Information Retrieval Support for Ontology Construction and Use

Willem Robert van Hage[⋆], Maarten de Rijke, and Maarten Marx

Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam
wrvhage,mdr,marx@science.uva.nl

Abstract Information retrieval can contribute towards the construction of ontologies and the effective usage of ontologies. We use collocation-based keyword extraction to suggest new concepts, and study the generation of hyperlinks to automate the population of ontologies with instances. We evaluate our methods within the setting of digital library project, using information retrieval evaluation methodology. Within the same setting we study retrieval methods that complement the navigational support offered by the semantic relations in most ontologies to help users explore the ontology.

1 Introduction

Today's prevalent methods of searching, navigating, and organizing Internet information builds on decades of research in information retrieval (IR) [2]. Based on statistical laws governing human language use, these methods are being used not only in document retrieval but also in semantically richer tasks such as question answering [29]. One vision of the Semantic Web is that it will be much like the Web as we know it today, except that documents will be enriched with machine understandable markup [19]. These annotations will provide metadata about the documents and machine interpretable statements capturing the semantics of documents' content [7]. We discuss how the IR paradigm can contribute to this effort, by aiding the architects of non-trivial ontologies. IR techniques can aid in defining, populating, and checking the consistency of ontologies. Specifically, eight stages can be distinguished in the ontology building process [1]:

1. Determine the scope of the ontology.
2. Consider reusing (parts of) existing ontologies.
3. Enumerate all the concepts you want to include.
4. Define the taxonomy of these concepts.
5. Define properties of the concepts.
6. Define facets of the concepts such as cardinality, required values etc.
7. Define instances.
8. Check the consistency of the ontology.

[⋆] Current affiliation: TNO TPD and the Free University Amsterdam.

Of these stages, we address 3 and 7 with IR-based techniques as we believe that these stages can be usefully tackled using retrieval technology available today. While equally suitable for automation, stage 4 is far from being a solved problem [22], and stage 8 is best left for purely symbolic reasoning methods as implemented in, e.g., the FACT and RACER provers.

In addition to being used to assist ontology builders, IR techniques can also assist users in searching, browsing, and providing serendipity. People will want to use the Semantic Web to search not just for documents, but also for information about specific semantic relationships, for instance in the setting of digital libraries [25]. Thus, we explore approaches to "retrieval within a concept hierarchy," where exact-match search as provided by most navigation tools and ontology editors may not be adequate [26].

By making today's document retrieval algoritms useful to building and exploiting the Semantic Web infrastructure, improvements in the former lead directly to improvements in the latter. But we have a more methodological reason for bringing IR and Semantic Web efforts closer together. The IR community has long emphasized the importance of evaluation. With the advent of the Text REtrieval Conferences (TREC, [27]), experimental evaluation of retrieval related tasks received a significant boost, which led to rapid progress in the tasks evaluated. Similar benefits occur with other retrieval-related evaluation exercises (CLEF [5], INEX [10], NTCIR [21]), and with efforts to evaluate semantically richer language processing tasks (e.g., CoNLL [6] and Senseval [24]). The Semantic Web community would benefit from a stronger emphasis on evaluation, and on tasks that can be evaluated, than it has so far had. Eating our own dog food, we conduct experimental evaluations for all tasks addressed in this paper.

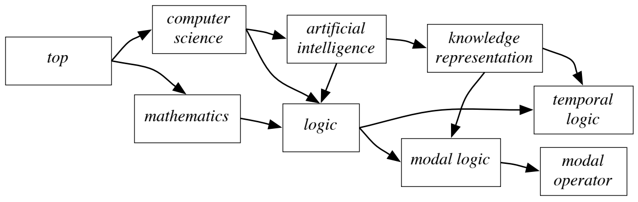
Section 2 discusses the setting in which our work takes place: the Logic and Language Links (*LoLaLi*) project, aimed at providing ontology-based access to an electronic handbook on the interface of linguistics and logic. In Section 3 we discuss the automation of stage 3 and its evaluation. In Section 4 we discuss the automation of stage 7 and its evaluation. In Section 5 we discuss and evaluate search in the *LoLaLi* concept hierarchy. We conclude in Section 6.

2 LoLaLi: Logic and Language Links

Our work, and the experiments on which we report below, take place in the setting of a digital library project. Specifically, the *Logic and Language Links* (LoLaLi) project [4, 16] explores methods to extend the traditional format of scientific handbooks with electronic tools. These tools help readers explore the content of the handbook and make it easier to locate relevant information.

As a case study the project focuses on the *Handbook of Logic and Language* [28], a 20 chapter, 1200 page publication; for our experiments we used the LaTeX sources (about 4.5MB of text). The *LoLaLi* project uses a WordNet-like concept hierarchy to provide access to (an electronic version of) the handbook. Concept hierarchies are often used for navigating through large collections of documents [30, 17]. They are useful for the organization, display and exploration

Figure1. An excerpt from the *LoLaLi* concept hierarchy.



of a large amount of information, and users carrying out a hypertext search task who have hierarchical browsing patterns perform better than users with sequential browsing paths [20]. Hence, architectures for electronic handbooks should allow for, or even enforce, hierarchical patterns: a concept hierarchy is a good way of doing this. The *LoLaLi* concept hierarchy is being built by hand, by domain experts, who have currently collected, organized, and related close to 600 concepts. At the back-end, a Sesame-based server stores the hierarchy information, which is edited and updated through a collection of purpose-built scripts and Protege. In Section 3 we discuss how basic IR techniques can help concept authors determine which concepts to consider for inclusion in the hierarchy.

Each concept in the *LoLaLi* hierarchy is annotated with a gloss, which briefly describes it. Moreover, concepts come with a longer description, also provided by the authors of the concept. The hierarchy consists of a TOP concept, with four main branches underneath it: *computer science*, *mathematics*, *linguistics*, and *philosophy*, organized by relations of subtopic-supertopic. The latter relations are *typed*, and the types include 'is-a' and 'part-of.' The *LoLaLi* hierarchy is a graph rather than a strict tree, as multiple parenthood is allowed; see Figure 1. Non-hierarchical relations are also allowed, and are used for navigational purposes; these include 'sibling,' 'other meanings,' and 'associated concepts.' Concepts in the *LoLaLi* hierarchy are also connected to external resources. Chief among these is the *Handbook of Logic and Language*; other examples include links to highly relevant online tools and demos. The (hypertext) links to the *Handbook* take a concept in the hierarchy as source and highly relevant segments in the *Handbook* as targets. In Section 4 we describe how IR techniques help address this task.

At present, users can access an early 'alpha' version of the hierarchy. Following the outcomes of an early user study, navigation along the semantic relations mentioned above has been complemented with search facilities that allow users to locate concepts in the hierarchy in an arbitrary manner. In Section 5 we describe and assess the underlying IR techniques.

3 Assisting Ontology Builders

When building an ontology for an established scientific domain, as in the *LoLaLi* project, there is a wealth of literature whose content should be "covered" by the ontology. We report on IR support for addressing the following question: Which concepts should be included in the ontology? Rather than manually mining the

Table1. Precision at different ranks in the result lists.

rank	10	25	50	100	250	750
precision at rank (*Handbook*)	1.00	0.96	0.9	0.85	0.79	0.55
precision at rank (relative to the CLEF corpus)	1.00	1.00	0.98	0.99	0.94	0.80

literature, we describe methods to identify candidate concepts from domain specific text using term extraction. Concept names are usually noun phrases. Hence, recognizing noun phrases is likely to be a good first step for detecting candidate concepts. We distinguish between two cases, as they exploit different techniques: single word candidates and multi-word candidates.

3.1 Single Noun Concepts

To discover interesting nouns, we first POS-tag the *Handbook* text, and then select all nouns. We used two ways to rank them: by raw frequency, and by relative frequency, that is, by the number of occurrences divided by the number of occurrences in a general purpose document collection (we used the English CLEF collection [5]). The resulting lists were assessed by three assessors who were asked, for each noun in the result lists whether they would include it in a comprehensive list of important or useful notions in the domain, aimed at novices and experts. For our "gold standard", a noun was considered relevant if the majority of assessors deemed it relevant.

With this gold standard, we computed precision @ n scores (what fraction of the top n results is relevant?), for increasing values of n; see Table 1, where the second row concerns the result list ordered by raw frequency, and the third the result list ordered by relative frequency. Surprisingly, even the raw frequency result list, is of very high quality, with now non-relevant high-frequency nouns in the top. And by taking into account domain specificity (as in the list ordered by relative frequency), very high precision scores can be obtained. What about recall? It is hard, if not impossible, to compile an exhaustive list of important or useful nouns in the domain of the *Handbook*. Instead, we decided to approximate recall by using *concept recall* (CR): what fraction of the single noun concepts in the *LoLaLi* hierarchy did we identify, and what were their ranks in the result lists? Of the 522 concepts in the version of the concept hierarchy used, 158 are single nouns; hence, CR was measured against those 158. The noun extraction algorithm identified 77% (121) of the single noun concepts in the *LoLaLi* hierarchy; 70% of these are in the top 750. While this is not a perfect recall score, our ontology builders found the suggestions to be very helpful in further development of the hierarchy, telling us that the suggestions often inspired them to think of additional concepts, hence indirectly addressing the recall problem.

3.2 Multi-word Noun Phrases

Let us turn to the extraction of multi-word noun phrases now. We present a simple yet useful method that is based on collocations and that can be subdivided into three steps: (1) Shallow parse the text. (2) Filter out word sequences

Table2. Part-of-speech tag patterns for collocation filtering.

POS-tag pattern	example collocation
Adjective Noun	*logical study*
Noun Noun	*computer science*
Adjective Adjective Noun	*floating decimal point*
Adjective Noun Noun	*recursive enumerable set*
Noun Adjective Noun	*card programmed calculator*
Noun Noun Noun	*program storage unit*
Noun Preposition Noun	*theory of computation*

with interesting POS-tag patterns for closer examination. (3) Decide for each word sequence if it is a noun collocation. Step 1 is done with Schmid's *Tree-Tagger* POS-tagger [23]. Step 2 is accomplished by a method due to Justeson and Katz [12, 18], which uses the POS-tag patterns shown in Table 2. We scan the tagged text and discard everything that does not match one of the listed POS-tag patterns. Step 3 is done by testing whether the words in the sequence occur together significantly more often than is to be expected if all the words in the text would be ordered randomly. Following Krenn and Evert [15], who addressed the related task of detecting PP-Verb collocations, we use the the t-test for addressing Step 3. Our null hypothesis will be that, in the text, the words that make up the sequence appear completely independently of each other.

When we apply our multi-word method to the *Handbook of Logic and Language*, we get promising results. As an example, the 10 noun collocations with the highest t-scores are shown in Table 3. Exactly how well did we do? As in the

Table3. The top 10 collocations extracted from the *Handbook* (left) and our web collection (right), with "usefulness" assessments by all three assessors, together with the "gold standard" (last column).

noun-collocation	useful?				noun-collocation	useful?			
natural language	yes	yes	yes	*yes*	computer science	yes	yes	yes	*yes*
computer science	yes	yes	yes	*yes*	other hand	no	no	no	*no*
modal logic	yes	yes	yes	*yes*	natural language	yes	yes	yes	*yes*
lambda calculus	yes	yes	yes	*yes*	university press	no	no	no	*no*
situation theory	yes	yes	yes	*yes*	modal logic	yes	yes	yes	*yes*
discourse representation	no	yes	yes	*yes*	induction hypothesis	yes	yes	yes	*yes*
artificial intelligence	yes	yes	yes	*yes*	first order	yes	yes	yes	*yes*
phrase structure	yes	yes	yes	*yes*	district page	no	no	no	*no*
other hand	no	no	no	*no*	description post	no	no	no	*no*
proof theory	yes	yes	yes	*yes*	post manual ref	no	no	no	*no*

single noun case, we use concept recall (CR) and precision (P) to answer this question. Of the 522 concepts in the version of the concept hierarchy used, 364 are multi-word expressions; hence, CR was measured against those 364. Working on the *Handbook* our algorithm yielded 3896 collocations, 99 of which are concepts from the hierarchy. I.e., we found 28% of the multi-word concepts; 73% of these are in the top 750. Concerning P, we asked our three assessors to assess the candidate concepts returned (as in the earlier single noun case). Table 3 contains a sample of results produced, together with human assessments. Table 4 contains the resulting precision scores, at different ranks; the precision drops sharply as we move down the result list.

Table4. Precision at different ranks in the result list.

rank	10	25	50	100	250	750
precision at rank	0.90	0.76	0.74	0.65	0.67	0.63

While precision, and especially early precision, is at an acceptable level, concept recall leaves something to be desired. There are several ways to improve recall: develop more extraction patterns, make the patterns less strict, or increase the amount of data they work on. The second option might hurt precision too much, and the first likely yields highly specific patterns, making little or no difference in terms of concept recall. We go for the third option: many interesting noun phrases only occur once in the *Handbook*, and since our detection method essentially works through redundancy we will not be able to find those words.

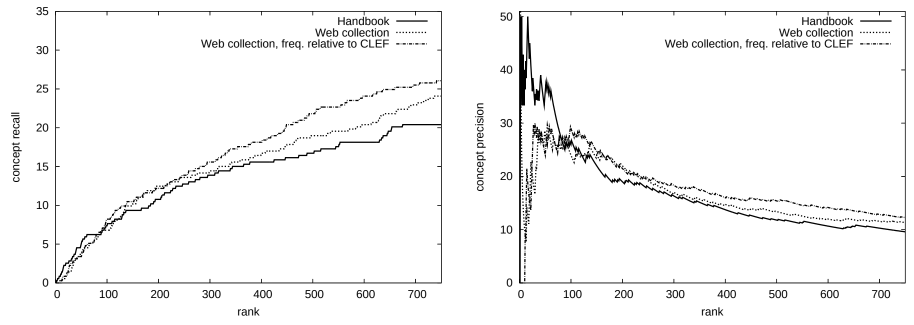
To create a larger corpus of relevant data, we proceeded as follows. Each of the 522 concepts in the *LoLaLi* hierarchy was fed to a web search engine (as a separate query), while restricting the output to PDF files (assuming that this restriction would increase the chance of retrieving scientific papers in our domain, rather than arbitrary documents). Per query, the top 20 results were kept; text was extracted using *pstotext*, producing 358MB of usable text. We extracted 206,475 collocations; a total of 197 (out of 364) concepts were found, and more importantly, 44% of those were found amongst the top 750 results. So, CR has certainly gone up, when compared against the results of running our algorithm against the *Handbook* text. The top 10 results (Table 3, right) reveal that early precision is seriously hurt by moving to our Web collection. The precision at various ranks in the result list for the Web collection does not drop as dramatically as for the *Handbook*; in fact, it goes up (Table 5, row 3). In Table 5 we also list the precision figures for the Web collection, relative to the CLEF corpus (bringing in domain specifity, as in the earlier single noun case). Domain specificity helps to get rid of expressions such as ‘other hand’, but it pushes expressions such as ‘next section’ to the top of the ranking, which explains the low p@10 score in row 3.

Table5. Precision at different ranks in the result list, *Handbook* vs. Web collection.

rank	10	25	50	100	250	750
precision at rank (*Handbook*)	0.90	0.76	0.74	0.65	0.67	0.63
precision at rank (*Web collection*)	0.50	0.40	0.56	0.53	0.56	0.47
precision at rank (*Web collection, rel. to CLEF corpus*)	0.20	0.52	0.54	0.60	0.62	0.55

To examine the interplay between precision and recall, we looked at the *concept* precision (in addition to concept recall, both in terms of concepts from the *LoLaLi* hierarchy), and compiled plots for concept precision and concept recall. In Figure 2 we plotted concept recall (Left) and concept precision (Right) of collocations found, in the *Handbook*, in our Web collection, and in the latter,

Figure2. (Left) The percentage of concepts present in the *LoLaLi* hierarchy identified at different points in the ranking. (Right) The fraction of concepts present in the *LoLaLi* hierarchy at different points in the ranking.



relative to the CLEF collection; the rank (plotted on the X-axis) is obtained by sorting by *t*-test score. As is to be expected, for the larger Web collection, concept recall is highest, followed by the Web collection-relative-to-CLEF, followed by the *Handbook*. For concept precision, the relative order is reversed.

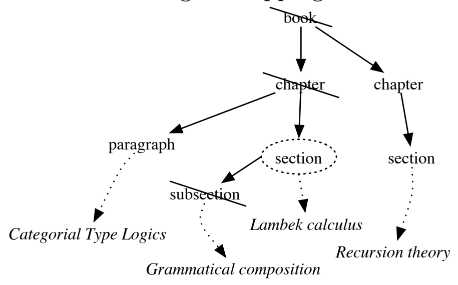
3.3 Conclusions and Further Steps

A simple single noun method and a simple collocation-based method can both yield valuable suggestions for concepts to be included in a concept hierarchy, thus addressing step 3 from Antoniou and Van Harmelen's list (see Section 1). For extracting multi-word expressions, more data proves useful to improve recall. Our results were further improved by filtering out the generic English expressions. Our scores are far from perfect, but as a source of suggestions, our ontology builders found the output of our methods extremely valuable. It may help to make our Web collection more focussed; if some concepts are available before collocation detection takes place, they could be used to constrain the text: new interesting concepts can be expected to occur in the proximity of the old ones.

4 Automatically Defining Instances

Ontologies rarely exist for their own sake, and their envisaged usage determines how ontologies should be populated. In settings where ontology-like structures are being used as navigational aids, an important group of instances are hypertext links to (fragments of) documents. In this section we describe and evaluate methods for automatically defining such instances, again within the setting of the *LoLaLi* project. The task we are addressing is to link concepts in the *LoLaLi* hierarchy to highly relevant text fragments in the *Handbook*. We view this task as a high-precision information retrieval task by treating every concept as a topic and every text fragment as a document: to define the required instances, we need to identify highly relevant text fragments for every concept in the *LoLaLi* hierarchy. How much of the specific digital library and ontology aspects that we

Figure3. Eliminating overlapping units of retrieval



have access to, can usefully exploited to address the task at hand. Our strategy is an incremental one: starting from a simple baseline we determine the impact of exploiting document structure, text markup cues, and collocations.

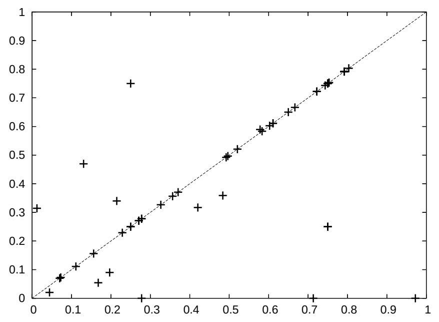
4.1 Under the Hood

The document collection in which we have to identify link targets consists of LaTeX documents: semi-structured data with explicit markup of segments: `\chapter`, `\section`, `\subsection`, etc. These annotations provide a segmentation of the entire *Handbook* into (we assume) topically coherent text fragments. As hyperlink targets (that is, as units to be returned by the retrieval engine), we allow any segment at any level, from chapter down to paragraph. While it is sensible to allow this (as some topics are discussed in a single paragraph only, while others may exhaustively cover bigger units), it does raise a problem. If a subsection contains a relevant piece of text, then so does the section it belongs to and so does any larger surrounding annotation. How do you decide how small or large the unit of retrieval should be? Following our experience with XML retrieval [13], we decided not to allow returning overlapping pieces of text. In cases where we have to decide between two overlapping units of retrieval we choose the one ranked highest by the retrieval system. This is illustrated in Figure 3: if we choose to return the section then we are not allowed to return anything contained in that section or anything that contains the section [14].

To simplify the evaluation of the hyperlink generation task, we made use of the entries from the back-of-the-book index of the *Handbook*. Specifically, we used all entries in the back-of-the-book index that also occur in the *LoLaLi* hierarchy; there are 141 such entries. Each entry is explicitly marked up in the LaTeX source of the *Handbook* (with the `\index{<entry>}` command); on average, an entry comes with three occurrences of the corresponding `\index{..}` tag. Our "gold standard" used for assessment consists of the 141 concepts mentioned as our "topics" (in IR parlance). A text segment is considered relevant for a topic if it is marked up with the corresponding `\index{..}` command. Clearly, the quality of our results depends on the quality of the back-of-the-book index.

We use *Incremental R-Precision* as our metric; it returns 0 when none of the relevant documents have been found for a given topic, and 1 when all have been found. It has a higher value when a returned document appears high in the

Figure4. An example quantile plot of the baseline compared to a run that exploits emphasis annotations.



ranking than when it appears further down in the ranking. Given a topic, $p@n$ (precision at n) is $|\{d \in \mathit{Relevant} \mid \mathrm{rank}(d) \leq n\}|/n$, and

$$\mathit{Incremental\ R\text{-}Precision} = \sum_{n=1}^{|\mathit{Relevant}|} \frac{p@n}{|\mathit{Relevant}|},$$

where *Relevant* is the set of relevant documents for the given topic. The score for a run or experiment is obtained by averaging over all topics.

If the distribution of the performance differences is not skewed and if there are few outliers then it is safe to use the paired t-test for the evaluation of information retrieval [9]. These two conditions can be tested with quantile plots. The points in such a plot are supposed to be balanced around the identity line. We checked that this is the case for all our data; an example is shown in Figure 4. Hence, we test how significant the differences between two methods are by comparing the results per query with the paired t-test. Significant differences (95% confidence, $t \geq 1.645$) are marked with $\triangle$,$\triangledown$ and material differences (99% confidence, $t \geq 2.346$) with $\blacktriangle$,$\blacktriangledown$. Insignificant differences are not marked.

4.2 Experiments

Our baseline uses a standard *tf.idf* weighting scheme [2], and we reduce all words to their stem using the *TreeTagger* lemmatizer [23]. On top of that we experimented with additional layout cues, and with collocations. First, words that appear inside the titles of units (such as sections) are likely to be good indicators of the main topic covered by those units. Word sequences that are emphasized (the *Handbook* convention is to use `\emph{..}` and `{\em ..}`) seem more important than others. We implement the preference for terms with a certain annotation, or occurring in a certain context, fairly naively, by increasing the score of the text segment at hand. If a segment starts with a title and that title contains the query words we double the segment's score. If the segment's title contains nothing else, i.e., it coincides with the query, we double the segment's score again. If a document contains `\em` or `\emph` environments that literally contain the query we double the score; and if the emphasis is on nothing but the query we double it again. The experimental outcomes are shown in Table 6

Table6. Link generation experiments. (Top) Runs that exploit layout cues. (Bottom) Runs that exploit collocations.

Experiment	*Inc. R-Precision*
baseline	.35
title bonuses	.41 ▲
emphasis bonuses	.33
title and emphasis bonuses	.39 △
collocation bonus	.35
title and collocation bonus	.46 ▲ (relative to .41)
emphasis and collocation bonus	.33

(Top). There is a material difference for runs that prefer titles, but no significant difference between runs that prefer emphasized text and those that do not: even though emphasis is frequently used to stress key phrases, there appear to be more cases where it is used to stress unimportant words.

With the resources developed in the previous section, we tried to boost the performance of our link generation method. Using the collocations obtained from our Web collection, we increased a segment's score for a topic whenever it contained a collocation from the topic. Using a similar naive scoring mechanism as before, whenever a segment contains one or more collocations its score is doubled. (A segment's score can only be doubled once.) The idea is to bring some word order to the topic-segment matching process, which should increase precision whenever we know that word order may matter (as is the case for our collocations: e.g., *proof theory* is likely to be more informative than *theory proof*). Table 6 (Bottom) shows the results of the experiments with collocation bonuses. There is no significant difference between the baseline and runs that only exploit collocations, but when we combine title and collocation preference there is a material difference with the baseline and even with the title preference only run: the literal occurrence of the query in a segment's body or emphasized text does not say more about a segment's probability of being relevant to the query than its *tf.idf* score. However, the (literal) occurrence of query terms in both a segment's title and body is overwhelming evidence of relevancy.

4.3 Conclusions and Further Steps

We showed that exploiting title fields and collocations can improve the performance of automatic hyperlinking. The methods we used are quite crude and it is very likely that further improvements can be realized by optimization. A more careful method to combine the evidence provided by *tf.idf*, title markup and collocations could be beneficial to the results, thus leading us to consider more sophisticated weighting schemes that standard ones used here.

5 Searching in Ontologies

After Sections 3 and 4, which were aimed at IR support for ontology construction, we change tack and address support for end users that access ontologies for navigational purposes. The process of browsing through the ontology to find a concept can give the user a good impression about how the ontology (and the underlying domain) is organized and how the concepts are related, but it can also be a difficult and laborious process. Examples where browsing frustrates the information access process are well-known, and include such things as not knowing were in the hierarchy a concept might be located, using a different vocabulary, or simply getting lost because of the sheer size of the ontology. In such cases IR techniques can help address these information needs. Instead of following the semantic relations hard-wired in an ontology, IR offers random access to an ontology and a flexible interpretation of a user's information need.

The task we want to address in the remainder of this section is the following: given a query, find relevant concepts in a concept hierarchy. In other words, users' information needs are formulated using arbitrary keywords, while the "documents" to be returned are concepts, in the *LoLaLi* hierarchy.

5.1 Under the Hood

When trying to retrieve relevant concepts from an ontology, we have to deal with a number of issues. (1) The queries tend to be very short. The number of keywords per topic can be expected to be roughly equal to that of web search engine queries, on average two keywords per topic [11, 3]. (2) The documents are very short too. Even if we have an extended description of the concepts (as many of the concepts in the *LoLaLi* hierarchy do), the documents to be retrieved are short compared to standard test collections. (On average, the descriptions are 23.3 words long, and the concept names 1.8 words; so the average document is about 26 words long.) (3) The document collection is small. This means that recall may be an issue. Summing up, retrieval against the *LoLaLi* hierarchy is a high precision task, but, potentially, with high recall requirements.

Our topics, of which there are 26, have been made by four different authors and are based on what first-year artificial intelligence students of the University of Amsterdam typed into a prototype search engine in a user test of the *LoLaLi* user interface. A "gold standard" was established using three assessors, in the same manner as for the link generation task addressed in Section 4. The metric used is also the same as in Section 4: incremental R-precision.

All documents and topics are stripped of all non-word characters except hyphens and, as with the hyperlinking task described in Section 4, lemmatized using *TreeTagger* [23]. Each topic is then compared to documents in the inverted index, producing a ranked list of documents, which is presented to the user.

5.2 Experiments

As a baseline we choose a simple *tf.idf* based retrieval model. As in the previous section, we are interested in finding out to which extent the structure of the concepts and concept hierarchy can help improve retrieval effectiveness. Specifically, we tried the following ideas on top of the baseline, all aimed at high precision without hurting recall: (1) Give concepts of which the name exactly matches the topic a bonus over other concepts. e.g. If the user types in 'logic' then the concept 'logic' would be preferred over 'modal logic'. (2) Give concepts that share a collocation with the topic a bonus over concepts that share the constituents in some other order. (3) Group concepts that are related to each other together, allowing concepts that are related to concepts in the top of the ranking to benefit.

The first thing we try to improve over the baseline is to exploit characteristics of the syntax of the documents. The results from our automatic hyperlinking experiments suggests that we should prefer words in titles to words in the body of a section. Similarly, we will prefer occurrences of query terms in a concept's name to cases where it appears only in its description; in the former case the scores are simply doubled. Since the concept hierarchy is filled with very specific terms, the influence of word order can be expected to be even greater in this experiment than with the automatic hyperlinking. So we will try applying the same method as with that experiment too. When a concept contains a collocation that also appears in the query, we double its score. When a concept's name is exactly equal to the topic it is unlikely that the user desires another concept. So we apply the same technique as before: we double the concept's score.

The results of these techniques are shown in Table 7 (Top). Given the small

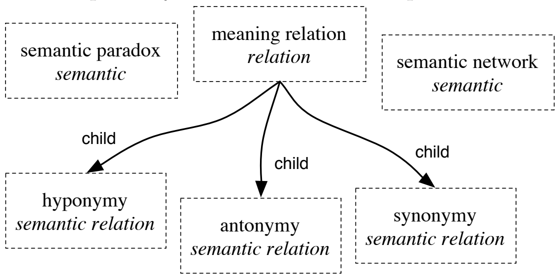
Table7. (Top) Results of collocation and exact match bonuses. (Bottom) Results of concept relation based reranking.

	Inc. R-Precision
baseline	.53
only collocations	.55
only exact match	.56
coll. and exact match	.58
grouping	.65 ▲
grouping adding context	.68 ▲
grouping with coll. and exact match	.67 ▲
grouping adding context with coll. and exact match	.74 ▲

number of topics we can not conclude that the improvement is significant. We can only say with about 90% confidence that there is a difference.

We now turn to further improvements over the baseline, ones that try to exploit the semantics that are encoded by the relations in the concept hierarchy. For brevity we only report on the use of the subsumption relations is-a and subclass-of for this purpose. Concepts inherit information from their parents and specify them in some way; the converse holds for parent concepts in relation to their children. Queries have to be answered as precisely as possible: not too general, and not too specific. Often, the concept that gets the highest score from a weighting scheme is the right concept, but sometimes things are more complicated, as in Figure 5. Here, the query is 'semantic relation' and the desired concept is 'meaning relation,' but the only concepts that contain 'semantic relation' literally are the children of 'meaning relation.' To address this problem we propose to rerank the list of concepts produced by the weighting scheme in such

Figure5. Related concepts may be better than concepts that match the query well.



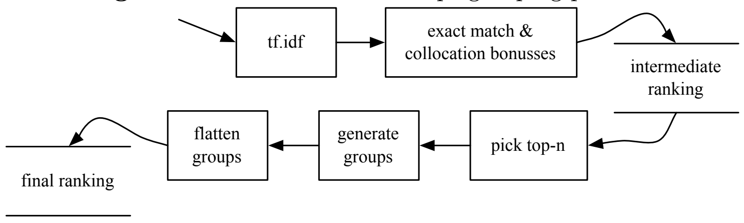
a way that related concepts appear close to each other. This allows concepts that are related to a concept that gets a high score to benefit from this relation and move up the ranking. The rules we use to group related concepts together are listed below:

1. Every matching concept should be clustered under its parent; this parent concept shows the context of the concept.
2. Matching concepts with the same parent should be put together under that common parent, ordered by their own score.
3. Every chain of parent-child related matching concepts should end in a non-matching concept that shows the context of the chain.
4. Unrelated clusters are joined together as a forest, ordered by the maximum score of the cluster.
5. When parents have the same children they are joined together and get the highest of the two scores.

These rules let parents benefit from their children and vice versa, and they let siblings benefit from higher scoring siblings.

To constrain the size of the groups, and, hence, to allow more than one group to fill the top of the ranking, we discard everything after a certain discrete cut-off point and apply grouping on the concepts that are left. After some experimentation, we chose 10 as the cut-off point, based on the average number of relevant documents per topic. An overview of the retrieval-plus-reranking process is shown in Figure 6. The results of the grouping rules are shown in Table 7 (Bottom). Even with the small number of queries we can conclude that there is a material improvement of the scores when we exploit concept relations, and that

Figure6. Overview of the concept grouping process.



the combination of all techniques discussed so far improves most, suggesting that distinct techniques have distinct effects.

5.3 Conclusion and Further Steps

While search in a concept hierarchy has special features that may require special IR approaches, we have seen that standard retrieval techniques offer acceptable levels of performance, but that material improvements can be achieved by exploiting the structure of the concept hierarchy. This, we believe, is a very interesting combination of IR and Semantic Web techniques. Obvious further steps worth exploring to improve ontology search include using additional relations from the hierarchy, as well as using different retrieval models.

6 Conclusion

We used collocation-based keyword extraction to suggest new concepts, and we studied the automatic generation of hyperlinks to automate the population of ontologies with instances. We evaluated our methods within the setting of an ontology-based digital library project. Within the same setting we explored retrieval methods aimed at helping users search an ontology, and found that a mixture of IR techniques and result reranking based on the underlying concept hierarchy was the most effective method.

We should point out that, except for the grouping methods deployed in the previous section, the IR methods that we used are mostly standard ones; however, their applications in the Semantic Web setting are novel. The methods and results on which we reported in this paper should be interpreted as providing baselines for their respective tasks. There is a wealth of IR methods that, we believe, can make further contributions to ontology construction, and to the effective usage of ontologies.

Acknowledgments

Maarten de Rijke was supported by the Netherlands Organization for Scientific Research (NWO), under project numbers 612-13-001, 365-20-005, 612.069.006, 612.000.106, 220-80-001, 612.000.207, 612.066.302, and 264-70-050. Maarten Marx was supported by a grant from NWO under project number 612.000.106.

References

1. G. Antoniou and F. van Harmelen. *A Semantic Web Primer.* MIT Press, 2004.
2. R. Baeza-Yates and N. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press, 1999.
3. N.J. Belkin, C. Cool, D. Kelly, G. Kim, J.-Y. Kim, H.-J. Lee, G. Muresan, M.-C Tang, and X.-J Yuan. Query length in interactive information retrieval. In *Proc. of SIGIR*, 2003.

4. C. Caracciolo. Towards modular access to electronic handbooks. *Journal of Digital Information*, 3(4), 2003. Article No. 157, 2003-02-19.
5. CLEF. Cross-Language Evaluation Forum, 2003. URL: `http://www.clef-campaign.org/`.
6. CoNLL. Conference on Natural Language Learning, 2003. URL: `http://cnts.uia.ac.be/signll/conll.html`.
7. J. Heflin, J. Hendler, and S. Luke. Shoe: A prototype language for the semantic web. *Linköping Electronic Articles in Computer and Information Science*, 6, 2001. URL: `http://www.ep.liu.se/ea/cis/2001/003/`.
8. D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, Universiteit Twente, 2000.
9. D. Hull. Evaluating evaluation measure stability. In *Proc. of SIGIR 2000*, 2000.
10. INEX. INitiative for the Evaluation of XML Retrieval, 2004. URL: `http://inex.is.informatik.uni-duisburg.de:2004/`.
11. B.J. Jansen. An investigation into the use of simple queries on web ir systems. *Information Research: An Electronic Journal*, 6(1), 2000.
12. J.S. Justeson and S.M. Katz. Technical terminology: some linguistic properties and an algorithm for identification in text., 1995.
13. J. Kamps, M. Marx, M. de Rijke, and B. Sigurbjörnsson. XML retrieval: What to retrieve? In *Proc. of SIGIR*, 2003.
14. G. Kazai, M. Lalmas, and A. de Vries. The overlap problem in content-oriented XML retrieval evaluation. In *Proc. of SIGIR*, 2004.
15. B. Krenn and S. Evert. Can we do better than frequency? A case study on extracting PP-verb collocations. In *Proc. ACL Workshop on Collocations*, 2001.
16. LoLaLi. Logic and Language Links, 2003. URL: `http://lolali.net/`.
17. M. Dewey. Dewey Decimal Classification, 1870. URL: `http://www.oclc.org/dewey`.
18. C.D. Manning and H. Schütze. *Foundations of Statistical Language Processing*. The MIT Press, 1999.
19. J. Mayfield and T. Finin. Information retrieval on the Semantic Web: Integrating inference and retrieval. In *Proc. SIGIR 2003 Semantic Web Workshop*, 2003.
20. J.E. McEneaney. Visualizing and assessing navigation in hypertext. In *Proc. ACM Conference on Hypertext and Hypermedia*, pages 61–70, 1999.
21. NTCIR. NII-NACSIS Test Collection for IR Systems, 2003. URL: `http://research.nii.ac.jp/ntcir/`.
22. M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *Pro. SIGIR 1999*, pages 206–213, 1999.
23. H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proc. of International Conference on New Methods in Language Processing*, 1994.
24. Senseval. Evaluation of Systems for the Semantic Analysis of Text, 2003. URL: `http://www.senseval.org/`.
25. U. Shah, T. Finin, A. Joshi, R.S. Cost, and J. Mayfield. Information retrieval on the semantic web. In *Proc. CIKM 2002*, 2002.
26. H. Stuckenschmidt and F. van Harmelen. Approximating terminological queries. In *Proc. FQAS'02*, 2002.
27. TREC. Text REtrieval Conference, 2003. URL: `http://trec.nist.gov/`.
28. J. van Benthem and A. Ter Meulen, editors. *Handbook of Logic and Language*. Elsevier, 1997.
29. E.M. Voorhees. Overview of the TREC 2003 Question Answering Track. In *NIST Special Publication 500-255: TREC 2003*, 2004.
30. Yahoo! Yahoo!, 1995. URL: `http://www.yahoo.com/`.