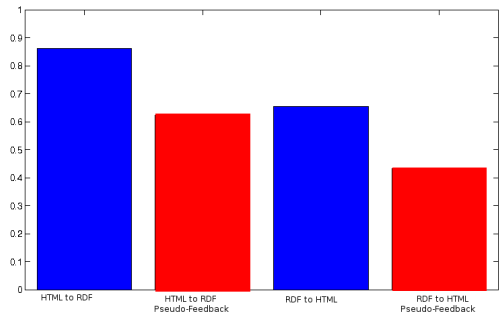

Fig. 12. Comparing Relevance Feedback (red) to Pseudo-Relevance Feedback (blue) on the Semantic Web (RDF) and Hypertext Web (HTML)

Can pseudo-feedback from the Semantic Web improve hypertext search? The answer is yes, but barely. The best result for average precision is 0.4321 ($p < .05$), which is better than the baseline of just using pseudo-feedback from hypertext Web results to to themselves, which has an average precision of 0.3945 ($p < .05$) and the baseline without feedback at all of 0.4284 ($p < .05$). However, the pseudo-feedback results are both still significantly worse performance by a large margin than using Semantic Web documents judged relevant by humans, which had an average precision of 0.6549 ($p < .05$). These results can be explained because, given the usual ambiguous and short one or two word queries, the Semantic Web tends to return structured data spread out of over multiple subjects even moreso than the hypertext Web. Therefore, adding pseudo-relevance feedback increases the amount of noise in the language model as opposed to using actual relevance feedback, hurting performance while still keeping it above baseline.

8. Inference

In this section the effect of inference on relevance feedback is evaluated by considering inference to be document expansion. One of the characteristics of the Semantic Web is that the structure should allow one 'in theory' to discover more relevant data. The

Semantic Web formalizes this in terms of type and sub-class hierarchies in RDF using RDF Schema [5]. While inference routines are quite complicated as regards the various Semantic Web specifications, in practice the vast majority of inference that can be used is on the Semantic Web is of two types (as shown by our survey of Linked Data [12]), *rdf:subClassOf* that indicates a simple sub-class inheritance hierarchy and *rdf:type* that indicates a simple type. For our experiment, we followed all explicit *rdf:subClassOf* statements up one level in the sub-class hierarchy and explicit *rdf:type* links. The resulting retrieved Semantic Web data was all concatenated together, and then concatenated yet again with their source document from the Semantic Web. In this way, Semantic Web inference is considered as *document expansion*.

Inference was first tried using normal relevant feedback, again with the same best-performing parameters (tf with $m = 10,000$ and $\epsilon = .2$). In the first case, the inference is used to expand Semantic Web documents in semantic search, and then the hypertext results are used as relevance feedback to improve the ranking. However, as shown in Figure 13, deploying inference only causes a drop in performance. In particular, using hypertext Web results as relevance feedback to the Semantic Web, the system drops from a performance of 0.8611 to a performance of 0.4991 ($p < .05$). With pseudo-feedback over the top 10 documents, the performance drops even more, from 0.6240 to 0.4557 ($p < .05$). The use of inference actually makes the results worse than the baseline performance of language models of 0.5043 ($p < .05$) without either relevance feedback or inference.

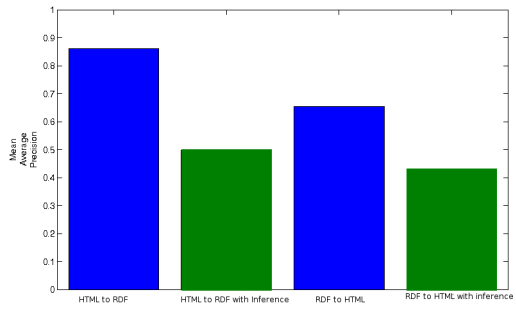

Fig. 13. Comparing the Relevance Feedback on the Semantic Web (RDF) and Hypertext Web (HTML) both without (blue) and with (green) Semantic Web inference

The results of using inference to boost hypertext Web results using Semantic Web equally fail to materialize any performance gains. In this case, inference is used to expand Semantic Web documents, which are then used via relevance feedback to improve the ranking of hypertext search. Using the same parameters as above, the feedback from the expanded Semantic Web data to the hypertext Web results in an average precision of 0.4273, which is insignificantly different from the baseline of not using relevance feedback at all of 0.4284 ($p < .05$) and considerably worse than not using inference at all, which has a MAP of 0.6549 ($p < .05$). When pseudo-feedback is used, the results fall to the rather low score of 0.3861, which is clearly below the baseline of 0.4284 ($p < .05$). So, at least one obvious way of use of simple type and sub-class based Semantic Web inference seems to only lead to a decline in performance.

Why does inference hurt rather than help performance? One would naively assume that adding more knowledge in the form of Semantic Web would help the results. However, this assumes the knowledge gained through inference would somehow lead to the discovery of new relevant terms. However, in the case of much inference with the Semantic Web, this is not the case. For example, simply consider the Semantic Web data about the query for the singer 'Britney Spears.' While the first Semantic Web document about Britney Spears gives a number of useful facts about her, such as the fact that she is a singer, determining that Britney Spears is a person via inference is of vastly less utility. For example, the Cyc ontology [18] declares that Britney Spears is a person, namely that "Something is an instance of Person if it is an individual Intelligent Agent with perceptual sensibility, capable of complex social relationships, and possessing a certain moral sophistication and an intrinsic moral value." In this regard, inference only serves as noise, adding irrelevant terms to the language models. For example, adding 'sophistication' to a query about 'Britney Spears' will likely not help discover relevant documents. Inference would be useful if it produced surprising information or reduced ambiguity. However, it appears that at least for simple RDF Schema vocabularies, information higher in the class hierarchy is usually knowledge that the user of the search engine already possesses (like Britney Spears is a person) and that the reduction of ambiguity is already done by the user in their selection of keywords. However, it is possible that more sophisticated inference techniques are needed, and that inference may help

in specialized domains rather than open-ended Web search. Further experiments in parametrization of inference would be useful given that our exploration in this direction showed no performance increase, only performance decrease.

9. Deployed Systems

In this section we evaluate our system against 'real-world' deployed systems. One area we have not explored is how systems based on relevance feedback perform relative to systems that are actually deployed, as our previous work has always been evaluated against systems and parameters we created specifically for experimental evaluation. Our performance in Section 6.1.1 and Section 6.2.1 was only compared to baselines that were versions of our weighting function without a relevance feedback component. While that particular baseline is principled, the obvious needed comparison is against actual deployed commercial or academic systems where the precise parameters deployed may not be publicly available and so not easily simulated experimentally.

9.1. *Results*

The obvious baseline to choose to test against is the Semantic Web search engine, FALCON-S, from which we derived our original Semantic Web data in the experiment. The decision to use FALCON-S as opposed to any other Semantic Web search engine was based on the fact that FALCON-S returned more relevant results in the top 10 than other existing semantic search engines at the time using a random sample of 20 queries from the set of queries described in Section 3.2. Combined with the explosive growth of Linked Data over the last year and the changes in ranking algorithms of various semantic search engines, it is difficult to judge whether a given Semantic Web search engine is representative of semantic search. However, we would find it reasonable that if our proposed hypothesis works well on FALCON-S, it can be generalized to other Semantic Web search engines.

We used the original ranking of the top 10 results given by FALCON-S to calculate its average precision, 0.6985. We then compared both the best baseline, rm, as well as the best system with feedback in Figure 14. As shown, our system with feedback had significantly ($p < .05$) better average precision (0.8611) than FALCON-S (0.6985), as well better ($p < .05$) than the 'best' language model baseline without feedback (0.5043) as reported earlier as given in Section 6.1.1.

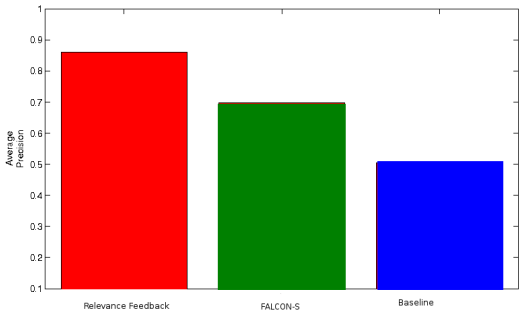

Fig. 14. Summary of Best Average Precision Scores: Relevance Feedback From Hypertext to Semantic Web

Average precision does not have an intuitive interpretation, besides the simple fact that a system with better average precision will in general deliver more accurate results closer to the top. In particular, one scenario we are interested in is having *only* the most relevant RDF data accessible from a single URI returned as the top result, so that this result is easily consumed by some program. For example, given the search 'amnesia nightclub,' a program should be able to consume RDF returned from the Semantic Web to produce with high reliability a single map and opening times for a particular nightclub in Ibiza in the limited screen space of the browser, instead of trying to display structured data for every nightclub called 'amnesia' in the entire world. In Table 3, we show that for a significant minority of URIs (42%), FALCON-S returned a non-relevant Semantic Web URI as the top result. Our feedback system achieves an average precision gain of 16% over FALCON-S. While a 16% gain in average precision may not seem huge, in reality the effect is quite dramatic, in particular as regards boosting relevant URIs to the top rank. So in Table 3, we present results of how our best parameters tf with $m = 10,000$ lead to the most relevant Semantic data in the top result. In particular, notice that now 89% of resolved queries now have relevant data at the top position, as opposed to 58% without feedback. This would result in a noticeable gain in performance for users, which we would argue allows Semantic Web data to be retrieved with high-enough accuracy for actual deployment.

Results:	Feedback	FALCON-S
Top Relevant:	118 (89%)	76 (58%)
Non-Relevant Top:	14 (11%)	56 (42%)
Non-Relevant Top Entity:	9 (64%)	23 (41%)
Non-Relevant Concept:	5 (36%)	33 (59%)

Table 3
Table Comparing Hypertext-based Relevance Feedback and FALCON-S

While performance is boosted for both entities and concepts, the main improvement comes from concept queries. Indeed, as concept queries are often one word and ambiguous, not to mention the case where the name of a concept has been taken over by some company, music band, or product, it should not be surprising that results for concept queries are considerably boosted by relevance feedback. Results for entity queries are also boosted. A quick inspection of the results reveals that the entity queries were the most troublesome, and that these entity queries gave both FALCON-S and our feedback system problems. These problematic queries were mainly very difficult queries where a number of Semantic Web documents all share similar natural language content. An example would be a query for 'sonny and cher,' which results in a number of distinct Semantic Web URIs: one for *Cher*, another one for *Sonny and Cher* the band, and another for 'The Sonny Side of Cher,' an album by Cher. For concepts, one difficult concept was the query 'rock.' Although the system was able to disambiguate the musical sense from the geological sense, there was a large cluster of Semantic Web URIs for rock music, ranging from *Hard Rock* to *Rock Music* to *Alternative Rock*. These types of queries seem to present the most difficulties for Semantic Web search engines.

Although less impressive than the results for using hypertext web-pages for relevance feedback for the Semantic Web, the feedback cycle from the Semantic Web to hypertext does improve significantly the results of even commercial hypertext web-engines, at least for our set of queries about concepts and entities. Given the unlimited API-based access offered by Yahoo! Web Search in comparison to Google and Microsoft web search, we used Yahoo! Web Search for hypertext searching in this experiment, and we expect that the results in a coarse-grained manner should generalize to other Web search engines. The hypertext results for our experiment were given by Yahoo! Web Search, and we calculated a mean average precision for Yahoo! Web Search to be 0.4039. This is slightly less than our baseline language model ranking, which had an average precision of of 0.4284. As shown in Figure 15, given that our feedback based had an average precision of 0.6549, our relevance feedback system performs significantly ($p < .05$) better than Yahoo! Web Search and ($p < .05$) the baseline *rm* system.

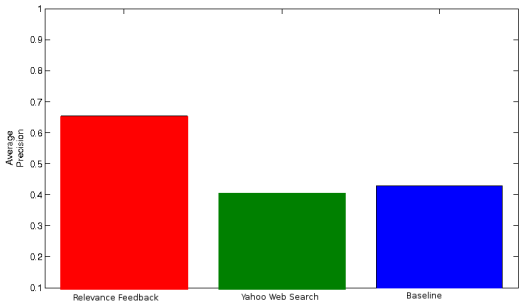

Fig. 15. Summary of Best Average Precision Scores: Relevance Feedback From Semantic Web to Hypertext

9.2. *Discussion*

These results show our relevance feedback method works significantly better than various baselines, both internal baselines and state of the art commercial hypertext search engines and Semantic Web search engines. The parametrization of the precise information retrieval components used in our system is not entirely arbitrary, as argued above in Section 6.1.2 and Section 6.2.2. The gain of our relevance feedback system, a respectable 16% in average precision over the engine FALCON-S, intuitively makes the system's ability to place a relevant structured Semantic Web data in the top rank acceptable for most users.

More surprisingly, by incorporating human relevance judgments of Semantic Web documents, we make substantial gains over state of the art systems for hypertext Web search, a 25% gain in average precision over Yahoo! search. One important factor is the constant assault of hypertext search engines by spammers and others. Given the prevalence of a search engine optimization and spamming industry, it is not surprising that the average precision of even a commercial hypertext engine is not the best, and that it performs less well than Semantic Web search engines. Semantic Web search engines have a much smaller and cleaner world of data to deal with than the unruly hypertext Web, and hypertext Web

search must be very fast and efficient. Even without feedback from the Semantic Web, an average precision of 40% is impressive, although far from the 65% precision using relevance feedback from the Semantic Web.

Interestingly enough, it seemed that pseudo-feedback only helps marginally in improving hypertext Web search using Semantic Web data. Therefore, it is somewhat unrealistic to expect the Semantic Web to instantly improve hypertext Web search. Even with the help of the Semantic Web, hypertext search is unlikely to achieve near perfect results anytime soon. This should not be a surprise, as pseudo-feedback in general performs worse than relevance feedback. However, the loss of performance given by pseudo-feedback in comparison with traditional relevance feedback show that for the Semantic Web using pseudo-feedback for concepts and entities is difficult, as many results that are about highly different things and subject matters may be returned. However, both pseudo-feedback and traditional relevance feedback help a fair amount in improving Semantic Web search using hypertext results, and as relevance judgments can be approximated by click-through logs of hypertext Web search engines, it is realistic and feasible to try to improve semantic search using relevance feedback from hypertext search. In fact, it is simple to implement pseudo-feedback from hypertext Web search using hypertext search engine APIs, as no manual relevance judgments must be made at all and the API simply can produce the top 10 results of any query quickly.

10. Future Work

There are a number of areas where our project needs to be more thoroughly integrated with other approaches and improved. The expected criticism of this work is likely the choice of FALCON-S and Yahoo! Web search as a baseline, and that we should try this methodology over other Semantic Web search engines and hypertext Web search engines. Lastly, currently it is unknown how to combine traditional word-based techniques from information retrieval with structural techniques from the Semantic Web, and while our experiment with using inference as document expansion did not succeed, a more subtle approach may prove fruitful. At this point, we are currently pursuing this in context of creating a standardized evaluation framework for all Semantic search engines. The evaluation framework presented here has led to the first systematic evaluation of Semantic Web search at the Semantic Search 2010 workshop [21]. Yet in our opinion the most exciting work is to be done as regards scaling our approach to work with live large-scale hypertext Web search engines.

While language models, particularly generative models like relevance models [16], should have theoretically higher performance than vector-space models, the reason why large-scale search engines do not in general implement language models for information retrieval is that the computational complexity of calculating distributions over billions of documents does not scale. However, there is reason to believe that relevance models could be scaled to work with Web search if they built their language sample from suitably large 'clean' sample of natural language and also compressed the models by various means.

One of the looming deficits of our system is that for a substantial amount of our queries there are *no* relevant Semantic Web URIs with accessible RDF data. This amount is estimated to be 34% of all queries. However, these queries with no Semantic Web URIs in general *do* have relevant information on the hypertext Web, if not the Semantic Web. The automatic generation of Semantic Web triples from natural language text could be used in combination with our system to create automatically generated Semantic Web data, in response to user queries.

Another issue is how to determine judgments for relevance in a manner that scales to actual search engine use. Manual feedback, while providing the more accurate experimental set-up for testing relevance feedback, does not work in real search scenarios because users do not exhaustively select results based on relevance, but select on a small subset. However, pseudo-feedback does not take advantage of users selecting web-pages, but just assumes the top x are relevant. A better approach would be to consider click-through logs of search engines incomplete approximations of manual relevance feedback [8]. As we only had a small sample of the Microsoft Live Query log, this was unfeasible for our experiments, but would be compelling future work. There is a massive amount of human user click-through data available to commercial hypertext search engines although Semantic Web data has little relevance feedback data itself. While it is easy enough to use query logs to determine relevant hypertext Web data, no such option exists for the Semantic

Web. However, there are possible methodologies for determining the 'relevance' of Semantic Web data, even if machines rather than humans are consuming the data. For example, Semantic Web data that is consumed by applications like maps and calendar programs can be ascertained to be actually relevant.

Finally, while generic Semantic Web inference may not help in answering simple keyword-based queries for entities and concepts, further research needs to be done to determine if inference can help answer complex queries. While in most keyword-based searches the name of the information need is mentioned directly in the query, which in our experiment results from choosing the queries via a named entity recognizer, in complex queries only the type or attributes of the information need are mentioned directly. The name of particular answers is usually unknown. Therefore, some kind of inference may be crucial in determining what entities or concepts match the attributes or type mentioned in the query terms. For example, the SemSearch 2011 competition's 'complex query' task was very difficult for systems that did well on keyword search, and the winning system used a customized crawling of the Wikipedia type hierarchy [21].

11. Conclusion

This study features a number of results that impact the larger field of semantic search. First, it shows a rigorous information retrieval evaluation, the 'Cranfield paradigm', can be applied to semantic search despite the differences between the Semantic Web and hypertext. These differences are well-recorded in our sample of the Semantic Web as taken via FALCON-S using a query log, and reveals a number of large differences between the Semantic Web data and hypertext data, in particular that while relevant data for ordinary open-domain queries does appear on the Semantic Web, Semantic Web data is in general more sparse than hypertext data when given a keyword query from an ordinary user's hypertext Web search. However, when the Semantic Web does contain data relevant to a given query, that data is likely to be accurate information, a fact we exploit in our techniques.

Unlike previous work in semantic search that focuses usually on some form of PageRank or other link-based ranking, we concentrate on using techniques from information retrieval, including language models and vector-space models, over Semantic Web data. Relevance feedback from hypertext Web data can improve Semantic Web search, and even *vice versa*, as we have rigorously and empirically shown. While relevance feedback is known to in general improve results, our use of wildly disparate sources of data such as the structured Semantic Web and the unstructured hypertext Web to serve as relevance feedback for each other is novel. Furthermore as regards relevance feedback, we show using vector-space models over hypertext data is optimal while language models are optimal when operating over Semantic Web. These techniques (as evidenced by the failure of relevance feedback to beat baseline results with incorrect parametrizations) must be parametrized correctly and use the correct weighting and ranking algorithm to be successful. It is shown by our results to be simply false to state that relevance feedback always improves performance over hypertext and Semantic Web search, but only under certain (although easily obtainable) parameters. We do this by treating both data sources as 'bags of words' and links in order to make them compatible and find from the Semantic Web high quality terms for use in language models. Also, untraditionally, we turn the URIs themselves into words. Our results of demonstrate that our approach of using feedback from hypertext Web search helps users discover relevant Semantic Web data. The gain is significant over both baseline systems without feedback and the state of the art page-rank based mechanism used by FALCON-S and Yahoo! Web search. Furthermore, the finding of relevant structured Semantic Web data can even be improved by pseudo-feedback from hypertext search.

More exciting to the majority of users of the Web is the fact that apparently relevance feedback from the Semantic Web can improve hypertext Web. However, pseudo-feedback also improves the quality of results of hypertext Web search engines, albeit to a lesser degree. Interestingly enough, using inference only hurt performance, due to the rather obscure terms from higher-level ontologies serving functionally as 'noise' in the feedback. Lastly, pseudo-feedback from the hypertext Web can help Semantic Web search today and can be easily implemented.

The operative question is: Why does does relevance feedback work? Although there appears to be a huge gulf between the Semantic Web and the hypertext Web, it is precisely because the same *kind of information* is encoded in the unstructured hy-

pertext and the structured Semantic Web that these two disparate sets of data can be used as relevance feedback for each other. Indeed, the key to high performance for search engines is the use of high quality data of any kind for query expansion, whether it is stored in a structured Semantic Web format or the hypertext Web. However, the Semantic Web, by its nature as a source of curated and formalized data, seems to be a better source of high quality data than the hypertext Web itself, albeit with less coverage. While it is trivial to observe that as the Semantic Web grows, semantic search will have more importance, it is even more interesting to demonstrate that as the Semantic Web grows, the Semantic Web can actually improve hypertext search.

References

[1] J. Allan, M. Connell, W. B. Croft, F. F. Feng, D. Fisher, X. Li, INQUERY and TREC-9, in: Proceedings of the Ninth Text REtrieval Conference (TREC-9), 2000.

[2] R. Baeza-Yates, From capturing semantics to semantic search: A virtuous cycle, in: Proceedings of the 5th European Semantic Web Conference, Tenerife, Spain, 2008.

[3] R. A. Baeza-Yates, A. Tiberi, Extracting semantic relations from query logs, in: Proceedings of the Conference on Knowledge Discovery and Data-mining (KDD), 2007.

[4] R. Blanco, H. Halpin, D. Herzig, P. Mika, J. Pound, H. Thompson, T. T. Duc, Repeatable and Reliable Search System Evaluation using Crowd-Sourcing, in: Proceedings of the 34th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, ACM Press, Beijing, China, 2011.

[5] D. Brickley, R. V. Guha, RDF Vocabulary Description Language 1.0: RDF Schema, Recommendation, W3C, http://www.w3.org/TR/rdf-schema/ (Last accessed on Nov. 15th 2008) (2004).

[6] G. Cheng, W. Ge, Y. Qu, FALCONS: Searching and browsing entities on the Semantic Web, in: Proceedings of the the World Wide Web Conference, 2008.

[7] N. Craswell, H. Zaragoza, S. Robertson, Microsoft Cambridge at trec-14: Enterprise track, in: Proceedings of the Seventh Text REtrieval Conference (TREC-7), 2005.

[8] H. Cui, J.-R. Wen, J.-Y. Nie, W.-Y. Ma, Probabilistic query expansion using query logs, in: Proceedings of the 11th International Conference on World Wide Web (WWW 2002), ACM, New York, NY, USA, 2002.

[9] L. Ding, T. Finin, Characterizing the Semantic Web on the Web, in: Proceedings of the International Semantic Web Conference (ISWC), 2006.

[10] J. Fleiss, Measuring nominal scale agreement among many raters, Psychological Bulletin 76 (1971) 378–382.

[11] R. Guha, R. McCool, E. Miller, Semantic search, in: Proceedings of the International Conference on World Wide Web (WWW), ACM, New York, NY, USA, 2003.

[12] H. Halpin, A query-driven characterization of linked data, in: Proceedings of the Linked Data Workshop at the World Wide Web Conference, Madrid, Spain, 2009.

[13] D. Hawking, E. Voorhees, N. Craswell, P. Bailey, Overview of the trec-8 web track, in: Proceedings of the Text REtrieval Conference (TREC), ACM, 2000.

[14] P. Hayes, H. Halpin, In defense of ambiguity, International Journal of Semantic Web and Information Systems 4 (3).

[15] G. Klyne, J. Carroll, Resource description framework (rdf): Concepts and abstract syntax, Recommendation, W3C, http://www.w3.org/TR/rdf-concepts/ (2004).

[16] V. Lavrenko, A Generative Theory of Relevance, Springer-Verlag, Berlin, Germany, 2008.

[17] V. Lavrenko, W. B. Croft, Relevance-based language models, in: Proceedings of the Twenty-Fourth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, ACM Press, New Orleans, Louisiana, USA, 2001.

[18] D. Lenat, Cyc: Towards programs with common sense, Communications of the ACM 8 (33) (1990) 30–49.

[19] C. Mangold, A survey and classification of semantic search approaches, International Journal of Metadata, Semantics, and Ontologies 2 (1) (2007) 23–34.

[20] A. Mikheev, C. Grover, M. Moens, Description of the LTG system used for MUC, in: Seventh Message Understanding Conference: Proceedings of a Conference, 1998.

[21] E. S. E. over Structured Web Data, Roi blanco and harry halpin and daniel herzig and peter mika and jeffrey pound and henry thompson and thanh tran duc, in: Proceedings of the 1st International Workshop on Entity-Oriented Sarch workshop on Entity-Oriented Search (SIGIR 2011), ACM, New York, NY, USA, 2011.

[22] J. M. Ponte, A language modeling approach to information retrieval, Phd dissertation, University of Massachusets (1998).

[23] S. Robertson, H. Zaragoza, M. Taylor, Simple bm25 extension to multiple weighted fields, in: Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM), ACM, Washington, D.C., USA, 2004.

[24] S. E. Robertson, K. Spärck Jones, Relevance weighting of search terms, Journal of the American Society for Information Science 27 (1976) 129–146.

[25] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, Okapi at TREC-3, in: Proceedings of the Third Text REtrieval Conference (TREC-3), 1994.

[26] J. Rocchio, Relevance feedback in information retrieval, in: G. Salton (ed.), The SMART Retrieval System: Experiments in Automatic Document Processing, Prentice-Hall, Inc., Uppder Saddle River, New Jersey, USA, 1971, pp. 313–32.

[27] C. Silverstein, H. Marais, M. Henzinger, M. Moricz, Analysis of a very large web search engine query log, SIGIR Forum 33 (1) (1999) 6–12.

[28] J. Xu, W. B. Croft, Query expansion using local and global document analysis, in: Proceedings of the Nineteenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, Zurich, Switzerland, 1996.

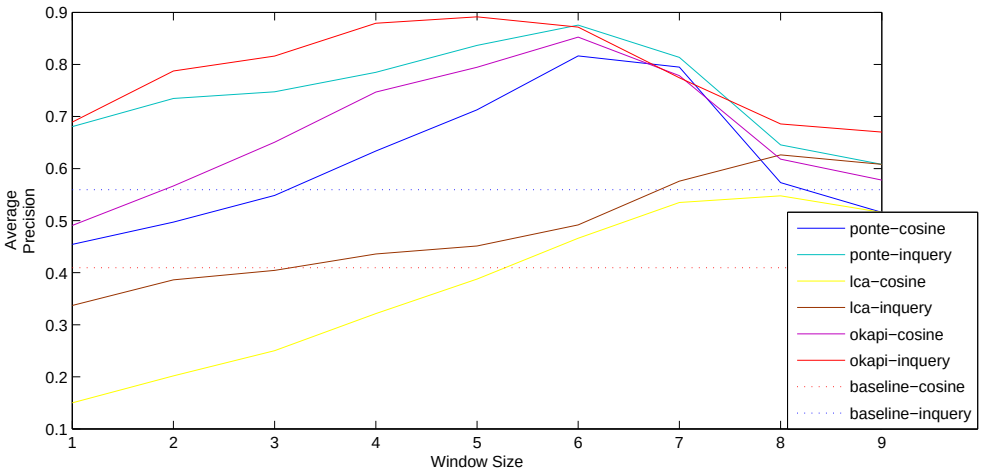

Fig. 8. Average Precision Scores for Vector-space Model Parameters: Relevance Feedback From Hypertext to Semantic Web

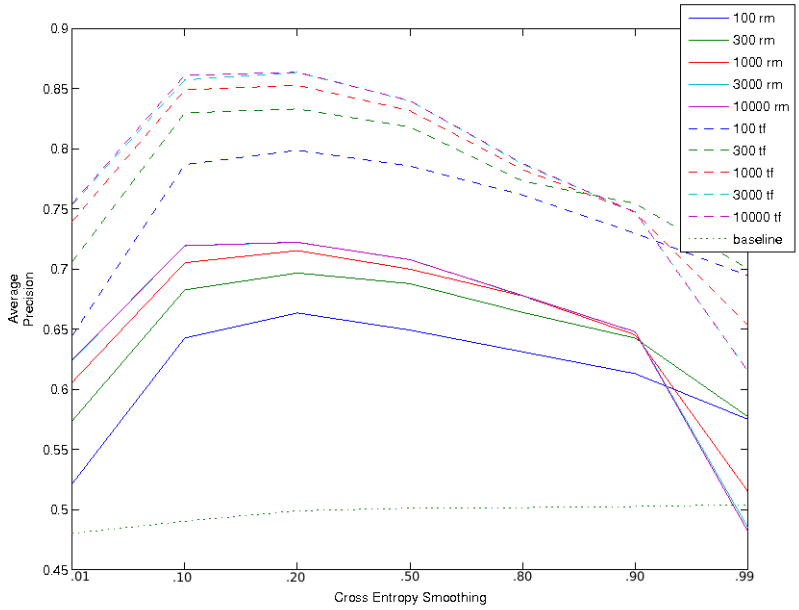

Fig. 9. Average Precision Scores for Language Model Parameters: Relevance Feedback From Hypertext to Semantic Web

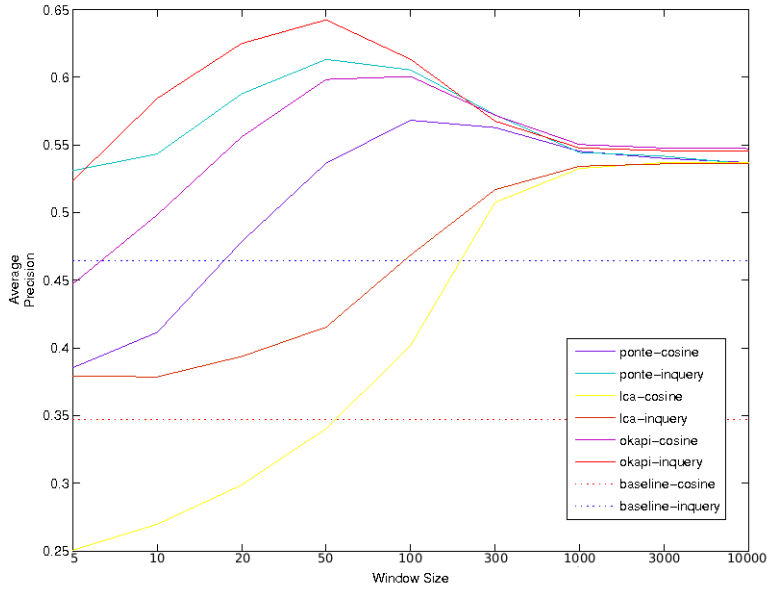

Fig. 10. Average Precision Scores for Vector-space Model Parameters: Relevance Feedback From Semantic Web to Hypertext

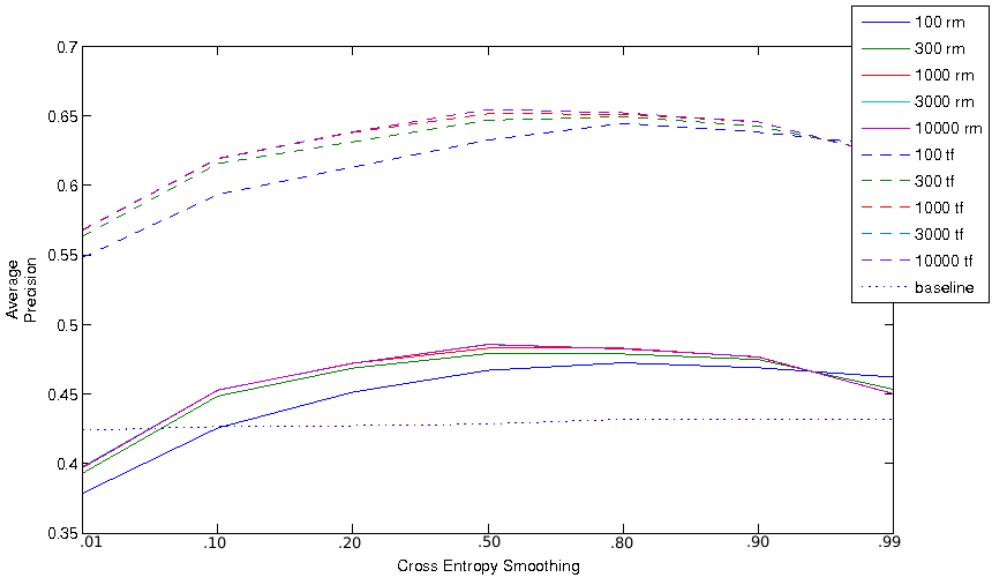

Fig. 11. Average Precision Scores for Language Model Parameters: Relevance Feedback From Hypertext to Semantic Web