Using OWL Contexts and OQE to Improve Web Document Search Precision and Recall

David George [1], **Zimin Wu** [2]
[1, 2] School of Computing, Engineering and Physical Sciences, University of Central Lancashire, Preston, Lancashire, UK
[1] email: dgeorge@uclan.ac.uk

Abstract

Ontologies describe domains and integrate information semantically. This paper examines semantic search, using ontology modules (contexts) and an independent corpus of Web documents, to see how ontology-based query expansion (*OQE*) might improve information retrieval precision and recall (P&R) compared to traditional keyword-only query; by giving added weight to documents that contain terms relevant to a query context ontology and also by returning relevant Web documents containing none of the original search keywords.

To assess search effectiveness, a semantic search tool (*SemSeT*) has been developed to conduct *OQE* experiments; by matching OWL file concepts with document text, without resorting to structured semantic query languages or reasoning to automatically refine queries. A vector space model has been used to calculate document relevance scores and the results show that, compared to basic keyword query, *OQE* can markedly improve P&R.

Keywords: Information Retrieval; Ontology Context; Ontology-based Query Expansion; Precision and Recall, Semantic Search.

1. Introduction

Traditional Web search and information retrieval (IR) rely on matching keywords/phrases with indexed documents to generate a ranked list of potentially relevant documents, optimised in terms of search effectiveness - typically *precision* and *recall* (P&R) [30]. However, traditional IR is unlikely to return documents having semantically related terms, if they contain none of the query terms. Therefore, rather than relying solely on such IR approaches, could Semantic Web [3] ontologies help search tools retrieve relevant documents more effectively by giving greater weight to documents having wider contextual relevance to queries? Whilst there are ontology based search examples, a review of Semantic Web near-term prospects [2] and examination of use cases [34], added to existing commercial search engine approaches, does not suggest imminent or significant semantic search breakthrough; and yet ontology expressivity seems to offer much for query expansion.

The role of ontology [6, 11-13] in the Semantic Web is to formally describe a domain of interest, facilitate reusability [19], and semantically link and integrate information by overcoming structural and semantic heterogeneity. A Semantic Web lift-off requires a critical mass of RDF/OWL [32, 33].

Ontologies have featured in various academic search initiatives:

i. crawler-based locators of RDF and ontology resources, e.g. Swoogle [7] and Sindice [20]; search support in specialist knowledge domains, e.g. bioinformatics and the Gene Ontology [1, 27];
ii. international organisation support, e.g. World Bank and Organisation for Economic Co-operation and Development (OECD) [15] and in legal document search [4], where ontology query uses technical terms to find related information, terms and documents;
iii. other research projects involving ontology-based query expansion [5] to exploit ontological relations [8, 16] and semantic network-based sense definitions [18], where ordinary keyword terms drive ontology traversal, using reasoning-based semantic query languages for query expansion concepts. Related lexical work has included [31].

Commercial semantic search has included natural language processing search companies Hakia [14] and Powerset [21].

This research has similarities with research in (iii), except that it will measure the benefits of ontology-based query expansion (*OQE*) without resorting to semantic reasoning-based structured query languages. A semantic search tool (*SemSeT*) has been developed to show that *OQE* can enhance P&R compared to purely keyword-driven search. The research will show how small ontology modules (contexts) can support users

and *OQE* and can return relevant Web documents containing none of the user's original search terms, e.g. if query terms *Europe* and *transport company* were matched only in an ontology, SemSeT could search for semantically related concepts via the class hierarchy from *TransportCompany* to find *North_West_Trains* and from *Europe* via classes and relations to find *Manchester* and *England* - as in Fig. 1.

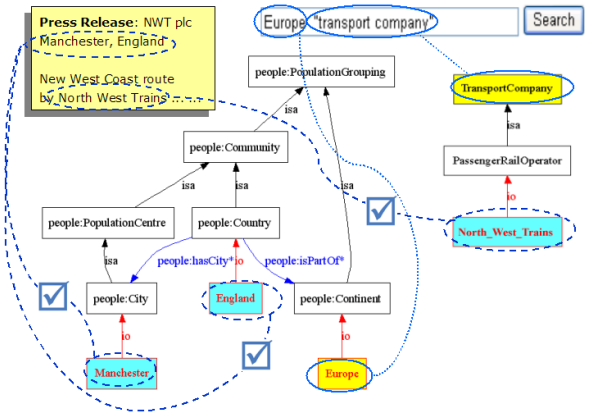

Fig. 1. SemSeT *OQE* finds a related document.

1.1. Research Objectives and Tasks

The primary objective was to develop SemSeT, using a classic document relevance algorithm and generating P&R measures, to compare *OQE* search effectiveness against keyword-only search. A query tool for both keyword and *OQE* was necessary because ontologies would be incorporated within a query process and meaningful document relevance comparisons with a search engine would be difficult given the effect of page indexing and relevance algorithms, e.g., the PageRank link analysis [10]. A further objective was to develop small self-standing ontology modules for use in both *OQE* and the user's selection of a query relevant ontology context and terms. The tool also required ontology traversal algorithms and concept weightings to reflect semantic distance between base query terms and *OQE* terms.

1.2. IR and Document Relevance Ranking

IR techniques present results ranked by order of relevance using various algorithms, e.g., the Vector Space Model (VSM) [24], an algebraic model for representing a document's collection of terms (vector) as a scalar value to reflect similarity to a query vector, and Probabilistic Models [26], where similarities are calculated by probabilistic inference that term distribution will be different in relevant and non-relevant documents. For this initial research, comparing the use of keywords against *OQE*, the VSM relevance measure was selected.

The VSM has been extended by a classic *tf-idf* measure [25] for term-weighting, where a term's importance is increased by its frequency in the document but is reduced by the frequency of the term in the document corpus. To rank a document's relevance, the *tf-idf* measure calculates the sum of a document's term weights: where *tf* represents the frequency *F* of any term *t* in document *d* (i.e., F_{td}) and *idf* is the inverse document frequency calculated by the *log* of the number of documents *D* in a corpus, divided by the number of documents *n* containing term *t*. A *term* weight vector W_{td}, e.g. for term t_i is then:

$$W_{td} = \sum_{t_i \in d, d \in D} F_{t_i d} * \ln\left(\frac{D}{n_{t_i}}\right)$$

To minimise exaggerated weightings when documents contain excessively repeated terms, the frequency F_{td} for each term can be normalised by dividing it by the highest term frequency maxF_{td} found in the document:

$$W_{td} = \sum_{t_i \in d, d \in D} \left(\frac{F_{t_i d}}{\max F_{td}}\right) * \ln\left(\frac{D}{n_{t_i}}\right)$$

The document weight vector (W_d) is then the sum of all matched term weights:

$$W_d = \sum_{t_i, , , t_n \in d} [\, W_{t_i d}, , , W_{t_n d} \,]$$

The *tf-idf* result is a weighted statistical measure of a term's importance to a document and can be used in P&R measures to determine search effectiveness.

1.3. Search Effectiveness

The IR community has traditionally evaluated search effectiveness using the set-based P&R measures [30]:

$$P = \frac{|\ relevant\ documents \cap documents\ retrieved\ |}{|\ documents\ retrieved\ |}$$

$$R = \frac{|\ relevant\ documents \cap documents\ retrieved\ |}{|\ relevant\ documents\ |}$$

P represents the number of relevant documents in a returned set and *R* is the number of relevant documents returned from a relevant set.

1.4. *Modularisation of Ontology Concepts*

Small, self-standing ontologies can be developed by specifying concepts in an equally modular way, e.g. by adopting Rector's best practice approach [22]. Rector advocates that self-standing *primitives*, *roles* and *relations* are combined in asserted conditions to form more complex *defined* classes, to avoid the often hidden, inexplicit meaning in a concept's name.

Given the use of search keywords, primitive (stand-alone/atomic) classes would seem consistent with the notion of such query terms and with *OQE* and document text matching.

1.5. *Organisation of the paper*

The paper is organised as follows: section 2 discusses the SemSeT process and methods, search topic ontology contexts, and queries for P&R comparisons; section 3 provides pseudo code for algorithms considered in section 2, i.e. ontology traversal for *OQE*; section 4 presents *OQE* results and discusses their significance; section 5 presents conclusions.

2. Experiment Design and Development

This section considers the design and development of SemSeT for IR experiments made on the independent TREC WT2g [29] ¼ million Web document corpus; which has known relevance outcomes. SemSeT uses various ontology traversal algorithms to identify query term matching/related *OQE* terms and calculates P&R measures from document relevance scores generated by a modified *tf-idf* algorithm. The OWL ontology representation language [32] was used to specify ontology contexts.

2.1. *The SemSeT Interface*

SemSeT is a prototype *keyword* and *ontology*-based search tool developed with the Jena Ontology API [17]. It offers a controlled environment for testing OWL query algorithms and ranking documents for P&R comparisons. The interface (Fig. 2.) has three components:

- query setup in panel [i];
- query mode, query term selection and VSM *tf-idf* scoring feedback in [ii];
- ranked document results output in [iii].

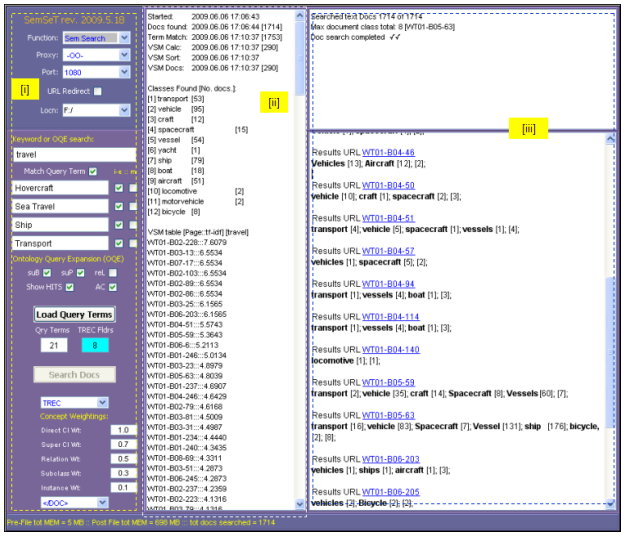

Fig. 2. SemSeT's interface.

Query setup [i] requires a search mode, i.e. keyword or ontology context, and up to 4 query terms. Input boxes use an adaptive text process, e.g. entering "t" reduces an ontology context list to tourism and travel and "tr" selects travel. The system then displays all travel classes in [ii] and adaptive text is used to find *OQE* query terms. If a matching ontology term is not found, the user's input is accepted. Given a base term set, e.g. Hovercraft, Sea Travel, Ship and Transport, the base terms plus sub, super or equivalent class *OQE* set is generated in [ii], as shown in Fig. 3.

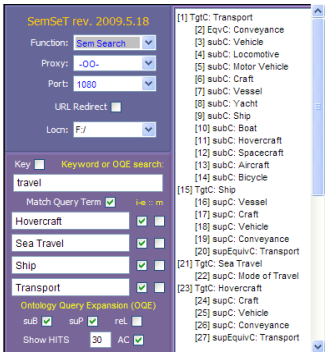

Fig. 3. Base query terms *OQE* set.

Various *OQE* options can be chosen, e.g. *sub* and *super* class or *sub* and *super* and *relation* class *OQE*. Document hits and rankings are output in [ii] and [iii], as shown in Fig. 2 above.

2.2. *Search Process and Methods*

The SemSeT process, see Fig. 4, involves keyword /context selection (A), keyword entry (B), ontology traversal, term weighting and query term set generation (C), document text

analysis using pattern matching and a regular expression (D), term weight allocation, VSM *tfidf* document relevance algorithm for P&R (E). The process only differs in query term set generation stage (C), based on the search mode required.

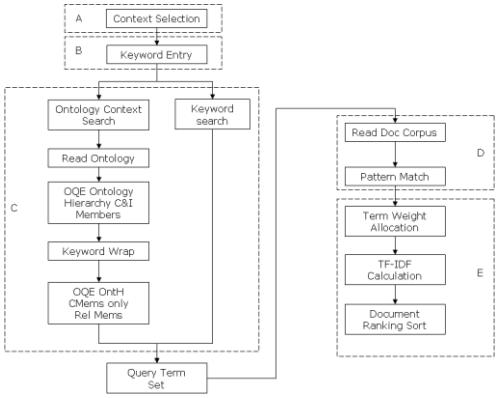

Fig. 4. Schematic of SemSeT process.

2.2.1. *Term Pattern Matching and Validation*

Query testing was conducted on a small TREC document set so that text pattern matches could be validated. The *regular expression* was refined until reliable query term matches and term counts were returned, e.g. ensuring ship, ships, ship's or –ship, are counted as one term.

2.2.2. *Ontology Traversal for OQE*

The *OQE* process creates a query term set for each matched query term, by traversing the ontology and adding sub, super, equivalent and relation classes to the set, subject to the *OQE* option required, i.e. the query term plus related sub and super classes (*S+S OQE*); or *S+S OQE* plus relation classes (*S+S+R OQE*), as in 2.2.3; or *all* ontology classes (*All OQE*).

2.2.3. *Class and Relationship Weighting*

A weight can be applied to a query expansion concept based on its semantic distance from a query term [5, 8, 9, 23, 28].

As the weighting issue was unclear during early experimentation, it was decided to initially test SemSeT's ranking algorithm using weights similar to [8]. Fig. 5 shows how weight O_w might reflect an ontology concept's hierarchy position for *OQE*, e.g. the concept Ship is a super class of keyword matching class CargoShip and could rank lower (0.7) than the direct match CargoShip (1.0) but higher than CargoShip's sub class Tanker (0.3). The rationale was that a super class could be weighted above a sub class as a CargoShip is always a Ship but not necessarily a Tanker.

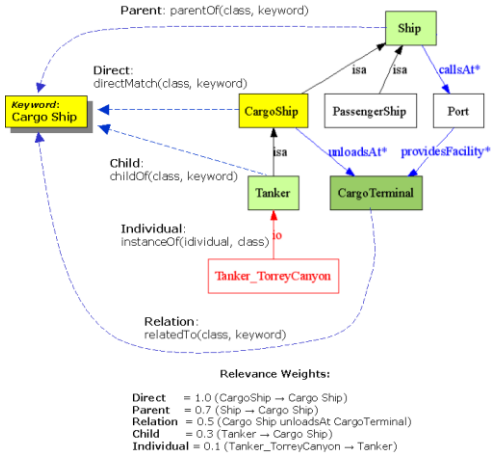

Fig. 5. Semantic distance weights.

Other weightings can apply, e.g. *relation* class CargoTerminal has no inheritance relationship with CargoShip but does have an asserted condition *relation* CargoShip *unloadsAt* CargoTerminal; it was weighted at 0.5 and individual Tanker_TorreyCanyon weighted 0.1.

The document weight W^{+}_{d} is then modified by multiplying O_w with term weight vector F_{td}:

$$W_{td} = ((F_{td} * O_w) / maxF_{td}) * ln(D / n_t)$$

Different weightings were tested after the main query experimentation and their P&R impact is discussed in the results section.

2.3. *Ontology Contexts Developed for OQE*

The ontology context and *OQE* experiments were based on the *narrative* given for selected TREC query topics: TREC 401 "Foreign minorities, Germany" (T401) and TREC 416 "Three Gorges Project" (T416); otherwise, ontologies were specified independently, i.e. without examining the TREC corpus.

2.3.1. *T401 "Foreign minorities, Germany"*

The query description for T401 was "What language and cultural differences impede the integration of foreign minorities in Germany?" The Border Agency site and glossary of terms, http://www.bia.homeoffice.gov.uk/, were used to identify *immigration* concepts and an ontology extract is shown in Fig. 6; it has a fairly shallow class hierarchy supporting 41 query terms, which limits *S+S OQE* potential.

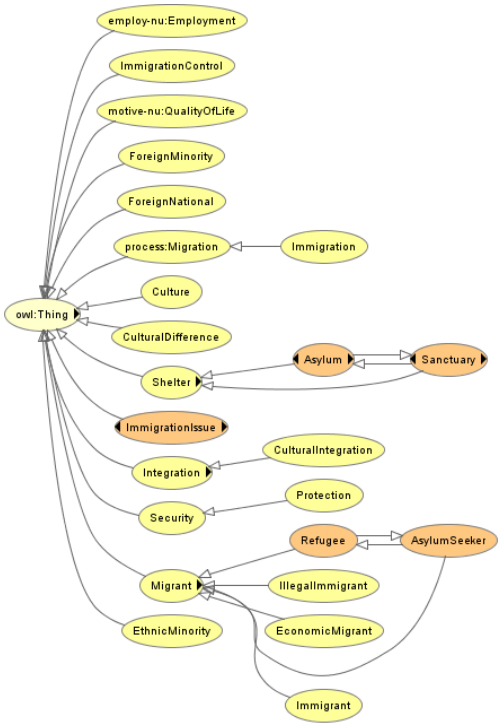

Fig. 6. T401 Immigration ontology.

2.3.2. *T416 "Three Gorges Project"*

The query description for T416 was "What is the status of The Three Gorges Project?" Primary information sources for a *hydro-electric* ontology, see extract in Fig. 7, were the British Dam Society http://www.britishdams.org/ and Wikipedia's Three Gorges Dam content http://en.wikipedia.org/wiki/Three_Gorges_Dam.

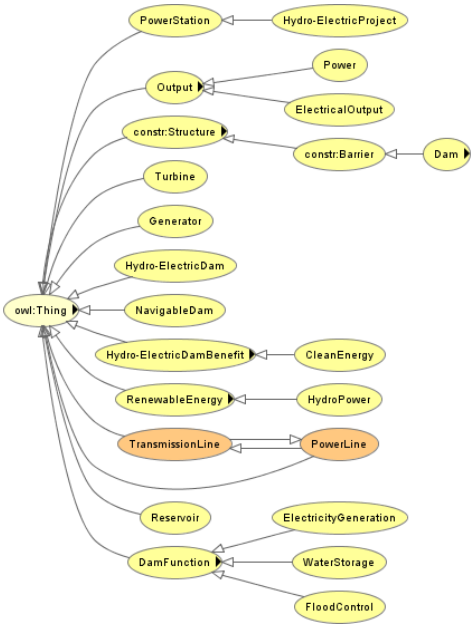

Fig. 7. T416 Hydro-electric ontology.

As the ontology supports 58 query terms in an equally shallow hierarchy, *relation* classes were specified using asserted conditions, see Fig. 8, for *S+S+R OQE* query mode (2.2.2, 2.2.3).

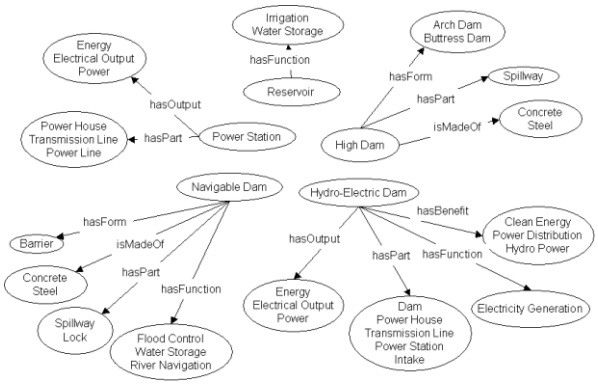

Fig. 8. Hydro-electric class relations.

2.3.3. *Query Matrix and P&R Comparisons*

Both T401 and T416 keyword (*K*) to *OQE* comparisons use a matrix of 10 query sets, providing a spread of ontology classes to satisfy the TREC topic query guidelines. The data from 50 queries allowed P&R graph comparisons to be made by plotting cumulative document precision at 10% recall intervals.

T401's matrix used every ontology term (41), as only limited *S+S OQE* was possible. Query sets 1-6 compared *K* against *All OQE* and sets 7-10 compared *K* against *S+S OQE*.

To examine T416's *S+S OQE* limitations, approximately 40 ontology terms were used in two *OQE* approaches; by comparing *K* against *S+S OQE* and then against *S+S+R OQE*.

3. Ontology Traversal Algorithms

Simplified algorithms are shown for *class hierarchy* and *relation class OQE*. The appendix has more complete Jena API-based versions.

3.1. *OQE Algorithm – Class Hierarchy*

Super class *OQE* adds direct super class line classes, i.e. it excludes sub class branches.

```
for each ontology class c {
  if c subclass c_sub or c superclass c^sup OQE required {
    for each keyword {
      if c equals keyword {
        if c^sup required {do c^supProc. }
        if c_sub required {do c_subProc and do c_subindividualProc. }
        do cProc. }
      if cProc {
        add c to OQE array.
        for c  list equivalent classes c^eq {
          add c^eq to OQE array. } }
```

```
    if do c^sup Proc AND c has c^sup {
      for c list c^sup {
        set Top class equal to next c^sup.
        do c^sup individualProc.
        add c^sup to OQE array. }
      if do c^sup individualProc {
       for Top class list individuals {
        add individuals to OQE array. } }
    }
    if do c_sub Proc AND c has c_sub {
      for c list c_sub { add c_sub to OQE array. } }
    if do c_sub individualProc AND (c^sup individualProc NOT executed) {
      for c list individuals { add individual to OQE array. } }
 } }
 else if (c_sub AND c^sup OQE) NOT required {
   add c to OQE array. } // Get All classes
}

if (c_sub AND c^sup OQE) NOT required {
  for each ontology class list individuals {
    add individual to OQE array. } } // Get All individuals
```

3.2. OQE Algorithm – Relation Classes

Relation classes are only generated for classes found by the *class hierarchy* algorithm.

```
for each ontology class c {
  for each OQE array term where c equals OQE array term {
    for each c anonymous c^sup list object property values p^v {
      if p^v NOT (null OR Resource OR Restriction OR Class) {
        add p^v to PV array. } } } } // relation class?
for each PV array p^v {
  for every c  where p^v equals c {
    if vector does not contain c { // relation class
      add c to vector and add c and weight to OQE array. } } }
```

4. Results and Discussion

In T401, 13,065 documents were queried to find 37 relevant documents and in T416, 160,838 docs were queried to find 10 relevant documents.

The grouped P&R results were considered by comparing P&R averages in two ways to account for queries containing skewed data:

i. precision calculated on the average *number* of relevant documents returned at each 10% recall point;

ii. precision based on pooling individual P&R curves, using a micro-evaluation averaging (MEA) approach [30], i.e. averaging precision *percentages*. MEA-based graphs are shown where P&R differs significantly from (i).

4.1. T401 "Foreign minorities, Germany"

The result of T401's query group Q1-6 comparisons (*K* vs. *All OQE*) is shown in Fig. 9. The *All OQE* mode resulted in significantly higher average precision performance over *K* across the 10-90% recall points. As the MEA-based P&R profile was similar it is not shown.

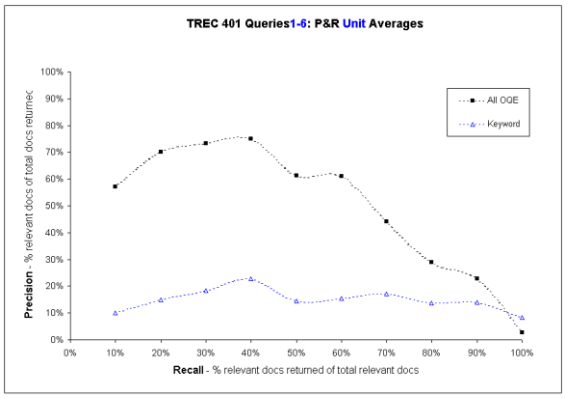

Fig. 9. T401 P&R: *K* v. *All OQE*.

The query group Q7-10 (*K* vs. *S+S OQE*) results show little difference between the *K* and *S+S OQE* profiles, see Fig. 10, with *K* precision marginally better than *S+S OQE* at 10% and 20% recall but lower at 30-80% recall.

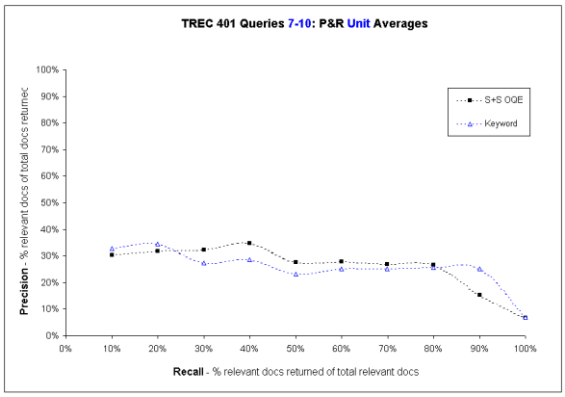

Fig. 10. T401 P&R: *K* v. *S+S OQE*.

However, the group results were partly distorted because, whilst query sets Q7, 8 and 9 showed better *S+S OQE* mode precision than *K*, the results averaging was affected solely by significantly higher document numbers returned in query set Q10. Fig. 11 shows the MEA-based P&R, with Q10 units issue minimised.

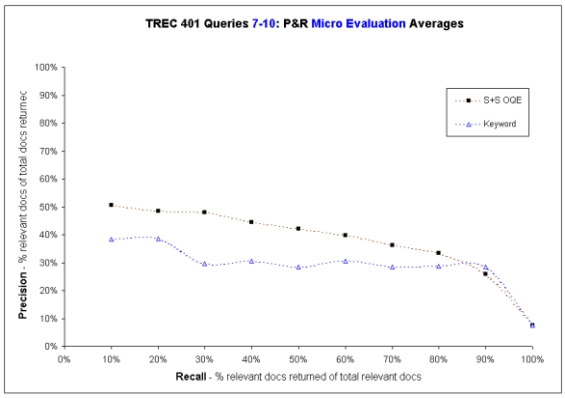

Fig. 11. T401 MEA P&R: *K* v. *S+S OQE*.

The combined P&R graph for all 10 T401 query sets Q1-10, i.e. comparing *K* to *OQE* modes (*All* and *S+S OQE*) is shown in Fig. 12.

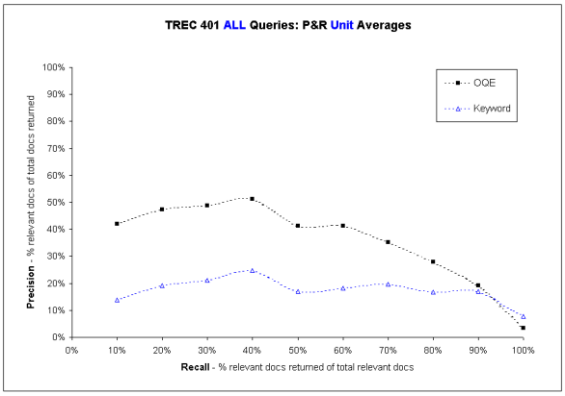

Fig. 12. T401 P&R: *K* v. *All/S+S OQE*.

The graph shows *OQE* produced a markedly better P&R curve than *K* between 10-90% recall. The MEA-based graph showed a raised precision (+10%) at 10-60% recall points for both *K* and *OQE* but is not shown, as the precision curve differentials were maintained at all recall points.

Recall was 100% for both *K* and *OQE*. On average *All* and *S+S OQE* generated 28 query terms, with 25 (91%) matched in document search, suggesting the ontology was very relevant. The lowest *OQE*s were for *S+S OQE*, with an average 8 terms, all matched.

4.2. T416 Three Gorges Project

The T416 P&R graphs compare combined results for all query groups Q1-10, for *K* vs. *S+S OQE* and *K* vs. *S+S+R OQE* modes - see Fig. 13.

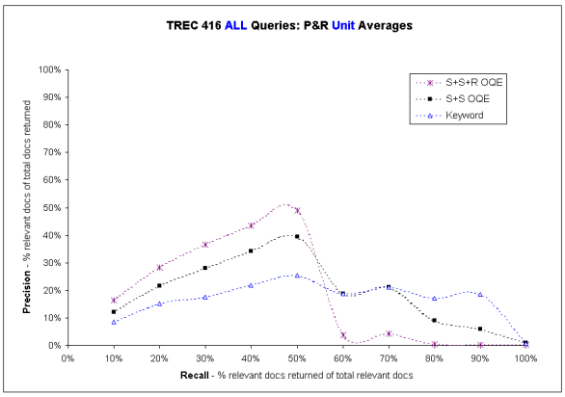

Fig. 13. T416 P&R: *K v. S+S/S+S+R OQE*.

The MEA-based graph shows similar relative P&Rs but precision is higher (≈10%) for *K* and *S+S OQE* and even better (≈20%) for *S+S+R OQE*) between 10-60% recall - see Fig. 14.

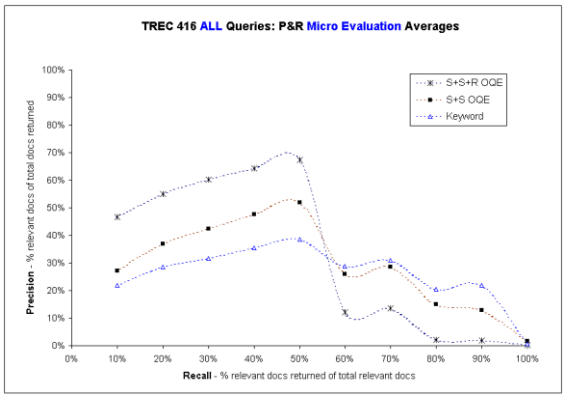

Fig. 14. T416 MEA P&R: *K v. S+S/ S+S+R.*

In both graphs, *S+S OQE* showed better precision than *K* mode at 10-50% recall but *S+S+R OQE* eclipsed both. *S+S OQE* generated on average 8 query terms, with 7 (93%) matched, while *S+S+R OQE* generated 21 terms, with 19 matched (95%). On average 95% of the 10 relevant documents were found using *K* and *S+S OQE* but 100% were retrieved by *S+S+R OQE*; suggesting *S+S+R OQE* can achieve higher recall than *K* and *S+S OQE* and the small ontology was again very relevant.

4.3. Comparing Query Mode P Success

Table 1 shows T401 query mode successes, i.e. how often *K* and/or *OQE* (*All* and *S+S OQE*) modes achieved the *top* precision P, at 10, 20 and 30% recalls R. Tied results are also shown.

Table 1

Mode	10% R	20% R	30% R	Ave. % top
K	2	1	1	13%
OQE	6	8	9	77%
Tied result	2	1	0	10%

OQE mode was on average most P effective 77% of the time and *K* mode 13% of the time. Both modes were joint top 10% of the time.

Table 2 shows the frequency of T416 *K*, *S+S* and *S+S+R OQE* query mode successes.

Table 2

Mode	10% R	20% R	30% R	Ave. % top
K	0	0	0	0%
S+S	1	1	1	10%
S+S+R	7	7	6	67%
Tied result	2	2	3	23%

S+S+R mode had the best average top P result (67%) over *S+S* (10%) and *K* (0%) but there was a higher level of tied P scores (23%).

4.4. Revised Concept Relevance Weights

Relevance weightings were modified for some later test queries: weights were *removed* by applying a standard weight of 1.0 rate to all concepts, and super class (0.7) and sub class (0.3) weights were reversed to 0.3 and 0.7 respectively – see P&Rs in Fig. 15.

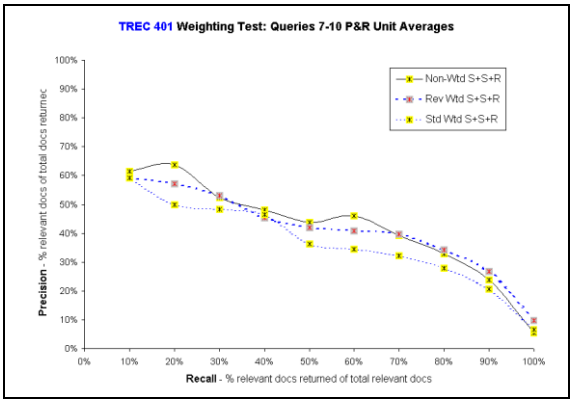

Fig. 15. P&R results of revised weightings.

The original section 2.2.3 weightings proved to be least favourable. Reverse weightings were slightly better but "removal" (applying 1.0 for all) produced the best result.

5. Concluding Remarks

The experiments have shown that overall, without resorting to semantic reasoning-based structured query languages, *OQE* can improve P in the 10-50% recall range. The Fig. 15 weight change results suggest that the original T401 and T416 *OQE* results could have been even more favourable. However, as some *K* queries gave better results, it seems that *K* and *OQE* modes can provide complimentary search options.

Small ontology contexts with flat hierarchies can restrict basic *S+S OQE* and require greater expressivity, e.g. for relation classes; however, small and specialised contexts may help to limit superfluous OQE, which could be problematic with a larger, generalised ontology.

Further work will refine the algorithms and could include extending keyword query term input by using text analysis to reduce a natural language sentence (long tail) query into keyword (short tail) sets for *OQE*.

Acknowledgements

Thanks to Peter Gray for his helpful comments on improving this work.

References

[1] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin and G. Sherlock, Gene Ontology: tool for the unification of biology, Nature Genetics 25 (1) (2000) 25-29.

[2] V. R. Benjamins, J. Davies, R. Baeza-Yates, P. Mika, H. Zaragoza, M. Greaves, J. M. Gómez-Pérez, J. Contreras, J. Domingue and D. Fensel, Near-Term Prospects for Semantic Technologies, IEEE Intelligent Systems 23 (1) (2008) 76-88.

[3] T. Berners-Lee, J. Hendler and O. Lassila, The Semantic Web, Scientific American 284 (5) (2001) 34-43.

[4] D. Berrueta, J. E. Labra and L. Polo, Searching over Public Administration Legal Documents Using Ontologies, in: Proceedings of Seventh Joint Conference on Knowledge-Based Software Engineering, 2006, pp. 167-175.

[5] J. Bhogal, A. MacFarlane and P. Smith, A review of ontology based query expansion, Information Processing and Management 43 (4) (2007) 866-886.

[6] W. N. Borst, Construction of Engineering Ontologies for Knowledge Sharing and Reuse, Ph.D. Thesis, SIKS - Dutch Graduate School for Information and Knowledge Systems, 1997.

[7] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi and J. Sachs, Swoogle: A Search and Metadata Engine for the Semantic Web, in: Proceedings of ACM Thirteenth Conference on Information and Knowledge Management (CIKM04), ACM, 2004, pp. 652-659.

[8] W.-D. Fang, L. Zhang, Y.-X. Wang and S.-B. Dong, Toward a semantic search engine based on ontologies, in: Proceedings of Fourth International Conference on Machine Learning and Cybernetics, 2005, pp. 1913-1918.

[9] R. Gligorov, W. ten Kate, Z. Aleksovski and F. van Harmelen, Using Google distance to weight approximate ontology matches, in: Proceedings of 16th international conference on World Wide Web, 2007, pp. 767-776.

[10] Google, Corporate Information, 2008, http://www.google.com/corporate/tech.html [Accessed 28 August 2008].

[11] T. R. Gruber, Toward principles for the design of ontologies used for knowledge sharing, International Journal of Human-Computer Studies 43 (5-6) (1995) 907-928.
[12] T. R. Gruber, A Translation Approach to Portable Ontology Specifications, Knowledge Acquisition 5 (2) (1993) 199-220.
[13] N. Guarino, Formal Ontology and Information Systems, in: Proceedings of 1st International Conference on Formal Ontologies in Information Systems (FOIS'98), IOS Press, 1998, pp. 3-15.
[14] Hakia, Search Engine Beta, 2008, http://www.hakia.com/ [Accessed 3 March 2008].
[15] H. H. Kim, ONTOWEB: Implementing an ontology-based Web retrieval system, Journal of the American Society for Information Science and Technology 56 (11) (2005) 1167-1176.
[16] Y. Lei, V. Uren and E. Motta, SemSearch: A Search Engine for the Semantic Web, in: Proceedings of 15th International Conference - Managing Knowledge in a World of Networks - EKAW 2006, Springer Berlin / Heidelberg, 2006, pp. 238-245.
[17] B. McBride, Jena: A Semantic Web Toolkit, IEEE Internet Computing 6 (6) (2002) 55-59.
[18] R. Navigli and P. Velardi, An analysis of ontology-based query expansion strategies, in: Proceedings of International Workshop on Adaptive Text Extraction and Mining at 14th European Conference on Machine Learning and 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, 2003, pp. 42-49.
[19] N. F. Noy and C. D. Hafner, The State of the Art in Ontology Design - A Survey and Comparative Review, AI Magazine 18 (3) (1997) 53-74.
[20] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn and G. Tummarello, Sindice.com: A Document-oriented Lookup Index for Open Linked Data, 2008, http://www.eyaloren.org/pubs/ijmso2008.pdf [Accessed 7 October 2008].
[21] Powerset, Powerlabs, 2008, http://www.powerset.com/ [Accessed 7 May 2008].
[22] A. L. Rector, Modularisation of domain ontologies implemented in description logics and related formalisms including OWL, in: Proceedings of 2nd International Conference On Knowledge Capture, ACM Press, New York, NY, USA, 2003, pp. 121-128.
[23] C. Rocha, D. Schwabe and M. P. de Aragão, A Hybrid Approach for Searching in the Semantic Web, in: Proceedings of 13th international conference on World Wide Web, 2004, pp. 374-383.
[24] G. Salton, A. Wong and J. Yang, A vector space model for automatic indexing, Communications of the ACM 18 (11) (1975) 613-620.
[25] K. Spärck Jones, A statistical interpretation of term specificity and its application in retrieval, Journal of Documentation 60 (5) (2004) 493-502.
[26] K. Sparck Jones, S. Walker and S. E. Robertson, A probabilistic model of information retrieval: development and comparative experiments, Information Processing and Management 36 (6) (2000) 779-808.
[27] R. Stevens, C. A. Goble and S. Bechhofer, Ontology-based knowledge representation for bioinformatics, Briefings in Bioinformatics 1 (4) (2000) 398-414.
[28] S. Tiun, R. Abdullah and T. E. Kong, Automatic Topic Identification Using Ontology Hierarchy, in: Proceedings of Second International Conference on Computational Linguistics and Intelligent Text Processing, 2001, pp. 444-453.
[29] TREC, Text REtrieval Conference, 2008, http://trec.nist.gov/ [Accessed 4 April 2008].
[30] C. J. van Rijsbergen, Information Retrieval, Butterworths, 1979, http://www.dcs.gla.ac.uk/Keith/Preface.html [Accessed 20 April 2008].
[31] E. Voorhees, Query expansion using lexical-semantic relations, in: Proceedings of 17th annual international ACM SIGIR conference on Research and development in information retrieval, 1994, pp. 61-69.
[32] W3C, OWL Web Ontology Language Guide, World Wide Web Consortium, 2004, http://www.w3.org/TR/owl-guide/ [Accessed 24 October 2007].
[33] W3C, RDF Primer, World Wide Web Consortium, 2004, http://www.w3.org/TR/rdf-primer/ [Accessed 15 November 2007].
[34] W3C, Semantic Web Case Studies and Use Cases, World Wide Web Consortium, 2008, http://www.w3.org/2001/sw/sweo/public/UseCases/ [Accessed 24 March 2008].

Appendix

Pseudo code based on Jena API library, for *class hierarchy* and *relation class* OQE algorithms.

OQE Algorithm – Class Hierarchy

Super class *OQE* adds direct super class line classes, i.e. it excludes sub class branches.

```
for ontology iterator list named_classes C {
  if C sub_C OR C super_C required {//SUB/SUPER CLASS
    for each keyword {
     if C Label OR LocalName equals keyword {
       if super_C is required {
         do_super_C_proc; }
       if sub_C is required {
         do_sub_C_proc;
         do_sub_C_individuals_proc; }
       do_C_proc; }
    if do_C_proc {
      Do_OQE_PROC; // Do Named Class Proc
      for C equivalent_C iterator list equivalent_classes {
        if equivalent_C is NOT intersection_C {
          Do_OQE_PROC; }
        else if equivalent_C is intersection_C {
          for equivalent_C intersection_C iterator list
              intersection_C members {
            Do_OQE_PROC; } }
     }
    }
    if do_super_C_proc AND named_C has super_C {//Get Super Cs
      for C super_C iterator list super_classes {
       if super_C is not anonymous_C {
         if super_C is not a Restriction_C AND
              not a Thing_C AND not a Resource_C {
           set Top_Class equal to next super_C;
           do_super_C_individuals_proc;
           Do_OQE_PROC; } }
       else if super_C is anonymous_C {
         if super_C is union_C {
           for super_C union_C iterator list union_C members {
           Do_OQE_PROC; } }
         else if super_C is intersection_C {
           for super_C intersection_C iterator list intersection_C
                members {
             Do_OQE_PROC; } }
         }
      }
      if do_super_C_individuals_proc {
       for ontology iterator list named_classes CI {
         if CI Label or LocalName equals Top_Class {
           for CI individuals iterator list individuals {
           Do_OQE_PROC; } } } }
    }
    if do_sub_C_proc AND C has a sub_C {
      for C sub_C iterator list sub_classes {
        Do_OQE_PROC; } } // Get SUB CLASSES
    if do_sub_C_individuals_proc AND
              (do_super_C_individuals_proc NOT executed) {
      for C individuals iterator list individuals {
        Do_OQE_PROC; } } // Get INDIVIDUALS
  } }
  else if sub_C NOT required AND super_C NOT required {
     Do_OQE_PROC; } // Get ALL Ontology CLASSES
}

if sub_C NOT required AND super_C NOT required {
  for ontology iterator list individuals {
    Do_OQE_PROC; } } // Get ALL Ontology INDIVIDUALS

Do_OQE_PROC { // POPULATE OQE ARRAY
  add class/individual Label to OQE array;
  add class/individual LocalName to OQE array; }
```

OQE Algorithm – Relation Classes

Relation classes are only generated for classes identified by the *class hierarchy* algorithm.

```
for ontology iterator list named_classes C { //Get ont classes
  add C LocalName and Label to ONT array;
  for each existing OQE array term {
    if C equals OQE array term { //matched term
      for C super_C iterator list super_classes {
        if super_C is anonymous {
          //Get asserted condition
          for super_C_property_value iterator list property_values {
            if  property_V LocalName does NOT equal null OR
                        "Resource" OR "Restriction" OR "Class" {
              // possible relation classes?
              add property_V LocalName to property_value array;
} } } } } }

for each property_value array item {
  for each existing ONT array term {
    if property_value array item equals ONT array class/term {
      // handle duplicates
      if vector does not contain ONT array class/term {
        add ONT array class/term to vector;
        // New OQE relation class
       add ONT array class/term LocalName and Label to OQE array;
       add Semantic_distance_weight to OQE array;
} } } }
```