Semantic keyword search for expert witness discovery

Siraya Sitthisarn, Lydia Lau and Peter M Dew
School of Computing
University of Leeds
Leeds, UK
e-mail: {scssi,L.M.S.Lau,P.M.Dew}@leeds.ac.uk

Abstract— **In the last few years, there has been an increase in the amount of information stored in semantically enriched knowledge bases, represented in RDF format. These improve the accuracy of search results when the queries are semantically formal. However framing such queries is inappropriate for inexperience users because they require specialist knowledge of ontology and syntax. In this paper, we explore an approach that automates the process of converting a conventional keyword search into a semantically formal query in order to find an expert on a semantically enriched knowledge base. A case study on expert witness discovery for the resolution of a legal dispute is chosen as the domain of interest and a system named SKengine is implemented to illustrate the approach. As well as providing an easy user interface, our experiment shows that SKengine can retrieve expert witness information with higher precision and higher recall, compared with the other system, with the same interface, implemented by a vector model approach.**

Keywords-component; semantic keyword search; ontology; expert witness discovery (key words)

I. INTRODUCTION

Communities often need to discover members with specialised experience and knowledge referred to as experts. Today directory services and online social networks, such as LinkedIn, enable members to discover experts online. However these sources describe experts using broad categories and shallow vocabularies and do not enable users to identify experts in a specified domain. Further they do not represent information about experts in a machine readable format within data models that are well defined and include the meaning of the data. Thus whenever a keyword search is undertaken search engines are not able to understand either the information content discovered online or the keywords within the user request. Such a search may have high recall but has limited precision. This results in many irrelevant experts being identified.

In contrast, using a semantic data model (specifically using the OWL ontology language) information about experts can be stored and recalled in a way that is more precise and complete. This is because the data model incorporates the meaning of the data represented in RDF. Unfortunately the formal semantic query languages (such as SPARQL) used for interrogating semantically enriched knowledge bases of this kind require the user to know the ontology and master its syntax [1, 2]. For this reason there is considerable interest in the development of semantic keyword searches to help end-users without technical expertise to access these resources.

This paper reports on a novel semantic keyword search system based on a semantically-enriched data model and a mechanism which converts keywords into semantic queries. The system is referred to as the "Semantic and Keyword interface engine" (SKengine). The discovery of expert witness is used as an illustrative case study. It demonstrates the design of an extensible semantic data model storing information about experts and the design decision used to translate a conventional keyword query into a set of SPARQL queries for information retrieval from this knowledge base. A scenario involving the online discovery of expert witnesses for a legal dispute about toy safety is developed for an experimental study. The results show that the SKengine significantly improves the precision and relevance of the answers to the queries.

The paper is structured in the following way. The next section introduces the scenario involving a legal dispute about toy safety. It shows the limitations of existing practice for the discovery of expert witnesses for a court case. It also outlines the envisioned system for discovering expert witnesses using the SKengine. Section III gives details of the ontology for describing information about expert witnesses. Section IV describes the SKengine for semantic keyword discovery and using the scenario to illustrate how it works in detail. The evaluation study is provided in section V which shows SKengine has both high recall and precision compared with the well known Vector Model widely used for information retrieval. The relevant background literature is given in the next section followed by the conclusions and recommendations for future work.

II. PROBLEMS IN CURRENT EXPERT WITNESS DESCOVERY

A. Characteristics of an Expert Witness

Expert witness is a particular class of expert characterized by the ability to provide an accurate analysis and report in a court of law on one or more specified subject areas connected with the issue under dispute[3, 4]. Opinions from an expert witness form an important part of the evidence presented to a jury. This paper focuses on the problem of on-line expert witness discovery. This case study was suggested by Professor Jeremy Barnett who is a practitioner specialising in this area of criminal law. He stated that "many legal professionals prefer

to use an expert witness who has both significant expertise and court experience in the specific dispute". Finding experts who meet these two criteria requires a richer data model than is provided by the current generation of expert witness directory systems such as LegalHub or professional network as LinkedIn (see below). These systems store information supplied by the expert witnesses, for example their CVs. However to discover an expert who has germane subject knowledge as well as relevant court experience it is necessary to search for information on the expert's background in the dispute issue as well as evidence of his/her expertise. Hence, a richer data model for expert witnesses needs to include information of various kinds. This includes the expert's experience of the dispute issue (for example of risk and hazards associated with essential safety requirements). The data model also needs information representing the person's credibility as an expert witness such as qualifications, career pathway, publications, research interests and court appearances. In this paper the Web Ontology Language (OWL) is used to specify the semantic data model for the discovery of suitable expert witnesses. By using semantics, new types of information can be added to the data model as and when identified.

B. *Current practice of expert witness discovery*

Traditionally, legal professionals locate expert witnesses by using paper-based directories and by exploring their professional networks or communities [4]. The growth of the internet has extended their search to include online resources such as online expert witness directories (e.g. www.legalhub.co.uk and www.expertsearch.co.uk). More recently, legal professionals also use general social /professional networking web sites (e.g. LinkedIn, www.linkedin.com), or those specific for legal professionals (e.g. LawLink, www.lawlink.com).

C. *Limitations in current keyword searches*

A scenario was used to illustrate the limitations of current online approaches to the discovery of expert witnesses for a legal dispute concerning toy safety. Consider a dispute between the trading standards regulator and the manufacturer of a remote-control toy helicopter called "Fly Dragonfly". The dispute is about whether the toy meets toy safety regulations [5]. It is alleged that the toy is a fire and burn hazard to children because the rechargeable battery inside the helicopter toys may overheat. David is a legal professional in the regulatory team who has been asked to assemble evidence for the hearing. David needs to identify a number of potential witnesses with experience relevant to the dispute. He turns to online legal directory services to supplement his personal knowledge. A natural search strategy to start with is to enter the entire phrase "overheated remote-control toy" in the search box.

In running through this scenario using Legal hub and LinkedIn, no results were returned for the phrase entered. Subsequent refinement and generalization of the search terms were needed in order to reduce or expand the number of search results. The results of these keyword searches are shown in Table I. On further analysis, the keyword terms "Overheated" "remote-control" "toy" in the phrase produced no information is because both Legal hub and LinkedIn are not directly concerned with toy safety standard disputes, so it is unlikely that these terms will exist in their datasets. More general terms like toy manufacturing or manufacturing did return a number of results but with poor precision. In other words the query returned a number of expert witnesses who were not suitable for the legal dispute about toy safety. For example one return was a manufacturing engineer in the medical device industry whose surname is "Toy". This is because these systems have no understanding of the context of the search query to eliminate "toy" as a "surname" is irrelevant.

TABLE I. RESULTS OF THE KEYWORD SEARCH

Keyword terms	Legal hub	LinkedIn
	no. of returns	*no. of returns*
Overheated remote-control toy	-	-
Remote-control toy	-	57
Toy manufacturing	1	2,503
Manufacturing	19	1,062,272

In addition the current approaches use standard directory information (such as name, contact, address, qualifications and expertise area). This is inadequate for a legal professional to tell if the expert witness has court experience directly related to the dispute case. Legal Hub does provide limited information about the court experience of the expert witness such as the number of times that he/she has given evidence in or written reports for court per year. However it does not provide details of the dispute subject nor dispute issues. Hence the legal professional cannot be sure that the experience of the expert witness meets their requirements.

In summary, a richer description of the expert witness is required which can be achieved using a semantic data model.

D. *Envisioned system*

This section presents an overview of the proposed solution.

To address the limitations discussed above a semantically enriched expert witness information knowledge base has been constructed. This integrates expert witness information from various sources including self-entry and information publicly available from legal expertise directory services.

The common approach used by semantic keyword search systems in the literature [8, 9, 10, 11] will be adopted for the solution and this is shown in Fig. 1. This is based on the idea of automatically generate and select a set of SPARQL queries derived from the keywords entered by a user.

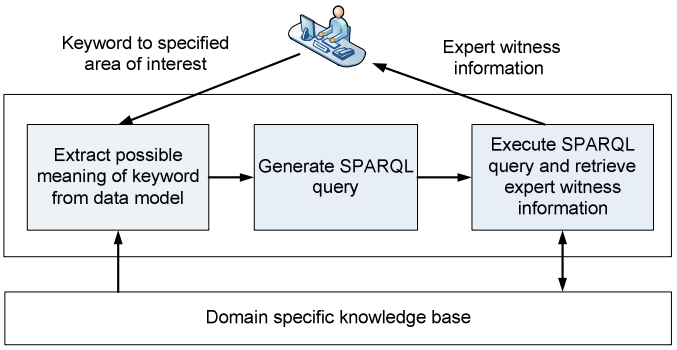

Figure 1. Proposed Semantic Keyword Search Approach

The SKengine is the proposed solution which provides a semantic keyword search interface to access a semantically-enriched knowledge base for the discovery of expert witnesses. The system will extract the possible meanings of the keywords from the domain specific knowledge base. It will then generate and select the SPARQL query that best fits the user requirements. Finally the generated SPARQL query is executed to retrieve the expert witness information from the knowledge base. The result of the semantic query is returned to the user.

The toy dispute domain in the above scenario was used to test the SKengine. The rest of this paper provides the explanation of how SKengine works for this scenario. The expert witness ontology for the toy dispute case study is given in the next section. This is then used in the Toy Dispute version of the SKengine to discover expertise witnesses in that domain.

III. THE SEMANTIC DATA MODEL FOR EXPERT WITNESS INFORMATION

Adding semantics to the data model is a first stage for the solution. The output of this process will form the "knowledge base" of the SKengine. OWL was used to describe the semantic data model. The choice of classes and their properties relevant to the domain of expert witnesses were informed by interviews with legal professionals on their current practice. This section explains the design of the expert witness ontology, and how it was verified by competency questions and legal professionals.

A. Design of the ontology

The expert witness ontology consists of two main groups of classes and properties: (i) expertise information and (ii) experience in the dispute case.

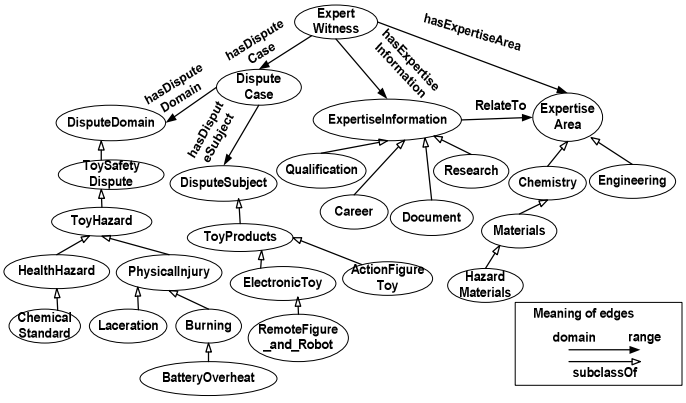

Figure 2. Relationship diagram of the expert witness ontology

As depicted in Fig. 2 the group of classes for expertise information starts with "ExpertisInformation" and "ExpertiseArea". This is used to capture information related to the expertise of the expert witness. The class "ExpertWitness" relates to the class "ExpertiseArea" by an object property *hasExpertiseArea*. Similarly, "ExpertWitness" and "ExpertiseInformation" are related by the object property *hasExpertiseInformation*; and "ExpertiseInformation" and "ExpertiseArea" are associated by the object property *RelateTo*.

The class "ExpertiseArea" includes subclasses such as "Engineering", "Chemistry" and so on. The class "ExpertiseInformation" currently has subclasses "Qualification", "Career", "Research" and "Document".

As depicted in Fig. 2, the group of classes for experience of expert witnesses in dispute cases starts with "DisputeCase". The class "ExpertWitness" relates to the class "DisputeCase" by an object property *hasDisputeCase* to denote that the expert witness has court experience in those cases. The class "DisputeCase" is also connected to the classes "DisputeSubject" and "DisputeDomain" by the object properties *hasDisputeSubject* and *hasDisputeDomain* respectively.

In our case study the class "ToyProducts" is a subclass of the "DisputeSubject" with further subclasses based on categories of toy products. As can be seen in Fig. 2 "ElectronicToys" and "RemoteFigure_and_Robot" are subclass of "ToyProducts". The subclass "ToySafetyDispute" includes possible disputes relating to issues such as toy hazards. The class "ToyHazard" includes "HealthHazard" and "PhysicalInjury" as its subclasses. "PhysicalInjury" includes a subclass "BatteryOverheat", representing the hazard of burning from an overheated battery, which is the subject of inquiry in the scenario.

B. Ontology fitness testing with competency questions

To ensure an ontology is fit for purpose, some literature [6, 7] have proposed the use of competency questions for testing. In our case, fit for purpose at this stage is about the ontology being able to return relevant answers to the keywords used in queries. These competency questions were designed to illustrate the behaviour of the SKEngine when the keywords match the classes defined in the ontology exactly or otherwise. These questions were also used to evaluate the information retrieval effectiveness of the SKengine (see detail given in section V).

A total of 50 competency questions were generated from six scenarios based on our investigation on the current practices, all in the toy safety domain. Table II shows four examples of competency questions generated from the scenarios. The full list of questions is available from the project Web Site: http://www.comp.leeds. ac.uk/scssi/SKengine.htm.

TABLE II. EXAMPLE OF COMPETENCY QUESTIONS

No.	Competency question	keywords
Q1	Who is an expert witness in a dispute case associated with a burning hazard, due to an overheated battery in a remote controlled toy?	Overheated remote-control toy
Q2	Who has a PhD in Engineering qualification?	PhD engineering
Q3	Who has expertise in hazardous materials?	Toxic materials
Q4	Who has experience in disputes involved with laceration hazards in any toy product?	Cutting hazard

These questions were used three times as the ontology was developed in an iterative manner. It was necessary to refine the

ontology twice as the competency questions showed some deficiencies in the first two versions. The ontology in Fig. 2 is the third version. It effectively retrieved relevant expert witness information associated with those questions.

C. User feedback on the semantic data model

We conducted a user evaluation to investigate the benefits of our semantic data model. Two legal professionals who are highly experienced in product safety and international trading participated in the evaluation. The use of SKengine for the scenario in section II was demonstrated to gauge their reaction on this proof-of-concept prototype. The main outcome of this evaluation was a confirmation that legal professionals need to be creative on what, how and where to search for suitable expert witness. To summarise feedback from the legal professionals:

- Their search criteria were not always predictable. Therefore, a richer data model with the possibility to add additional concepts would be useful.
- An option of checking the affiliation of an expert witness to any regulatory organisations or test centres will be useful.
- Filtering of irrelevant results was also mentioned as an important feature.
- There is a reliance on personal networks and experience to locate suitable expert witness.

These findings confirm the value of semantic data model which can provide (i) a flexible way of augmenting the data model with further concepts/classes (e.g. regulatory bodies); and (ii) richer meaning to the information which helps improve the precision of the results.

IV. SKENGINE ARCHITECTURE

An overview of the SKengine, adapted from the SPARK framework [9], is given in this section. Three important components, namely "entity mapping", "query graph construction" and "query graph ranking", will be discussed in detailed to illustrate the extension made by this paper.

A. An Overview of the Architecture

The SKengine architecture is shown in Fig. 3. At a high level, there are three main layers (i.e. knowledge base, pre-processing module, and formal query construction module). It shares the same generic components as the SPARK framework which also has been used by other researchers [8-11].

The *knowledge base* consists of the domain ontology (in OWL) and the associated data expressed in RDF. The design of the ontology is discussed in Section III. The content of this knowledge base will be used by the pre-processing module.

The *pre-processing* module had two types of indexes (i.e. entity index and ontology index). Entity index is constructed by using Lucene [20]. it will extract and index the entities from the RDF data files and the ontology. These entities will be used for keyword mapping later on. Ontology index will index relationships between the classes in the ontology. This will be used in the query graph construction below. The purpose of this pre-processing module is to speed up the performance during the generation of SPARQL queries.

The *formal query construction* module consists of 5 components: (i) entity mapping; (ii) query graph construction; (iii) query graph ranking; (iv) SPARQL query construction; and (v) SPARQL query execution. This module interacts directly with a user. After a user enters keywords in the search box the *entity mapping component* maps the keyword terms with the indexed knowledge base entities. The *query graph construction component* will then construct query graphs from the set of mapped entities. This component will produce all the possible query graphs by interpreting the meanings captured by semantics (e.g. a class, data and object properties, a literal). The *query graph ranking component* will rank and select the most relevant query graph that matches the keywords entered by the user. The selected query graph is forwarded to *SPARQL construction component* which will translate the query graph into a string conforming to the corresponding SPARQL syntax. The *SPARQL query execution component* will run the query and return the results to the user.

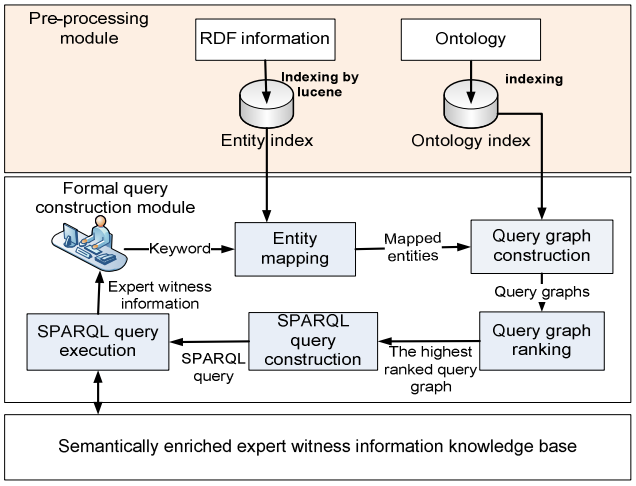

Figure 3. SKengine architecture

B. Entity mapping

Entity mapping is the first stage of formal query construction process. The SKengine uses Lucene search engine API [20] for the exact mapping between a keyword term and the indexed entities. If an exact match is not found, DISCO [21] (a software tool to retrieve the semantically similar words) will be used for mapping the indexed entities with similar terms to the keyword.

The input to this component is the "entity index". It is an inverted file of URI, label of entities (i.e. label of classes, label of object/data properties) as well as the literal of resources from every RDF data statements. For the scenario, when the user enters a search for "overheated remote-control toy" which consists of three keyword terms, "overheated", "remote-control" and "toy". Each keyword term is input to the Lucene search API and relevant entities from the index will be returned, ranked according to similarity to the keyword term.

The left hand side of Fig. 4 shows that the keyword term "overheated" is mapped to the class entity: BatteryOverheat,

and "remote-control" is mapped with class entity: RemoteFigure_and_Robot. The term "toy" is mapped with class entity: ToyProducts and literal entity: "marvel toy company". Note there is the term "toy" inside the literal.

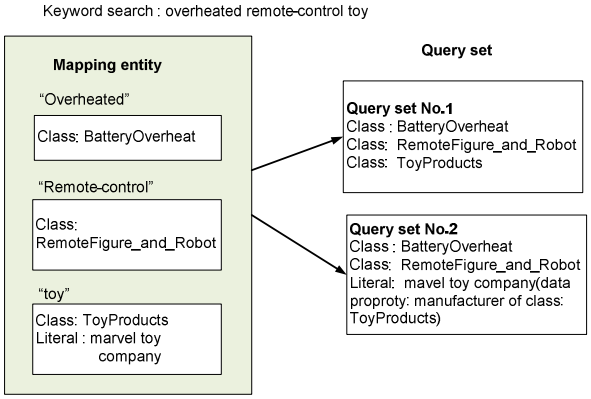

Figure 4. Enumeration of mapped entity to query sets

The right hand side of Fig.4 shows how the mapped entities are enumerated into query sets. The enumeration is an automatic process to produce the list of all possible combinations, one from each group of mapped entities. Each combination forms a query set. From the above example two queries sets, No.1 and 2, are identified. The entities in each query set will become the initial nodes for query graph construction. For example, the initial nodes for query set No.1 are illustrated in Fig. 5.

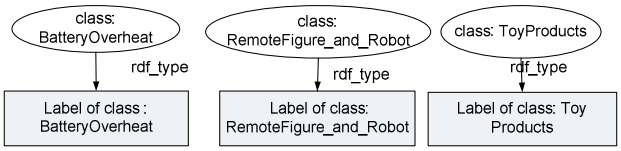

Figure 5. Initial nodes of query set No.1

C. *Fix-root query graph construction*

This stage is to create corresponding formal query graphs from the initial nodes of the identified query sets. Each query graph is represented by a tree (r, e_1,e_2,…e_n), where r is the root node designated for the answer to that query, e_n is a initial entity node corresponding to a keyword term, and n is the number of keyword terms [12]. In the scenario, the graph for query set No.1 is *(ExpertWitness(root), BatteryOverheat, RemoteFigure_and_Robot, ToyProducts)*.

The SKengine adapts the single level index search in the Blink's algorithm [12]. The algorithm produces a top-k scored query graphs with distinct roots (covering all possible nodes of answers to the search). However, the Blink's algorithm with all possible distinct answers is unnecessarily complex for expert witness discovery because the only required answer is an expert witness who has specific characteristics as specified in the keywords. Hence, the algorithm can be simplified by introducing the concept of a fixed root. We refer to this as the fix-root query graph construction.

The benefits of this approach are : (i) This reduces the cost of generating other types of root node with irrelevant classes for the answer. (ii) It has the potential of applying the algorithm to other targets (e.g. finding expert in another specified domain). This is because the algorithm only requires the system developers to specify the root node for their problem domain. While the algorithm is domain independence.

The basic principle is to convert all the query sets into query graphs. For each query set (e.g. see Fig. 5), the disjointed initial set of nodes (such as *BatteryOverheat, RemoteFigure_and_Robot, ToyProducts*) will be connected and traced right to the root node (i.e. ExpertWitness) step by step. Following is an explanation of the algorithm for achieving this.

The algorithm of fix-root query graph construction is shown in Fig. 6. Input to the algorithm is query set S= {initial $node_1$, initial $node_2$…initial $node_n$} where n = number of keyword terms; and r = the root node of interest (ExpertWitness). The output will be the query graph.

```
ConstructGraph(r, initial node1,…initial noden)
1 {
2   for i ∈[1,n] do
3       { nodei= new node (initial nodei)
4         nodei.flag = true }
5   for i ∈ [1,n] do
6       {  checkDisjoint(nodei)
7          checkRelation(nodei)}
8    while (checkEnd (∀ i∈[1,n]:  nodei)
9    {  for  i ∈ [1,n] do
10         {  if nodei.flag  = true
11          {nodei.new    = expansion(nodei)
12           nodei = nodei.i new
13           checkDisjoint(nodei )
14           checkRelation(nodei )} }}
15  }
checkDisjoint(nodei)
16  { for j ∈ [1,n] do
17     { if ((i ≠ j) and (nodei.flag= true) and (nodej.flag = true))
18     { if ((nodei.class = nodej.class) or  (superclass(nodei.class,nodej.class))
19     { nodei.merge = nodej
20      nodej.flag = false  }}}
21  }
checkRelation(nodei)
22  { for j ∈ [1,n] do
23    {  if ((i ≠ j) and (nodei.flag= true) and (nodej.flag = true))
      { if (have_relationship(nodei,nodej) )
24         nodei.merge = nodej
25         nodej.flag = false  }}
26  }
checkEnd((∀ i∈[1,n]:  nodei)
27  { if ( ∃ i ∈ [1,n]: nodei = r ) {nodei.flag = false}
28   if ( ∀ i ∈ [1,n]: nodei.flag=false) // nodes can be false by reaching the
29     {   checkEnd = false  }              root or  merging node
30    else {checkEnd = true}
31  return (checkEnd)
}
```

Figure 6. Query graph construction algorithm

The construction uses a bottom-up approach. There are two basic operations in the algorithm - node merging and node expansion. Merging between two nodes can take place when one of the following is true: (i) using the *function: checkDisjoint* to check if each pair of nodes has the same class

or if one is a super class of the other; and (ii) using the *function: checkRelation* to check if the pair of nodes have a object relationship with each other. Node expansion for a current node involves the location of next possible node(s) towards the root node using the ontology index (see Table III for an example: an obvious next possible node for the current node "BatteryOverheat" is "DisputeCase").

TABLE III. EXAMPLE OF ONTOLOGY INDEX ENTRIES

domain	object property	range
Class:DisputeCase	hasDisputeDomain	Class:BatteryOverheat
Class:DisputeCase	hasDisputeSubject	Class:ToyProducts
Class:ExpertWitness	hasDisputeCase	Class:DisputeCase

The construction process starts with placing all the initial nodes as the lowest level node (initial nodes) of the query graph. Perform node merging whenever possible. Where merging is not possible, perform node expansion. As see in Fig. 6 a node which cannot be merged with other nodes ($node_i$.flag = true) will then expand to new node ($node._{i.new}$). Generally, this is an iterative process with the aim to move up one level at a time towards the root node. There are complications during this process, for example, when more than one possible nodes are found after a node expansion. In this case, there are two possible criteria to determine which node to choose: (i) if it is the only node which can be merged with another node in the query tree constructed so far; (ii) if it has the short length to the root according to the ontology index. With this process, the nodes hopefully will eventually converge to the root node which is the condition to stop the iterative process. This is checked by *function:CheckEnd*. Otherwise, no results will be returned.

For the scenario, two query graphs are generated. Fig. 7 illustrates the query graph for query set No. 1.

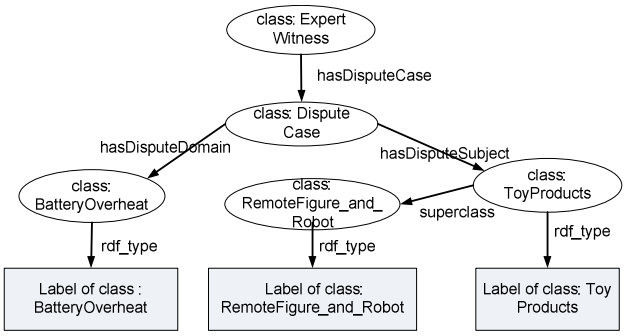

Figure 7. Query graph of query set No.1

D. Query graph ranking

The query graph construction may return more than one candidate query graphs. Therefore, a ranking process is necessary to compute the most relevant query graph to a user need. Three criteria are used for ranking the query graphs.

1) Association length: This criterion is commonly used for ranking semantic associations of graph as in [10, 13, 14]. On the assumption that users may be interested in corresponding keyword terms that are tightly associated (e.g. for plastic toy will usually refer to toy made of plastic). A shorter path between initial nodes to a merging node is given a higher ranking. Merge node is the node that links paths from initial node together. The path length and path length score can be calculated by (1) and (2) respectively.

Let *Rp* = the path length ranking score, p_i = path in the graph, i = the number of paths, e_i = the initial node in the $path(p_i)$ and $|e|$ = the number of initial nodes. Hence,

$$Path\ length = \frac{\sum_{p_i \in graph} \sum_{e_i \in p_i} distance(e_i, merging\ node)}{|e|} \quad (1)$$

$$Rp = 1/Path\ length \quad (2)$$

2) Entity mapping score: This criterion is based on traditional keyword based search engines to map terms of document with keywords. The higher the hitting score shows that the document is more relevant for the query [15]. The assumption of this criterion is "a query graph with a higher mapping score of initial node should be more relevant than the query graph with lower mapping score of initial node".

Let *Rm* = the average mapping score of query graph, e_i = the initial node in a query graph, and $|t|$ = number of keyword terms. Hence,

$$Rm = \frac{\sum_{p_i \in graph} \sum_{e_i \in p_i} mapping\ score(e_i, t)}{|t|} \quad (3)$$

3) Edge score: This criterion is based on the importance of an edge which can be defined as the number of times the associated property entity appears in the RDF data. The edge score can also be used for ranking [10]. A higher number indicates a higher importance of the edge.

$$Re = \frac{\sum_{pi \in graph} \sum_{edge \in pi} \frac{|r'(node_i, node_j)|}{|r|}}{|edge|} \quad (4)$$

In (4), *Re* = the average important edge score, when property $\acute{r}$ = an edge that connects $node_i$ and $node_j$. The $node_i$ = domain and $node_j$ = range of property. $|r|$ is number of property entities in RDF data, following type of property $\acute{r}$. If $\acute{r}$ is data property, $|r|$ = number of all data property, if $\acute{r}$ is rdf_type, $|r|$ = number of instance of class, and if $\acute{r}$ is object property, $|r|$ = number of all object property entity in RDF data.

In above equations (2), (3) and (4), we have defined 3 ranking criteria. We now define the overall association rank using these criteria as:

$$Rgraph = Rp * Rm * Re \quad (5)$$

In (5), the weight of each criterion is 1. However there is a potential to adapt weights for each criterion in future work.

After ranking all possible candidate query graphs the graph with highest ranking score is translated into SPARQL formal query. Finally, SPARQL is executed and presents expert witness information back to a user.

V. EVALUATION

The objective was to evaluate the SKengine against the Vector Model using the competency questions given in section IV. The Vector Model was used for this comparison because

it's widely used in information retrieval [15]. It has also been studied in the context of semantic searches [19].

The used metrics were precision and recall. Precision = $\frac{|Ra|}{|A|}$ and Recall = $\frac{|Ra|}{|R|}$ where |R| is the number of relevant expert witnesses that relate to the keyword query; |A| is the number retrieved; and | Ra | is the number retrieved that are relevant to the user requirement.

A. Experiment setup

The set of 50 competency questions, given in section III, were used in the comparison. We note that using manually generated SPARQL queries 100% recall and precision was achieved for each competency question. This result was used in comparing the performance of the Vector Model and the SKengine.

To decrease keyword selection bias the competency questions were divided into two groups of 25 questions. These were keyword type A, those that exactly mapped onto an entity in the semantic model and keywords used for Q1-Q2, in table II are example of keyword type A; and type B, those that couldn't be exactly mapped onto any entity in the semantic model. For these DISCO was used to find semantically similar words, keywords used to represent Q3-Q4 are example of keyword type B.

B. Experimental results

The results concerning average precision and recall from searches using these different approaches are summarised in Table IV. They show that compared with the manual approach the average precision of the SKengine is 89.19% while the Vector Method only achieved 35.87%. Both had high recall. We conclude that the SKengine meets our objective of significantly improving the precision while achieving high recall.

TABLE IV. THE INFORMATION RETRIEVAL EFFECTIVENESS OF 3 APPROACHES

Search approach	Average precision	Average recall
Manual SPARQL query	100	100
SKengine	89.19	90.00
Vector Model	35.87	83.28

The result in Table V shows that for both types A and B the SKengine performs very well. The use of DISCO to find semantically similar words is an important factor in improving the precision of the response to type B queries.

TABLE V. PERCENTAGE OF EXECUTING ACCEPTABLE SPARQL QUERIES IN EACH KEYWORD TYPE

Keyword type	No. questions	No. acceptably executed SPARQL queries	% of executing acceptable SPARQL queries	Average precision	Average recall
Type A	25	21	84	92.32	96.80
Type B	25	17	68	86.03	83.71
Total	50	38	76	89.18	90.00

However there are limitations with the SKengine that merit further investigation. For example if a user wants to discover an expert witness who has experience in a legal dispute about a toy doll made of plastic. Suppose the user inputs the keywords "doll plastic material" to the SKengine. It would return the expert witnesses who have expertise in the area of "plastic materials" and in dispute cases concerning "doll from any material". This is an example where the SKengine returns an ambiguous answer where the meaning of the query is context dependent. Clearly there is a loss of precision which needs further investigation.

VI. RELATED WORK

In recent years many papers have been published concerning the use of the OWL/RDF semantic data models to increase the effectiveness of information retrieval in various domains [16-18]. The challenge now is to make semantic searches easier for non-specialists by either using a familiar keyword interface as used in traditional web searches or by using some form of natural language interface. For our problem the use of a semantic search with a keyword interface helps the typical user access semantically enriched knowledge bases without having to learn new technologies to do so.

A number of useful papers have been published on semantic keyword search techniques [8-11, 13]. Semsearch [8] proposes using predefined query templates to construct formal queries. The templates are a combination of all possible entity types from mapped entities. At runtime each keyword term is mapped onto an entity. All mapped entities then are matched to the template to construct the formal query. This approach is appropriate when there are just 1-2 keyword terms but when there are more, a large number of complex patterns would need to be defined. In contrast, the SKengine proposed in this paper provides more flexible query graph construction. It does not fix the structure of the query in the form of templates.

In SPARK[9] a query graph construction applies Kruskal's minimum spanning tree algorithm. The query graph is constructed by using the algorithm to explore the RDF graph and discover the appropriate connecting nodes. These link together the mapped entities. [11] proposes another approach for interpreting a keyword query using a semantic knowledge base. The approach uses a traversal graph algorithm to construct a query graph. It does this by traversing an RDF knowledge base finding the neighbouring entities of each mapped entity within a limited range. After that the possible sub graphs which connect the mapped entities are extracted from the whole query graph. The main drawback with both SPARK and [11] is computational cost. Their query graph construction algorithms require the exploration of the entire RDF data graph. In contrast, the SKengine algorithm is faster and more efficient because it explores its predefined ontology index rather than the RDF data graph. The ontology index collects only ontology structures. The index is much smaller than the whole RDF graph so explorations and graph expansions are more limited and hence faster.

Q2semantic[10] is different from the above approach. It proposes using keyword searches on schemaless RDF data graphs or those that do not include an ontology. Q2semantic uses an RDF graph clustering technique to infer an ontology

structure. This inference enables the algorithm to generate top-k query graphs by exploring the ontology structure which is a limited data space. Such an RDF graph clustering technique is unnecessary for ontology based applications like the SKengine. This is because it contains an explicit ontology schema to support exploration. Q2semantic's query graph construction algorithm adopts a single-level search algorithm with distinct root nodes discussed in Blink[12]. Its ranking approach is used to generate the top-k of the query graphs. The SKengine algorithm is similar to Q2semantic in that it is also adapted from the Blink algorithm. However the SKengine is simplified by restricting the query graphs to those with fixed roots. In the case study the fixed root is the ExpertWitness node. This query graph restriction is sufficient to answer all queries associated with the discovery of expert witnesses. The algorithm avoids generating irrelevant roots not involved with expert witness discovery.

VII. CONCLUSION AND FUTURE WORK

This paper is concerned with the problem of automated discovery of expert witnesses using a semantically enriched knowledge base with a keyword interface. The SKengine has been used to investigate the effectiveness of the semantic keyword search for expert witness discovery. The initial results are promising and show that this kind of keyword search has high recall and high precision. It is clearly an advance on the Vector Method. The competence tests give confidence that the results are meaningful. Further research is required to address the problem of ambiguous answers where the meaning of the query is context dependent.

The SKengine improves the query graph construction algorithm for applications that can take advantage of the fixed root. Expert witness discovery is one example. This fixed root algorithm can be easily adapted for finding experts in other domains (e.g. academic or medical fields).

An important next step is to evaluate the SKengine against other semantic keyword methods to quantify the benefits of using the fixed root query graph construction algorithm. Further an end-user study is needed to evaluate the SKengine with legal professionals involved in disputes such as those associated with toy safety. Another area of research is to add an expert witness reputation function to the SKengine. One way of establishing an expert witness's reputation would be to collect users' ratings of his/her performance as an expert witness. Over time the user rating results would feed into a reputation function. This could help in selecting the most suitable expert witness for a particular dispute. Finally research is needed to integrate the SKengine into a semantic social network such as FOAF.

ACKNOWLEDGEMENT

We are grateful to Professor Jeremy Barnett for providing a user perspective on the research and for providing information on toy safety disputes from his personal experience; and to Dr Michael Shen for his input on the evaluation study.

REFERENCE

[1] D. Strasunskas and S.L. Tomassen, "A Role of Ontology in Enhanceing semantic search: the EvOQS Framework and its Initial Validation," International journal of Knowledge and Learning, vol. 4, no. 4, 2009, pp. 398-414.

[2] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF" 2008; http://www.w3.org/TR/rdf-sparql-query/.

[3] K. A.Staton, "Discovery of Attorney Work Product Reviewed by Expert Witness," Columbia Law Review, vol. 85, no. 4, 1985, pp. 812-836.

[4] M. J.Saks, "Expert Witnesses, Nonexpert Witnesses and Nonwitness Experts," Law and Human Behavior, vol. 14, no. 4, 1990, pp. 291-313.

[5] Department of Trade and Industry, "Product standard for toy Safety (URN 99/1019) in Guidance notes on the UK toys (safety) regulations 1995 (S.I.1995/204)", 2000.

[6] M. Gruninger and M.S. Fox, "Methodology for the Design and Evaluation of Ontologies," Proc. Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, 1995.

[7] N.F. Noy and D.L. McGuinness, Ontology Development 101: A Guide to Creating Your First Ontology Stanford KSL Technical Report KSL-01-05, 2000.

[8] Y. Lei, V. Uren and E. Motta, "Semsearch : A seach engine for semantic web," Proc. Proceeding of the 15th International conference on Knowledge Engineering and Knowledge management(EKAW), 2006, pp. 238-245.

[9] Q. Zhou, C. Wang, M. Xiong, H. Wang and Y. Yu, "SPARK:Adapting Keyword Query to Semantic Search," Proc. the 6th International Semantic web conference, 2007, pp. 694-707.

[10] H. Wang, K.Zhange, Q. Liu, T. Tran and Y. Yu, "Q2Semantic: A Lightweight Keyword Interface to Semantic Search,", Proc. the 5th European semantic web conference on the semantic web: research and applications,2008 , pp. 584-598.

[11] T. Tran, P. Cimiano, S. Rudolph and R. Studer, "Ontology-based Interpretation of Keywords for Semantic Search," Proc. the 6th International Semantic web conference, 2007, pp. 523-536.

[12] H. He, J. Yang and P.S. Yu, "BLINKS: Ranked Keyword Searches on Graphs," Proc. International conference on management of data, 2007, pp. 305-316.

[13] T. Tran, H. Wang, S. Rudolph and P. Cimiano, "Top-k Exploration of Query Graph Candidates for Efficient Keyword Search on RDF," Proc. The 25th International Conference on Data Engineering (ICDE'09), 2009.

[14] B. Aleman-Meza, C. Halaschek-Wiener, I.B. Arpinar, C. Ramakrishnan and A. Sheth, "Ranging Complex Relationships on the Semantic Web," IEEE Internet Computing, vol. 9, no. 3, 2005, pp. 37-44.

[15] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information retrieval, Addison Wesley, 1999.

[16] F. Ortiz-Rodriguez, "EGODO and Application: sharing, retrieving and exchanging legel document across e-Goverment," Proc. Semantic Web for Law workshop, 2007.

[17] P. Liu and P. Dew, "Using Semantic Web Technologies to improve expertise matching within Academia," Proc. I-KNOW '04, 2004.

[18] J. Li, et al., "Expert Finding for ecollaboration Using FOAF with RuleML Rules," Proc. The 2006 Conference on eTechnologies, 2006, pp. 53-65.

[19] P.Castells, M. Fernandez, and D. Vallet, "An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval," IEEE Transactions on Knowledge and Data Engineering. vol.19, no.2, 2007, pp. 261-272.

[20] http://lucene.apache.org/java/docs/

[21] http://linguatools.de/disco/disco_en.html