Relevance Feedback Between Hypertext and Semantic Search

Harry Halpin
Institute for Communicating and Collaborative Systems
University of Edinburgh
10 Crichton St.
Edinburgh, United Kingdom
H.Halpin@ed.ac.uk

Victor Lavrenko
Institute for Communicating and Collaborative Systems
University of Edinburgh
10 Crichton St.
Edinburgh, United Kingdom
vlavrenk@inf.ed.ac.uk

ABSTRACT

Relevance feedback is one method for creating a 'virtuous cycle' - as put by Baeza-Yates - between semantics and search. Previous approaches to search have generally considered the Semantic Web and hypertext Web search to be entirely disparate, indexing and searching over different domains. While relevance feedback have traditionally improved information retrieval performance, relevance feedback is normally used to improve rankings of a single data-set. Our novel approach is to use relevance feedback from hypertext Web search to improve the retrieval of Semantic Web data. We also inspect whether relevance feedback from Semantic Web data can improve hypertext Web search results. In both cases, an evaluation based on certain kinds of informational queries (abstract concepts, people, and places) selected from a query log and human judges show that relevance feedback works: relevance feedback from hypertext Web search can improve the retrieval of Semantic Web data, and vice versa. We evaluate our work over a wide range of algorithms, and show it improves baseline performance on these queries for deployed systems as well, such as the Semantic Search engine FALCON-S and the commercial Web search engine Yahoo! search.

1. INTRODUCTION

There has recently been a return of interest in 'Semantic Search.' In particular, this seems inspired mostly by the Linked Data initiative, that has released a massive amount of structured data on the Web from a diverse range of sources, leading to the rise of specialized Semantic Web search engines and more interest in the possibilities of combining structured data and ad-hoc information retrieval from traditional hypertext search. The hypothesis put forward by Baeza-Yates is that the search for structured data - called 'Semantic Search' - can be used to improve traditional ad-hoc information retrieval for Web search engines [2], and that techniques from information retrieval can be used to improve structured data, particularly on the Semantic Web. While this is restrictive, we will assume from hereon that 'Semantic Search' refers to indexing and retrieving Semantic Web data, as given by the Linked Data Web and as done by engines like Sindice and FALCON-S, and hypertext search refers to the indexing and retrieval of hypertext documents on the World Wide Web as done by search engines like Google and Yahoo search. We also assume a traditional, ad-hoc information retrieval system for both kinds of search.

We are the first to suggest that relevance feedback may be the primary method for creating a 'virtuous cycle' between semantics and search. Previous approaches like Microsearch and Semantic Search engines like FALCON-S have assumed that the Semantic Web - at least as represented by the native RDF triples in Linked Data - and the hypertext Web search to be entirely disparate, indexing and searching them differently [4], although moves by Semantic search engines like Sindice to index microformat and RDFa is blurring this distinction [11]. Relevance feedback usually improves information retrieval performance, but almost always the feedback is used to improve rankings over a single source of data. Our novel approach is to use relevance feedback from hypertext Web search, of which there is a massive amount of data available, to improve the retrieval of Semantic Web data. We focus on retrieving relevant data in the first position, as one problem exhibited by Semantic Web search engines like FALCON-S and even Microsearch is the retrieval of far too much Semantic Web data, with much of it being irrelevant. We imagine that what interests application developers the most would be selecting a small amount of high quality Semantic Web data, which would have a high assurance of being 'about the same thing' as the query. This data could be automatically consumed by applications like maps and calendar programs, or displayed in some special format by the result page of the search engine.

WWW2009, April 20-24, 2009, Madrid, Spain.

2. SYSTEM DESIGN

In order to deal with these problems, we will employ ***relevance feedback***, the *use of explicit relevance judgements 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 the known relevant documents. For example, the selection of a document by a user and their staying on the document for some period of time is a sign of relevance. 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 [18]. This has been shown in general to improve retrieval performance significantly, both in early studies and in later work like relevance modelling that creates relevance directly from the indexed documents rather than explicitly waiting for the user to make a relevance judgement [8].

Semantic search engines exist, but their rankings are known to be sub-optimal. This is even true for hypertext search engines to some extent. So, our novel solution is to use selected hypertext webpages as relevance feedback for improving the ranking of Semantic Web data. In our solution, we run the query against the hypertext Web search engine first and collect relevance judgements from

this. We then use these judgments to expand the query with highly-weighted words from the relevant documents. The expanded query is used to re-rank the results retrieved by a Semantic Web search engine specializing in indexing Linked Data in RDF. We can compare both Semantic Web data and hypertext data by considering 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 last rightmost hierarchical component.
- If URI contains a fragment identifier (#), consider all characters right of the fragment the last most hierarchical component.
- Remove non-rightmost hierarchical component.
- Tokenise on space, capitalization, and underscore.

So, the URI `http://www.example.org/hasArchitect` would be reduced to two tokens, 'has' and 'architect.' We can also then run the process backwards, using selected Semantic Web data as relevance feedback to improve hypertext Web search. This is not unfeasible, as one could consider the 'consumption' of Semantic Web data by a program to be a judgement of relevance.

3. SELECTING QUERIES

In order to select real queries from users in order to test our hypothesis, 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 capitalisation. 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 recogniser was employed to discover the names of people and places [10], 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 query log total of 509,659 queries were identified as either people or places by the named-entity recogniser, and we call these queries *entity queries*. Employing WordNet to represent abstract concepts, we chose queries recognised by WordNet that are *both* a hyponym and hypernym. This resulted in a more restricted 16,698 queries that are supposed to be about abstract concept, which we call *concept queries*. In order to select a subset of informational queries for evaluation, we selected 100 queries identified as abstract concepts by WordNet and then 100 queries identified as either people or places by the named entity recogniser, for a total of 200 queries needed for evaluation. Constraints were placed on crawled URIs, such that at least 10 Semantic Web documents were crawled for each query, 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 experimental results. Then, the same experimental query log was used to crawl the hypertext Web, resulting in a total of 1,000 web-pages about entities and 1,000 web-pages about concepts. The web-pages were retrieved using Yahoo! Search. A random selection of ten queries from the concept queries is given in Table 3 and another random selection of ten entity queries is given in Table 3. As one can tell, the queries about entities and concepts are spread across quite diverse domains, ranging from entities over locations (El Salvador) and people (both fictional such as Harry Potter and non-fictional such as Earl May) and for concepts over a whole range of abstraction, from sociology to ale.

1	ashville north carolina
2	harry potter
3	orlando florida
4	ellis college
5	university of phoenix
6	keith urban
7	carolina
8	el salvador
9	san antonio
10	earl may

Table 1: 10 Selected Entity Queries

131	sociology
133	clutch
134	telephone
135	ale
136	pillar
137	sequoia
138	aster
139	bedroom
140	tent
141	cinch

Table 2: 10 Selected Concept Queries

3.1 Relevance Judgements

For each of the 200 experimental queries, 10 hypertext web-pages and 10 Semantic Web documents need to be judged for relevance, leading to a total of 4,000 human judgements for relevance in total for our entire experiment. The human judges each judged 25 queries presented in a randomized order, and were given a total of 3 hours to test 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 Web documents. So for each query, the judge determines relevance for 20 results, first 10 web-pages and then 10 Semantic Web documents, leading to a total of 20 judgements per query per judge. Each result therefore was judged by three judges, with a total of 30 judges used in the entire experiment. So over a single session, the judges gave judgements to 20 distinct results. The judges were given instructions and trained on 10 sample results (5 web-pages and 5 Semantic Web results). The human judges are forced to make binary judgments of relevance, so each result must be either relevant or irrelevant to the query.In their instructions, relevance was defined *as whether or not a result is about the same thing as the query, which can be determined if accurate information about the information need is expressed by the result*.

To expand, a number of types 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 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 standardised 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 judgement results 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, any associated text in the subject, and any associated Semantic Web document) by using the open-source Disco Hyperdata Browser with Firefox.[1] In both cases, the resulting rendering of the Web representation was saved at 469 $\times$ 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 given the same experience in the experiment and to be presented with the web-page in its intended visual form. One side-effect of this is that web-pages that heavily depended on non-standardised technologies or plug-ins would not render and were thus presented as blank screen shots to the user. The user-interface broke 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 1. 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 Step 1 as shown in Figure 2, 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 Step 1, the judge had to determine whether or not the Semantic Web results are relevant.

After the ratings were completed, Fliess's κ correct statistic was taken in-order to test the reliability of inter-judge agreement over the relevancy ranking [6]. Simple percentage agreement is not sufficient, as it does not take into account the likelihood of purely coincidental agreement by the judges. Fleiss's κ both corrects for chance agreement and can be used for more than two judges [6].The null hypothesis is that the judges cannot distinguish relevant from irrelevant results, and so are judging results randomly. Overall, for both relevance judgements over Semantic Web results and web-page results, $\kappa = 0.5724$ ($p < .05$, 95% Confidence interval $[0.5678, 0.5771]$), indicating the rejection of the null hypothesis

[1]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/`.

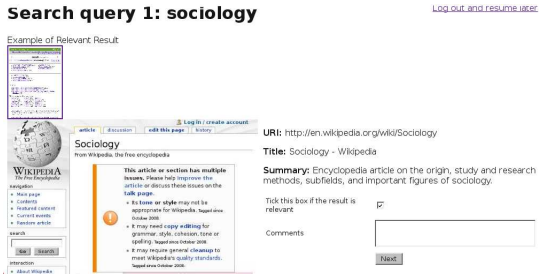

Figure 1: The interface used to judge web-page results for relevancy.

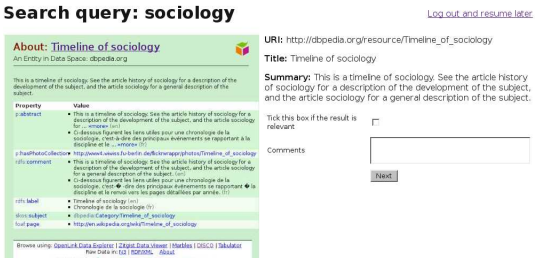

Figure 2: The interface used to judge Semantic Web results for relevancy

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 $[0.5859, 0.5991]$), is 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 judgements 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 et al. that there is more agreement in general about 'less' abstract things like people and places rather than abstract concepts [7]. However, agreement is still very similar and moderate for both information about entities and concepts.

For the queries, much of the data is summarised in Table 3.1. '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 the hypertext and Semantic Web relevancy corpora in order to allow direct comparison. The percentages for 'Top Relevant' and 'Non-Top Relevant' were computed as percentages over all relevant queries, and so ex-

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

Table 3: Results of Hypertext and Semantic Web Relevance Judgements

cludes unresolved queries. For ease of reference, a pie-chart for the hypertext relevancy is given in Figure 3 and for the Semantic Web relevancy in Figure 4.

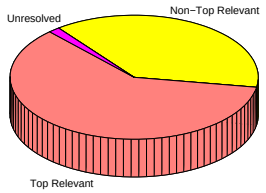

Figure 3: Results of Querying the Hypertext Web.

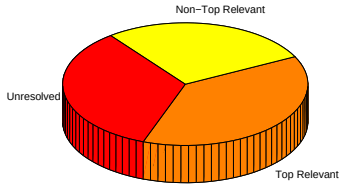

Figure 4: Results of Querying the Semantic Web.

Resolved queries are *queries that return at least one relevant result* in the top 10 results, while ***unresolved*** are *queries that return no relevant queries in the top 10 results*. Over 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 results in the top 10, but that for the Semantic Web almost a third of all queries did not return any relevant result. This only means there is much information that does not yet have a relevant form on the Semantic Web, unless it is hidden by poor-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. However, while the hypertext Web search had 121 (61%) top-ranked relevant results, for the Semantic Web there was only had 76 (58%) top-ranked results. A lack of top-ranked results becomes particularly acute on the Semantic Web for queries about concepts. What is more compelling for relevance feedback is the number of relevant results that were *not* the top-ranked result. Again over both 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 non-top relevant result, while for the Semantic Web, there were 56 (42%) of all queries that had a non-top relevant result. So, while queries on the Semantic Web are more likely to turn up no relevant results, when a relevant query is returned, both for the hypertext Web and the Semantic Web it is quite likely that a relevant result will be in the non-top position of the result list.

4. INFORMATION RETRIEVAL FRAMEWORK

In our experiment we tested two general kinds of information retrieval frameworks: vector-space models and language models. In the *vector-space model*, where document models are considered to be vectors of terms (usually called *words* as they are usually, although not exclusively, from natural language) where the weighing function and query expansion has no principled basis besides empirical results, although ranking is done via 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. However, 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 generative probabilistic approach to determining the ranking and weighting function. Instead of each document being considered some parametrised 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 modelling efforts [13], the probability that the language model of a document would generate the query was the ranking function of the document. A more sophisticated approach to language model 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, so the it 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 had 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 (D), the representation of a single document. Words with a zero frequency are excluded from the document model, and the query is given by Q.

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 [15] and a descendant of the *tf.idf* weighting scheme pioneered by Spärck Jones and Robertson [14]. 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 [5] competitions. 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 Comparison Function: Cosine and InQuery

The vector-space models have an intuitive comparison function in the form of cosine measurements. In particular, the cosine comparison function is given by Equation 2, for a document D with query Q, where both D and Q contain i words, iterating over all i 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 normalised 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 normalises the vector lengths and then proceeds to weight both the query terms and the vector terms by $BM25$. The version without normalisation is called $inquery$ after the *InQuery* system [1]. The $inquery$ comparison function is the same as $cosine$ except without normalisation each word in the query can be considered to have uniform weighing.

4.1.4 Relevance: 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* [18] and at one point used by the *Okapi* system [16], 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 $okapi$:

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

Another state of the art query expansion technique is known as *Local Content Analysis* (lca) [19]. 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 $D \in R$.

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

After each word $w \in V$ has been ranked by lca, then the query given 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 the result set (idf_w).

The final method we will use is the heuristic method developed by Ponte [12], 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.

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

4.2 Language Models

4.2.1 Representation

Language modelling 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. $P(w|u_D(v))$ [13]. The document model distribution $u_D(v)$ is then estimated using the parameter λ_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.

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

The parameter λ_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:

$$P(Q|D) = \prod_{qinQ} u_D(Q) \quad (7)$$

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:

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

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:

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

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 Comparison 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 [9] 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:

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

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 [9].

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

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. FEEDBACK EVALUATION

5.1 Hypertext to Semantic Web Feedback

5.1.1 Results

A number of parameters were tested for our system 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 algorithms, while the relevance-feedback systems supervised algorithm. Note that in machine-learning terms, the hypertext web-pages R can be considered to be training data, while the Semantic Web data D can be considered to be test data. The hypertext web-pages and Semantic Web data are disjoint sets ($D \cap R = \emptyset$). For evaluation we used mean average precision, with the standard Wilcoxian sign-test and mean average precision.

For vector-space models, the $okapi$, lca, and $ponte$ relevance weighting functions were all run, each trying both the $inquery$ and $cosine$ comparison functions. The primary parameter to be varied was the *window size* (m) top frequency non-zero 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 $5, 10, 20, 50, 100, 300, 1000, 3000$. The results in terms of mean average precision are given in Figure 5.

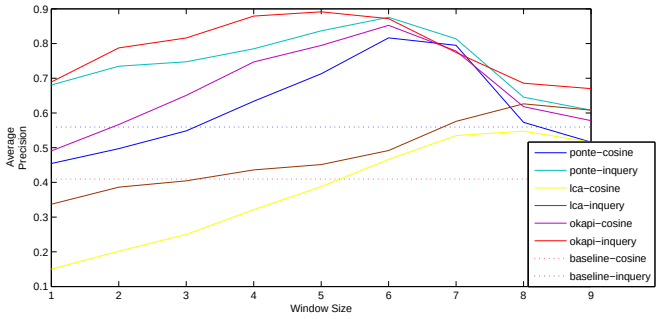

Figure 5: Average Precision Scores for Vector-space Model Parameters: Relevance Feedback From Hypertext to Semantic Web

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 pattern 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 modelling, the relevance models were not smoothed, but a number of different smoothing parameters for ϵ were investigated for the cross entropy comparison function, ranging from ϵ between 0.01,0.10,0.20, 0.50,0.80,0.90, and 0.99. The results are given in Figure 6.

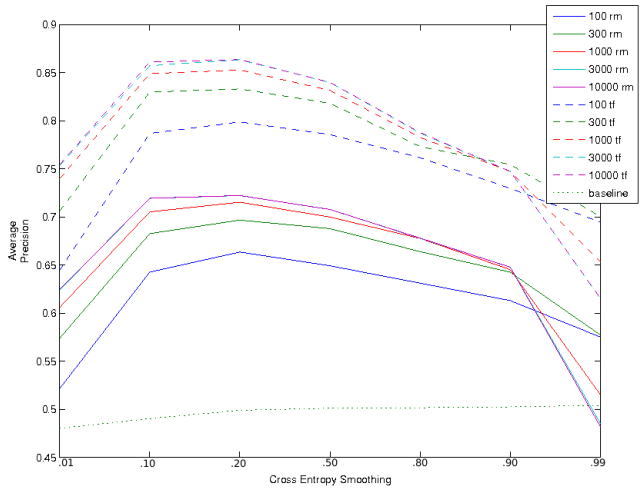

Figure 6: Average Precision Scores for Language Model Parameters: Relevance Feedback From Hypertext to Semantic Web

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.

5.1.2 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, the algorithms are very similar. One 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 normalisation. In essence, both $cosine$ and rm normalise by documents: $cosine$ normalises 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 normalise: $inquery$ 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 normalisation 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 earlier [11], 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 normalise 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 they are able to effectively keep track of both term frequency and inverse term frequency accurately. Unlike most other algorithms, $BM25$ provides a slight amount of rather unprincipled non-linearity in the importance of the various variables [17]. 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).

5.2 Semantic Web to Hypertext Feedback

In this section, we assume that the user or agent program has somehow accessed or otherwise examined the associated descriptions from the Semantic Web URIs, and these associated descriptions then form a relevant document model that is compared to hypertext documents in order to produce rankings. In this way, the feedback cycle has been reversed.

5.2.1 Results

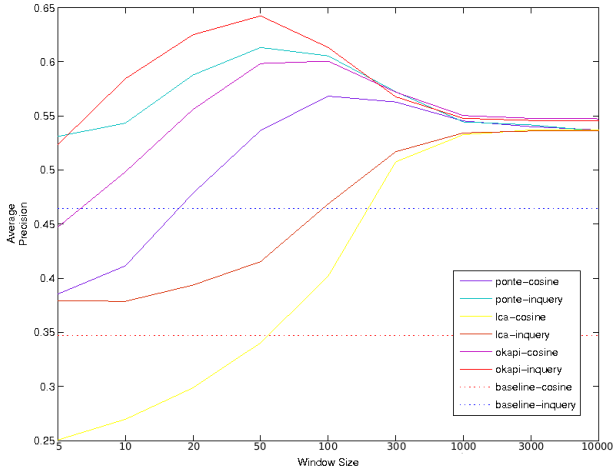

Figure 7: Average Precision Scores for Vector-space Model Parameters: Relevance Feedback From Semantic Web to Hypertext

The results for using Semantic Web documents as relevance feed-

back for hypertext Web search are surprisingly promising. The same parameters as explored in Section 5.1.1 were again explored. The average precision results for vector-space models are given in Figure 7. The general trends from Section 5.1.1 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 ($p > .05$) . Utilising 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 modelling were similar to the results in Section 5.1.1 and are given in Figure 8, 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 modelling 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 ($p < .05$).

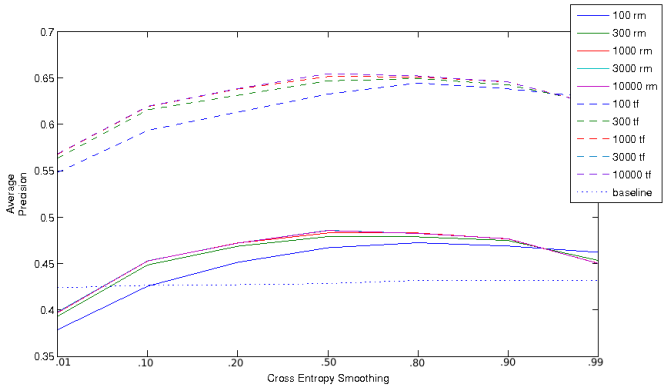

Figure 8: Average Precision Scores for Language Model Parameters: Relevance Feedback From Hypertext to Semantic Web

5.2.2 *Discussion*

Why is tf relevance modelling better than $BM25$ and $inquery$ vector-space models in using relevance feedback from the Semantic Web to hypertext? Rather shockingly, as the Semantic Web data is mostly manually high-quality curated data from sources like DBPedia, the actual natural language fragments on the Semantic Web, found for example in 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 data, 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 referent of the query to be used for query expansion.

5.3 Relevance Feedback against Deployed Systems

One area we have not explored is how our system performs against systems that are actually deployed, as our previous work has all evaluated against systems and parameters we created specifically for experimental evaluation. For example, our performance in Section 5.1.1 and Section 5.2.1 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. The obvious baselines to choose to test against the Semantic Web search engine, FALCON-S, from which we derived our original Semantic Web data in the experiment. 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, $inquery$, as well as the best ($okapi$ with $inquery$ and $m = 100$) feedback based system in Figure 9. As shown, our feedback based system had significantly ($p < .05$) better average precision (0.8914) than both FALCON-S (0.6985) and the baseline without feedback ($p < .05$).

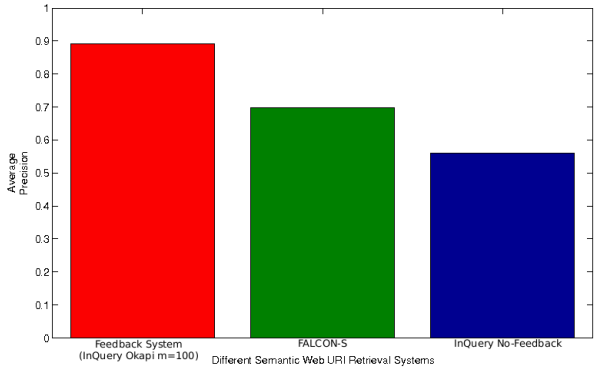

Figure 9: Summary of Best Average Precision Scores: Relevance Feedback From Hypertext to Semantic Web

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, we are interested in having *only* the most relevant RDF data accessible from a single URI returned for the *top result*. The experiment, in Table 3.1, shows that for a significant minority of URIs

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

Table 4: Table Comparing Hypertext-based Relevance Feedback and FALCON-S

(42%), FALCON-S returned a non-relevant Semantic Web URI as the top result. Our feedback system achieves an average precision gain of 20% over FALCON-S. While a 20% gain in average precision may not seem huge, in reality the effect is quite dramatic, particularly as regards boosting relevant Semantic URIs to the top rank of the results. In Table 4, we present clear results of how our best parameters $okapi - inquerym = 100$ 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%. 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.

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 often 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, and are now the most difficult kind of URI for our system to disambiguate. A quick inspection of the results reveals that the entity queries that gave both FALCON-S and our feedback system problems were mainly very difficult queries which has a number of Semantic Web URIs that all share similar natural language content in their associated descriptions. 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*. With a large cluster of very similar Semantic Web data, it is not surprising that both our system and FALCON-S had difficulty with certain queries. ??????????????

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 improves significantly the results of even commercial hypertext web-engines, at least for our set of queries about concepts and entities. 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 $inquery$ ranking, which had an average precision of of 0.4643. As shown in Figure 10, our feedback based system performs significantly ($p < .05$) better than Yahoo! Web Search and ($p < .05$) the baseline $inquery$ system.

5.3.1 Discussion

These results are not in need of a large discussion, as they clearly 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 parametrisation of the precise information retrieval components used in our system is not entirely arbitrary, as argued above in Section 5.1.2 and Section 5.2.2. The gain of our relevance feedback system, a respectable 19% in average precision over the engine FALCON-S, intuitively makes the ability of our system to place the correct URI in response to a query acceptable for most users. The most difficult step is to select the precise correct Semantic Web data for the user's need, and in this regard, even small differences can make a huge impact, so a move to 89% average precision for a given natural language query makes a large difference.

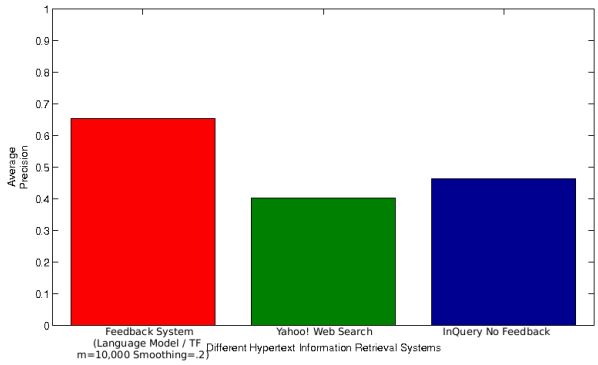

Figure 10: Summary of Best Average Precision Scores: Relevance Feedback From Semantic Web to Hypertext

Second, by incorporating human relevance from the Semantic Web, we make substantial gains over state of the art baseline systems for hypertext Web search. One important factor is the constant assault of hypertext search engines by spammers and others. Given the prevalence of a search engine optimisation 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 by a mean average precision of 29% 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. Thus, even with relevance from the Semantic Web, an average precision of 69% is impressive, although far from the almost oracle-like powers of 89% precision. Improving hypertext Web search is difficult even with relevance feedback. Even with the help of relevance feedback from the Semantic Web, hypertext search is unlikely to achieve near-perfect results anytime soon.

6. FUTURE WORK

There are a number of areas where our project needs to be more thoroughly integrated with other approaches and improved. The primary expected criticism of this work is likely the choice of FALCON-S and Yahoo Web search as a baseline, and that we should try over other Semantic Search engines and hypertext Web search engines, and we leave this for future work. However, there are also much to be done as regards scaling and queries with no relevant results.

6.0.2 Scaling to the Web

While language models, particularly generative models as given by [9], should in general have theoretically higher performance than vector-space models, we showed a slight but significantly better performance for vector-space than language models in relevance feedback from hypertext web-pages to the Semantic Web, likely due to the parameters of the language model being generated by the

infamously messy and non-parametric natural language data of the Web. Furthermore, 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 in general and Semantic Web search in particular if they built their language sample from a 'clean' and suitably large sample of natural language (as was done in our relevance-feedback experiment using relevant Semantic Web results) then these relevance models would be more effective. The computational complexity could be reduced via caching and the use of Bloom filters for the language model. This, combined with some sort of statistical query expansion that would help a user resolve ambiguous queries like `rock` into `rock music` or `geological rock`, would likely get our performance to about 89%. Further natural language processing, including better stemming and lemmatization, would also likely improve performance. Lastly, our system and experiment was only a *proof of concept* system, and it was tested only over a relatively small (although statistically significant) number of users and queries automatically harvested from a query engine. Far better would be to deploy this system with a global-scale hypertext search engine.

6.0.3 Creation of New Semantic Web data

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. This amount is estimated to be 34% of all queries, almost as many as there were queries where non-relevant Semantic data was the first result. However, these queries with no Semantic Web URIs in general *do* have relevant information on the hypertext Web, if not the Semantic Web. In this manner, the automatic generation of Semantic Web triples from natural language text as explored by Brewster et al. [3] could be used in combination with our system to create new URIs, with accessible and automatically generated Semantic Web data, in response to user queries.

7. CONCLUSION

These preliminary results of our experiment 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. These results, due to the significant and randomised number of queries used and the fact that relevance judgements involved three judges, point to a high reliability for these results, so we have reason to be believe the results will scale. Also, the reverse works as well: Assuming that applications are actually using structured data like the data on the Semantic Web, this can help serve as query expansion data for queries on the hypertext Web. The operative question is: Why does this work? It is precisely because the same *kind of information* is encoded in hypertext and the Semantic Web results, these two disparate sets of data can be used as relevance feedback for each other, beginning the first step in literally creating a "virtuous cycle" between semantics and search, a cycle driven by relevance feedback from users [2].

8. REFERENCES

[1] J. Allan, M. Connell, W. B. Croft, F. F. Feng, D. Fisher, and X. Li. INQUERY and TREC-9. In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, pages 551–562, 2000.

[2] Ricardo Baeza-Yates. From capturing semantics to semantic search: A virtuous cycle. In *Proceedings of the 5th European Semantic Web Conference*, pages 1–2, Tenerife, Spain, 2008.

[3] Christopher Brewster, Jose Iria, Ziqi Zhang, Fabio Ciravegna, Louise Guthrie, and Yorick Wilks. Dynamic iterative ontology learning. In *Proceedings of the Recent Advances in Natural Language Processing Conference (RANLP)*, Borovets, Bulgaria, 2007.

[4] Gong Cheng, Weiyi Ge, and Yuzhong Qu. FALCONS: Searching and browsing entities on the semantic web. In *Proceedings of the the World Wide Web Conference*, 2008.

[5] Nick Craswell, Hugo Zaragoza, and Stephen Robertson. Microsoft cambridge at trec-14: Enterprise track. In *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, page http://research.microsoft.com/apps/pubs/default.aspx?id=65241 (Last accessed January 10th 2009, 2005.

[6] J.L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76:378–382, 1971.

[7] Patrick Hayes and Harry Halpin. In defense of ambiguity. *International Journal of Semantic Web and Information Systems*, 4(3), 2008.

[8] V. Lavrenko, J. Allan, E. DeGuzman, D. LaFlamme, V. Pollard, and S. Thomas. Relevance models for topic detection and tracking. In *Proceedings of Human Language Technologies Conference, HLT 2002*, pages 104–110, 2002.

[9] Victor Lavrenko. *A Generative Theory of Relevance*. Springer-Verlag, Berlin, Germany, 2008.

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

[11] Eyal Oren, Renaud Delbru, Michele Catasta, Richard Cyganiak, Holger Stenzhorn, and Giovanni Tummarello. Sindice.com: a document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics, and Ontologies 2008*, 3(1):37–52, 2008.

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

[13] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the Twenty-First Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, Melbourne, Australia, 1998. ACM Press.

[14] S. E. Robertson and K. Spärck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146, 1976.

[15] S. E. Robertson, S. Walker, and M. M. Beaulieu. Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive track. In *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, pages 253–264, 1998.

[16] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 109–126, 1994.

[17] Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Simple bm25 extension to multiple weighted fields. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 42–49, Washington, D.C., USA, 2004. ACM.

[18] J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–32. Prentice-Hall, Inc., Uppder Saddle River, New Jersey, USA, 1971.

[19] J. Xu and 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*, pages 4–11, Zurich, Switzerland, 1996.