Slicing through the scientific literature

Christopher J. O. Baker[1], Patrick Lambrix[2], Jonas Laurila Bergman[2], Rajaraman Kanagasabai[3], Wee Tiong Ang[3]

[1] Department of Computer Science & Applied Statistics, University of New Brunswick, Canada
[2] Department of Computer and Information Science, Linköpings universitet, Sweden
[3] Data Mining Department, Institute for Infocomm Research, Agency for Science Technology and Research, Singapore

Abstract. Success in the life sciences depends on access to information in knowledge bases and literature. Finding and extracting the relevant information depends on a user's domain knowledge and the knowledge of the search technology. In this paper we present a system that helps users formulate queries and search the scientific literature. The system coordinates ontologies, knowledge representation, text mining and NLP techniques to generate relevant queries in response to keyword input from the user. Queries are presented in natural language, translated to formal query syntax and issued to a knowledge base of scientific literature, documents or aligned document segments. We describe the components of the system and exemplify using real-world examples.

This is a preprint of the paper published by Springer-Verlag:
Baker C, Lambrix P, Laurila Bergman J, Kanagasabai R, Ang WT, Slicing through the scientific literature, Proceedings of the 6th International Workshop on Data Integration in the Life Sciences - DILS09, LNBI 5647, 127-140, Manchester, UK, 2009.
The original publication is available at www.springerlink.com.
doi: 10.1007/978-3-642-02879-3_11

1 Introduction

Indispensable components of knowledge discovery infrastructures are online repositories of freely available unstructured text from the scientific literature. Information retrieval techniques are commonplace for the harvesting of documents while conversion of document formats to make them amenable to text mining is an ongoing irritation. Text mining techniques are swiftly being deployed in industrial strength platforms albeit with the need for domain specific customization. Yet despite the improving proficiency of text mining tools, text extracts are not always readily accessible to end users without augmentation with semantic metadata.

At the same time existing search paradigms using keyword search over indexes of text summaries continue to limit end users. Users do not know how to use tools that allow them to formulate more expressive queries e.g. queries involving known relations between entities. This amounts to a lack of knowledge of available search technology.

Nor do users know the limits of the domain coverage of a given resource that they are querying, for example does PubMed include documents on wearable electronic devices for personal health monitoring or is it beyond its scope. It would surely save users time if they know the extent of the answers they can obtain from a given body of knowledge. This amounts to a lack of knowledge of the domain.

In addition to the challenges posed by lack of semantic annotation to mined raw text fragments and poor cognitive support for query composition, system developers are faced with a lack of reusable domain specific metadata to facilitate semantic annotation. Semantic annotation of text segments relies on the existence of curated domain-specific controlled vocabularies and the mapping of semantic types in ontologies to canonical named entities. Up until recently the use of sophisticated 'domain' metadata was not widely adopted for the indexing of text segments derived from scientific documents, in part due to the dearth of suitably designed ontologies. Using domain ontologies scripted in W3C standard ontology languages, rich in expressive power and inference capability, to annotate mined text segments makes possible an advanced range of literature navigation and search capabilities.

In addition to the search of documents based on named entities or predicates using keywords, relationships or class names as entry points, metadata specific graph mining can further augment search tools. Moreover customized search applications can be readily developed for a multitude of end user requirements including content recommendation. In this paper we describe infrastructure for navigating literature that provides to users, through a natural language interface, (i) ontology driven query, (ii) context of query terms, (iii) cross domain query which obviates the need for users to have knowledge about query languages and underlying data sources, while providing an overview of the scientific domain, connections between entities in the domain and a comprehensive understanding of decisions in query strategy. Search results are linked to scientific documents, text segments and ontology terms. Our contributions in this paper include the definition of the theoretical foundations for this paradigm as well as a framework for systems based on this paradigm. Further, we illustrate this paradigm with some examples.

The remainder of the paper is organized as follows. In section 2 we describe an example scenario for how our new search paradigm can be used. The theoretical foundations and a framework for systems following the paradigm are developed in sections 3 and 4, respectively. Further, we show an implementation of the framework in section 5 and use this implementation to revisit our example scenario (section 6). Related work is given in section 7.

2 Example scenario

A user performs a keyword search, for example, 'lipid'. In current systems all documents containing the word 'lipid' are retrieved. Some systems that implement ontology-based querying, may also retrieve the documents that contain words representing subconcepts of lipid. The user, however, is not interested solely in retrieving these documents or abstracts, but also wants to investigate relationships between lipids and other concepts. The problem is that she does not know what relevant questions can be asked.

Consulting ontologies for properties can provide this knowledge. For instance, in Lipid Ontology 'lipid' is related to a number of other concepts, such as to 'protein' via the relation 'interacts with' and to diseases based on the relation 'implicated in'. The user would want the system to describe the query 'which proteins interact with lipids that are implicated in a disease ?' and make this query available, along with other relevant lipid related queries, to the user in the form of a natural language query. In addition, the user would want to be able to access the information and context relevant to one or more keywords. This context would include connections between the keywords (possibly via other terms) as well as terms that are related to the keywords. Again, ontologies can provide for a context by investigating the neighborhoods and connections of the keywords to other terms.

We may even need multiple ontologies, for instance, to find the answer to the query 'Which lipids interact with proteins (from the lipid ontology) that are involved in signal processing (proteins involved in signal processing - from the signal ontology) and are implicated in causing a disease (implicated_in from the lipid ontology).

Moreover the ontologies can be instantiated with named entities extracted from scientific documents. This can generate the response that certain oxidized polyunsaturated fatty acids are involved in phosphorylating p53, which is involved in apoptosis, and implicated in ovarian cancer which is derived from one or more text segments instantiated to one or more ontologies. Furthermore because of the addition of expressive literature metadata to the ontologies, querying for provenance information (documents / journals / authors in which the information was contained) is made possible. We could ask natural language queries like 'In which documents have lipids interacting with signaling proteins, and known to cause disease, been found ?'

3 Theoretical foundations

One of the main foundations of our paradigm is the use of ontologies. Intuitively, ontologies (e.g. [12]) can be seen as defining the basic terms and relations of a domain of interest, as well as the rules for combining these terms and relations. In our approach ontologies are used to guide the user in asking relevant questions. The intuition is that the ontologies define the domain knowledge including our knowledge about the connections between different terms in the domain. Therefore, the relevant queries regarding a set of terms are the ones for which the terms are connected in the ontology.[1] In this section we formalize the notion of 'relevant queries'.

3.1 Slices

Many ontologies in the life sciences can be represented by graphs where the concepts are represented as nodes and the relations (including is-a relations) are represented as

[1] If the terms are not connected, then they are not useful for guiding the user in asking relevant questions based on the ontology. However, in that case a sub-set of the query terms may still be relevant. Also, if a co-occurrence of terms (even without relations) is useful for the user, then traditional search approaches can be used.

edges[2]. In this case, given an ontology, a relevant query including a number of concepts and relations from the ontology can be seen as a connected sub-graph of the ontology that includes the nodes representing the given concepts and the edges representing the given relations. We define this formally using the notion of *query graph*. Further, the set of query graphs including a number of concepts and relations from the ontology is called a *slice*.[3]

Definition 1 *Given a graph G = (N,E) where N is the set of nodes in G and E the set of edges, and a set of nodes C ⊂ N and a set of edges R ⊂ E, a* **query graph in G based on C and R** *is defined as a connected sub-graph G' = (N',E') of G where C ⊂ N' and R ⊂ E'.*

Definition 2 *Given a graph G = (N,E) where N is the set of nodes in G and E the set of edges, and a set of nodes C ⊂ N and a set of vertices R ⊂ E, a* **slice in G based on C and R** *is defined as the set of query graphs in G based on C and R.*

As an example, consider the ontology graph in figure 1 and assume query terms related to nodes 2 and 6. Then there are several relevant queries i.e. query graphs in this ontology based on nodes 2 and 6. For instance, the sub-graph containing nodes 2, 1, 3 and 6 and edges e1, e2 and e5 is a query graph based on nodes 2 and 6. Another query graph consists of nodes 2, 1, 4 and 6 and edges e1, e3 and e6. A slice based on nodes 2 and 6 is then the set of all possible query graphs based on nodes 2 and 6.

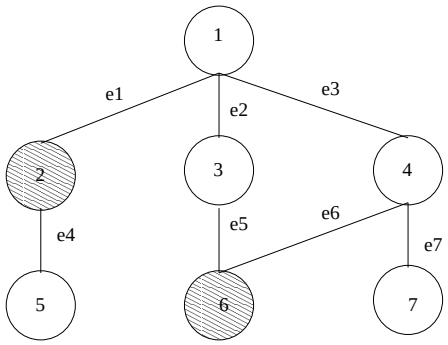

Fig. 1. Ontologies and query graphs.

There are a number of special cases of this definition. (i) When two or more concepts, but no relations are given (R = ∅), in this case we are looking for relevant queries containing given concepts only. While most keyword search algorithms would try to

[2] We assume undirected edges. An edge represents a relation and its inverse.

[3] For instantiated ontologies these definitions can be extended to also handle instances, by allowing nodes to represent instances. In our implementation (see section 5) we do allow queries to the knowledge base (see section 4) that involve instances.

find documents in which multiple terms co-occur, in this case there is an extra requirement that there are connections, albeit un-specified, between the search terms in the ontology, thereby augmenting the relevance of the returned documents; (ii) Where only a single concept and no relations are provided, in this case a slice represents all the relevant queries in which the query term features. Instead of just returning all documents containing the query term, this approach allows a user to browse the ontological environment of the term. It also guides the user in asking more specific questions, thereby removing many of the irrelevant documents.

3.2 Aligned ontology slices

In cases where we want to retrieve information that covers different but related domains or when we want to integrate information from different views on one domain, one ontology does not suffice. Queries will comprise of terms from different overlapping ontologies. Therefore an alignment i.e. a set of mappings between terms of overlapping ontologies, must be available. In the biomedical domain, for instance the Bioportal (bioportal.bioontology.org, [15]) repository of ontologies stores mappings between different ontologies and this can be used. If an alignment between the used ontologies is unavailable ontology alignment systems (e.g. overviews in [11, 17, 14, 9], the ontology matching book [5], and the ontology matching web site at http://www.ontologymatching-.org/) may be used for finding mappings.

When an alignment between the ontologies is given, we can deal with a query including terms from overlapping ontologies by connecting the part of a query using the terms in one ontology to the part of the query using the terms in another ontology through a mapping in the alignment.

In the definitions below, we first define an alignment in terms of a graph and then define query graphs with parts in two different ontologies.

Definition 3 *An* **alignment** *between G1 = (N1,E1) and G2 = (N2,E2) is a set of mappings between nodes in G1 and nodes in G2. The alignment is represented by a graph (NA,EA) such that*
(i) $NA \subset N1 \cup N2$*,*
(ii) each edge in EA connects a node in $N1 \cap NA$ *with a node in* $N2 \cap NA$*,*
and (iii) each node in NA is connected to another node in NA through an edge in EA.

The definition states that an alignment (set of mappings) is represented by a graph such that (i) the alignment graph uses only nodes from the source ontologies, (ii) an edge in the alignment graph represents a mapping between a node in the first ontology and a node in the second ontology, and (iii) every node in the alignment graph should participate in a mapping.

Definition 4 *Let G1 = (N1,E1) and G1Q = (NQ1,EQ1) be a query graph in G1 based on C1 and R1. Let G2 = (N2,E2) and G2Q = (NQ2,EQ2) be a query graph in G2 based on C2 and R2. Let A= (NA, EA) be an alignment between G1 and G2. An* **aligned query graph based on G1Q and G2Q given A** *is a connected graph G = (N, E) such that*
(i) $N \subset N1 \cup N2$*,* $E \subset E1 \cup E2 \cup EA$*,*

(ii) NQ1 ⊂ N, EQ1 ⊂ E, NQ2 ⊂ N, EQ2 ⊂ E,
and (iii) $\exists$ $n_1 \in NQ1$, $n_2 \in NQ2$, $n_{1a} \in N1 \cap NA \cap N$, $n_{2a} \in N2 \cap NA \cap N$, $e_a \in EA$ *such that: there is a path in* $G \cap G1$ *from* n_1 *to* n_{1a} *and a path from* n_{2a} *to* n_2 *in* $G \cap G2$*, and* e_a *is an edge between* n_{1a} *and* n_{2a} *in* $G \cap A$.

The definition states that: (i) the nodes in the aligned query graph belong to the source ontologies, and the edges in the aligned query graph belong to the source ontologies or to the alignment, (ii) the nodes and edges in the original query graphs are included in the aligned query graph, and (iii) the original query graphs are connected by at least one path going through a mapping in the alignment.

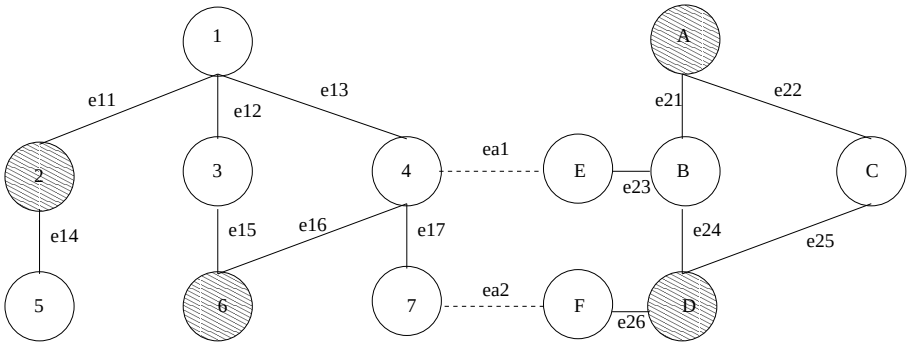

Fig. 2. Aligned ontologies and aligned query graphs.

As an example, consider the ontology graphs and alignment in figure 2. The alignment between the two ontologies is given by the mappings 4-E and 7-F. The query terms are represented by nodes 2, 6, A and D. The sub-graph containing nodes 2, 1, 3, and 6 and edges e11, e12 and e15 is a query graph based on nodes 2 and 6 in the first ontology. The sub-graph containing nodes A, B and D and edges e21 and e24 is a query graph based on nodes A and D in the second ontology. We can connect these two query graphs via the path containing the nodes 6, 4, E and B and the edges e16, ea1 and e23. One part of this graph is included in the first ontology, another part in the second ontology and a third part in the alignment. Therefore, one possible aligned query graph includes the nodes 2, 1, 3, 6, 4, E, B, A and D and the edges e11, e12, e15, e16, ea1, e23, e21 and e24. (Another possible aligned query graph may make use of the mapping 7-D.)

An aligned slice represents a set of aligned query graphs.

Definition 5 *Let S1 be a slice in G1 = (N1,E1) based on C1 and R1 and S2 be a slice in G2 = (N2,E2) based on C2 and R2. Let A= (NA, EA) be an alignment between G1 and G2. An* **aligned slice for S1 and S2 given A** *is defined as the set of aligned query graphs based on the query graphs in S1 and S2 given A.*

4 Framework

Using the theoretical foundations in section 3 we now proceed to define a framework for systems supporting our new search paradigm for literature document bases (see figure 3). As input the user gives a number of query terms. A first output is a list of suggestions for queries in natural language that are relevant with respect to the query terms and the ontologies. These queries can then be run and results are returned. A result can be in the form of knowledge extracted from the documents in the literature base or as documents or document segments.

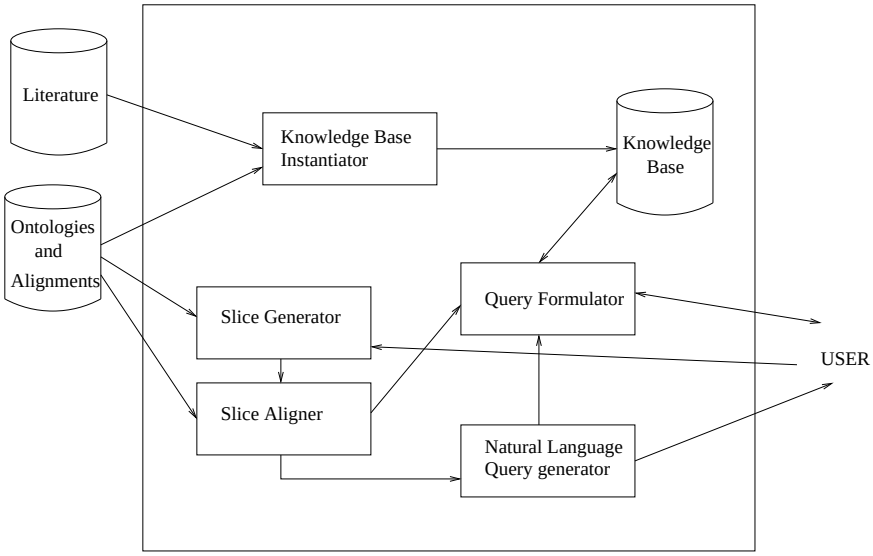

Fig. 3. Framework.

External Resources. The first external resource is the *literature document base*. It contains the documents that can be searched. The second external resource is an *ontology and ontology alignment repository*. It contains ontologies in which the query terms can be found as well as established alignments between the ontologies.

Computed Resources. The *knowledge base* contains instantiated ontologies. The instantiation represents two kinds of information. First, knowledge, in the form of named entities and relations from the literature, is extracted and normalised to canonical names - from which concept instances and relation instances are generated. This a form of semantic annotation. Further, the ontology instances are connected to instances of provenance documents and document segments in which they occur.

Process - Knowledge base Instantiation. The *knowledge base instantiator* creates the knowledge base with knowledge extracted from the literature and connections of the

ontology terms to documents or document segments. This component relies on different sub-components such as entity recognizers and text mining modules.

Process - Slice Generation and Alignment. The user drives the generation of slices by providing one or more query terms. For each of the ontologies in the ontology repository, the *slice generator* computes, given the query terms included in the ontology, a slice representing the relevant queries that are possible using the provided query terms and the ontology. The resulting slices are given to the *slice aligner*. This component generates aligned slices, representing queries involving terms from different ontologies, using the slices and the alignments in the repository. (If the query terms only occur in one ontology, no alignment is needed and this components just returns the slice.)

Translation - Slice to Query. The (aligned) slices represent sets of queries. The *natural language query generator* translates these queries to queries in human understandable natural language text. This component may have different sub-components such as generation, aggregation and surface realization / grammar checking of the query. These natural language queries are built from the labels of edges and nodes in the slices and presented to the user. Slices are also translated to the equivalent formal query, in syntax of the query language suitable for querying the knowledge base.

Query. When the user chooses a natural language query to be run, the *query formulator* issues the preformulated query representing the slice and the natural language query to the knowledge base and results are returned to the user.

5 Implementation

In this section we describe our current implementation of the framework and use the scenario from section 2 to exemplify the instantiation of the different components.

Literature document base. The literature document base used in our scenario was generated from a collection of 7498 PubMed abstracts that was identified by manual curation to be relevant to the subject of Ovarian Cancer (OC). Within this collection we found 683 papers that had lipid names from which 241 full papers were downloadable. Retrieved research papers were converted from their original formats to ascii text.

Ontologies. The ontologies that we used for this scenario are lipid ontology [2] and our version of the signal ontology. An alignment between these ontologies was generated using the ontology alignment system SAMBO [11].

Knowledge base. As representation language for the knowledge base we used OWL and adopted the conceptualization developed in our previous work [2] in which a Literature Specification of document metadata was introduced to the Lipid Conceptualization making it possible to instantiate simple axioms such as Lipid *Occurs_in* Sentence. The knowledge base instances are generated from full texts provided by the content acquisition engine using the BioText toolkit (http://datam.i2r.a-star.edu.sg/∼kanagasa/BioText/).

The instantiation of the knowledge base comprises of three stages: concept instance generation, property instance generation, and population of instances. Concept instances are generated by first extracting the name entities from the texts and then normalizing to canonical names and grounding them to the ontology concepts. We used a gazetteer that processes documents and recognizes entities by matching term dictionaries against the tokens of processed text, tagging the terms found [10].

We used the lipid name dictionary described in [2] which was a custom synthesis of terms from a Lipid Data Warehouse that contains lipid names from LIPIDMAPS, LipidBank and KEGG, IUPAC names, and optionally broad synonyms and exact synonyms. The manually curated Protein name list from Swiss-Prot (http://au.expasy.org/sprot/) was used for the protein name dictionary. A disease name list was created from the Disease Ontology of the Centre for Genetic Medicine (http://diseaseontology.sourceforge.net).

To evaluate the performance of our named entity/concept recognition we constructed a gold standard corpus of 10 full-texts papers related to the apoptosis (which is central to understanding ovarian cancer). We extracted 119 sentences and tagged the mentions of Protein name and Disease name. In these sentences we annotated all valid mentions of the two concepts and built the corpus. To evaluate performance of named entity/concept recognition a corpus without the concept annotations was passed to our text mining engine and the concepts recognized. Our system was evaluated in terms of precision and recall. Precision was defined as the fraction of correct concepts recognized over the total number of concepts output, and recall was defined as the fraction of concepts recognized among all correct concepts. The evaluation of entity recognition, in Table 1, shows that our text mining achieved performance comparable to that of the state-of-the-art dictionary-based approaches. In our future work, we plan to make use of advanced entity recognition techniques, e.g. fuzzy term matching and co-reference resolution, and also train our system on larger corpora, to address these issues.

Named Entities	Mentions		Precision	Recall
	Target	Returned		
Disease	32	37	0.54	0.62
Lipid	58	25	0.96	0.47
Protein	269	181	0.76	0.51
Micro average			0.75	0.51

Table 1. Precision and recall of named entity recognition

Our normalization and grounding strategy is as follows. Protein names were normalized to the canonical names entry in Swiss-Prot. Object property and Datatype property instances are generated separately. From the Lipid, Protein and Disease instances, four types of relation pairs namely Lipid-Protein, Lipid-Disease, Protein-Protein, and Protein-Disease are extracted. For relation detection, we adopt a constraint-based association mining approach whereby two entities are said to be related if they co-occur in a sentence and satisfy a set of specified rules. This approach is detailed in [2].

The concept instances are instantiated to the respective ontology classes (as tagged by the gazetteer), the Object Property instances to the respective Object Properties and the Datatype property instances to the respective Datatype properties. This was automated using a custom script developed with the OWL programming framework, JENA API (http://jena.sourceforge.net/) for this purpose.

Slices. Given a number of query terms matching ontology terms from one ontology, the query graphs (slices) based on these terms are generated. For efficiency reasons our

algorithm generates multiple query graphs at the same time, thereby computing slices immediately. Slices can be represented by graphs as well: a slice can be represented by G_s=(N_s,E_s) where N_s is the set of all nodes in all query graphs in the slice and E_s is the set of all edges in all query graphs in the slice.

We have currently focused on slices based on concepts, i.e. all query terms represent concepts and not relations. Our algorithm, which is an extension of the algorithm proposed in [1], starts from the given concepts and traverses the ontology graph in a depth-first manner to find paths between the given concepts. These paths can be put together to find the slices.

Slice alignment. Our implemented algorithm computes an important sub-set of the aligned slice as defined in definition 5. As input we use two slices represented as graphs, the original ontologies as well as an alignment. The algorithm generates the shortest paths from the concepts in the first ontology on which the first slice is based, to concepts in the second ontology on which the second slice is based, via concepts in the alignment. This heuristic implements the intuition that the shorter paths represent closer relationships between the concepts than longer paths. For instance, in figure 2, possible shortest paths would be 6 - e16 - 4 - ea1 - E - e23 - B - e21 - A, 6 - e16 - 4 - ea1 - E - e23 - B - e24 - D, and 6 - e16 - 4 - e17 - 7 - ea2 - F - e26 - D. The original slices together with these shortest paths constitute our result.

Natural language query generation. The aligned slice is then translated into natural language (see [1] for details). For each query graph contained in the aligned slice we generate a natural language query for consumption by domain experts. The input to the natural language query generation (NLQG) sub-system are aligned slices which are represented by as set of triples. Each triple represents an edge and its end-nodes in the aligned slice: <ARG1, PREDICATE, ARG2> represents the concepts ARG1 and ARG2 that are related to each other by the relation PREDICATE. To translate a triple into natural language, we primarily use a template-based NLQG methodology. In this approach, the domain-specific knowledge and language-specific knowledge required for NLG are encoded as rule templates. The rule templates were generated using a rule learning algorithm. Given a triple, a content determination module recognizes the domain entities in the triples and extracts them for use as content terms. Upper level entities such as concepts and relations are identified and extracted directly via a rule template. For extracting the lower level entities, e.g. the verb and noun in an object property, we employ the BioText toolkit to perform part of speech tagging and term extraction. This is done as a preprocessing of the triples and the results are passed to a rule matching engine. The rule matching engine applies the best matching rule and retrieves a corresponding template to generate the natural language query. When two or more text components are generated they are aggregated to generate a compact query. We employ a set of aggregation patterns that are applied recursively to combine two or more queries sharing the same entity as a conjunction, as well as a generalized aggregation pattern that employs property hierarchy for combination. In the final step the query statement is checked after after sentence aggregation for grammar and generates a human understandable query. We employ an open source grammar checker, called LanguageTool (http://www.languagetool.org/), which is part of the OpenOffice suite. We added several rules to enrich the grammar verification checker.

Query result. After (aligned) slices are generated, in addition to being translated into natural language for consumption by end users, their graph triples are formulated into the corresponding syntax of the A-box query language (nRQL) of the reasoning engine RACER [7]. (For details we refer to [1].) nRQL is an A-box query language for the description logic $\mathcal{ALCQHI}_{\mathcal{R}^+}(\mathcal{D}^-)$. All queries written in natural language have a corresponding syntactic version that is issued to the knowledge base. A range of queries can be formulated based on the slices generated by user input. Complex queries are formulated based on multiple triples found in a graph and their connection is based on whether each set of domain and range in different predicates has similar properties. At least one triple and an optional set of domain and range in a role assertion query are necessary. In addition the specification of joins between multiple triples, representing conjunction of predicates, unknowns (variables) and constraints is necessary. For instance, the query graph in figure 4 represents the natural language query 'Which proteins interact with lipids that are implicated in a disease?'. The nRQL format of the query is:

```
(RETRIEVE (?X ?Y ?Z)
   (AND (?X Protein) (?Y Lipid) (?Z Disease)
        (?X ?Y Interacts_with)
        (?Y ?Z Implicated_in)))
```

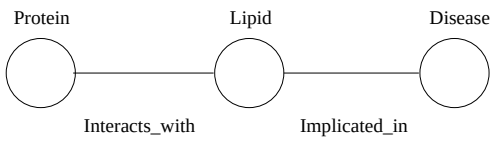

Fig. 4. Query graph example.

Our implemented system also allows queries to the knowledge base that involve instances. For instance, if we replace variable ?Z in the query above with the instance Ovarian Cancer, then this constrains the retrieval of all the instances of Protein to those that interact with a Lipid instance that is implicated in Ovarian Cancer.

6 Example scenario revisited

Given the Lipid ontology and our version of signal ontology, and the literature base as described in section 5, our system has instantiated a knowledge base. Upon a keyword query by the user for 'lipid', aligned slices are generated involving the lipid concept. The aligned query graphs in the slices are translated to natural language as well as to formal queries. The system then presents relevant queries involving lipid to the user. Examples of such queries are shown in figure 5. The user may learn about the ontological environment of lipid through the generated queries.

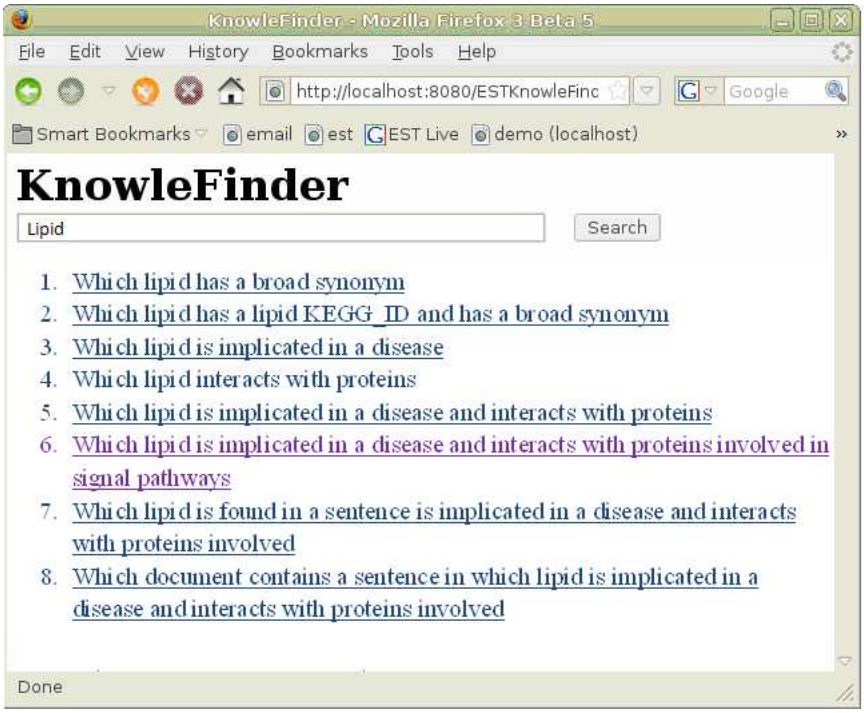

Fig. 5. Generated queries.

Question

NLG: Which lipid is implicated in a disease and interacts with proteins involved in signal pathways ?

```
nRQL:(RETRIEVE (?X ?Y ?Z ?W)
(AND (?X Protein) (?Y Lipid) (?Z Disease) (?W SignalPathway)
(?X ?Y Interacts_with)(?Y ?Z Implicated_in)(?X ?W Involved_in)))
```

Result

Protein	Lipid	Disease	Signal Pathway
P53	Unsat. Fatty Acid	Ovarian Cancer	Apoptosis

Fig. 6. Query and answer.

The user may be interested in the query 'Which lipid is implicated in a disease and interacts with proteins involved in signal pathways?' Running this query will result in a nRQL query to the knowledge base and an answer is returned: the lipid unsaturated fatty acids interacts with the protein p53, which is involved in apoptosis, and is implicated in Ovarian Cancer (see figure 6).

7 Related Work

We are not aware of any other work that fully deals with the problems of lack of knowledge of the domain and lack of knowledge of search technology. There are a number of systems that do tackle parts of these problems.

There exist very few systems that allow natural language querying. An example is askMEDLINE [6] that allows a user to pose a query in natural language and essentially uses the MeSH terms and other eligible terms in the natural language query to query PubMed using PubMed Entrez' E-Utilities. Although this helps the user with the lack of knowledge of available search technology, it does not alleviate the problem of lack of knowledge of the domain.

To aid the user in query formulation, [2] designed and deployed an interactive graphical query tool, Knowlegator, for the construction of queries using axioms provided in the ontology. Manipulation of these 'query atoms' from the OWL-DL ontology invokes A-box queries to a reasoner and subsequent ontology interrogation. The benefits of this paradigm include the ease of use and extensibility of query complexity far beyond the typical keyword searches and to a degree of query complexity suitable for domain experts who ask complex questions but who have limited agility with query syntax of various query languages. [10] extended this approach by taking advantage of transitive properties in populated ontologies to rebuild apoptosis pathways using protein entities mined from texts about apoptosis. A graph mining algorithm with graphical support for, (i) the selection of two pathway endpoints and (ii) rendering of pathways, was developed on top of the existing query tool, Knowlegator. It was also further customized to support bulk queries and rendering for all lipid-protein interactions relevant to a chosen pathway.

There are a number of systems that use ontologies to organize search results and allow a user to browse the literature via the ontology terms. For instance, GoPubMed [4] uses ontologies to index PubMed abstracts. Upon a keyword query, for each ontology term the number of PubMed abstracts containing the term or one of its descendants is computed. The results can then be browsed using the ontology. These systems alleviate the lack of knowledge of the domain problem, as the user can browse the results based on co-occurrence of the query term with ontology terms. However, it is still up to the user to decide whether this co-occurence is relevant or accidental. Also, these systems usually do not deal with multiple ontologies and their overlap.

There are a number of systems that use text mining and extract knowledge from documents based on ontologies. For instance, upon a keyword query, EBIMed [16] retrieves abstracts from Medline and finds sentences that contain biomedical terminology in the result. The terminology comes from public resources. The sentences and terminology are used to create overview tables representing associations between the

different terms. Textpresso [13] splits literature documents into sentences and words and labels them using ontology terms. The allowed queries are and/or combinations of keywords and labels.

Natural language generation technology is a mature technology dating back 10 years and is now being deployed in commercial settings, such as for providing query options to electronic health records [8]. Recently there have been initiatives aiming to produce textual summaries from Semantic Web ontologies. In the main they address how existing NLG tools can be adapted to take Semantic Web ontologies as their input. In their chapter [3] Bontcheva and Davis describe limitations of three such systems and highlight that quality of the generated text is highly dependent on the ontological constructs in the ontology and how their semantics is interpreted and rendered by the NLG system. Moreover before addressing knowledge transfer and NLG issues a re-assessment of appropriate metrics for evaluation may be required.

8 Conclusion

In this paper we have tackled the problems of lack of knowledge of available search technology and lack of knowledge of domain that users experience when they search for literature relevant to their task. We have proposed a framework that supports a search paradigm that uses (multiple) ontologies to generate relevant queries based on some keywords, translates these into natural language and allows a user via these natural language queries to query an instantiated knowledge base generated from the literature and the ontologies. We have defined the technical foundations and have described an implementation of the framework and its use.

There are still a number of issues that need further investigation. As our implemented algorithms do not compute the full slices or aligned slices, but use heuristics (e.g. shortest path for aligned slices), we want to investigate the influence of these as well as other heuristics. There is a trade-off between completeness (generating all possible queries) and information overload (showing all possible queries may not be instructive or may even be confusing for the user). Another interesting issue is whether it is possible to define a useful relevance measure for the generated queries, which could be used to rank the queries before showing them to the user. Further, as there is a connection between a slice generated from a set of keywords and a slice generated by a sub-set of this set of keywords, this connection could be used to optimize the process or to suggest the user possible interesting generalizations or specializations of the topic.

References

1. Wee Tiong Ang, Rajaraman Kanagasabai, and Christopher J. O. Baker. Knowledge translation: Computing the query potential of bioontologies. In *International Workshop on Semantic Web Applications and Tools for Life Sciences*, 2008.
2. Christopher J. O. Baker, Rajaraman Kanagasabai, Wee Tiong Ang, Anitha Veeramani, Hong Sang Low, and Markus R. Wenk. Towards ontology-driven navigation of the lipid bibliosphere. *BMC Bioinformatics*, 9(Suppl 1):S5, 2008.

3. Kalina Bontcheva and Brian Davis. Natural language generation from ontologies. In Davis, Grobelnik, and Mladenic, editors, *Semantic Knowledge Management, Integrating Ontology Management, Knowledge Discovery, and Human Language Technologies*, pages 113–127. Springer, 2008.
4. Andreas Doms and Michael Schroeder. GoPubMed - exploring PubMed with the gene ontology. *Nucleic Acids Research*, 33(Web server issue):W783–W786, 2005.
5. Jerome Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer, 2007.
6. Paul Fontelo, Fang Liu, and Michael Ackerman. askMEDLINE: a free-text, natural language query tool for MEDLINE/PubMed. *BMC Medical Informatics and Decision Making*, 5:5, 2005.
7. Volker Haarslev, Ralf Möller, and Michael Wessel. Querying the semantic web with Racer + nRQL. In *Proceedings of the International Workshop on Applications of Description Logics*.
8. Mary Dee Harris. Building a large-scale commercial nlg system for an EMR. In *Proceedings of the Fifth International Natural Language Generation Conference*, 2008.
9. Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.
10. Rajaraman Kanagasabai, Hong-Sang Low, Wee Tiong Ang, Markus R Wenk, and Christopher J. O. Baker. Ontology-centric navigation of pathway information mined from text. In *Knowledge in Biology - 11th Annual Bio-Ontologies Meeting co located with Intelligent Systems for Molecular Biology*, pages 1–4, 2008.
11. Patrick Lambrix and He Tan. SAMBO - a system for aligning and merging biomedical ontologies. *Journal of Web Semantics*, 4(3):196–206, 2006.
12. Patrick Lambrix, He Tan, Vaida Jakonienė, and Lena Strömbäck. Biological ontologies. In Baker and Cheung, editors, *Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences*, pages 85–99. Springer, 2007.
13. Hans-Michael Müller, Eimar E. Kenny, and Paul W. Sternberg. Textpresso: An ontology-based information retrieval and extraction system for biological literature. *PLoS Biology*, 2(11):e309, 2004.
14. Natalya F. Noy. Semantic integration: A survey of ontology-based approaches. *Sigmod Record*, 33(4):65–70, 2004.
15. NF Noy, N Griffith, and M Musen. Collecting community-based mappings in an ontology repository. In *Proceedings of the 7th International Semantic Web Conference*, pages 371–386, 2008.
16. Dietrich Rebholz-Schuhmann, Harald Kirsch, Miguel Arregui, Sylvain Gaudan, Mark Riethoven, and Peter Stoehr. EBIMed - text crunching to gather facts for proteins from Medline. *Bioinformatics*, 23:2237–e244, 2007.
17. Pavel Shvaiko and Jerome Euzenat. A survey of schema-based matching approaches. *Journal on Data Semantics*, IV:146–171, 2005.