The effect of lexical relationships on the quality of query clusters

Goh, Dion Hoe-Lian; Ray, Chandrani Sinha; Foo, Schubert

2006

Ray, C. S., Goh, D. H. L., & Foo, S. (2006). The effect of lexical relationships on the quality of query clusters. International Conference on Asian Digital Libraries ICADL (9th:2006:Japan), Lecture Notes in Computer Science, 4312, 223-233.

https://hdl.handle.net/10356/91554

https://doi.org/10.1007/11931584

Ray, C.S., Goh, D.H., and Foo, S. (2006). The effect of lexical relationships on the quality of query clusters. *Proceedings of the 9th International Conference on Asian Digital Libraries ICADL 2006*, (November 27-30, Kyoto, Japan), *Lecture Notes in Computer Science 4312*, 223-233.

The Effect of Lexical Relationships on the Quality of Query Clusters

Chandrani Sinha Ray, Dion Hoe-Lian Goh, Schubert Foo

Division of Information Studies, School of Communication and Information
Nanyang Technological University, Singapore 637718
{rayc0001, ashlgoh, assfoo}@ntu.edu.sg

Abstract. Query clustering is a useful technique that can help users frame an optimum query to obtain relevant documents. The content-based approach to query clustering has been criticized since queries are usually very short and consist of a wide variety of keywords, making this method ineffective in finding clusters. Clustering based on similar search results URLs has also performed inadequately due to the large number of distinct URLs. Our previous work has demonstrated that a hybrid approach combining the two is effective in generating good clusters. The present study aims to extend our work by using lexical knowledge from WordNet to examine the effect on the quality of query clusters as opposed to the other approaches. Our results show that surprisingly, the use of lexical knowledge does not produce any significant improvement in the quality of query clusters, thus demonstrating the robustness of the hybrid content-based plus search results-based query clustering approach.

Keywords: Query Clustering, Information Retrieval, Collaborative Querying

1 Introduction

Collaborative querying is gaining popularity among search engines as a tool for increasing the relevance of web search results. Collaborative querying exploits findings from research on information seeking behavior that interaction and collaboration with other people is an important part in the process of information seeking [8, 13]. The technique typically uses a history of past search experiences to enhance the current search [8]. Query clustering is often used in collaborative querying (e.g. [8], [20]), and makes use of the information contained in user logs to discover queries that are similar to other queries based on a variety of criteria. This is a form of implicit collaboration where other users' queries from the query logs are harnessed to reformulate or augment the current query.

Query clustering is different from document clustering where a document can be represented by a relatively large number of content words. In contrast, the information contained in queries is usually sparse and as a result, ambiguous [3]. Previous work has revealed that the average length of queries submitted to Web search engines is usually two to three words [12, 16]. Hence, it is often difficult to judge the specific information need or semantics of the query terms. For example, a user who types in "amazon" as a query may be looking for Amazon.com, the online bookstore or for information on the Amazon River. Further, people use a great variety of words to refer to the same thing [10]. Consider the case of two students seeking information about a top medical school in the United States in order to pursue higher studies in pediatrics. One may enter a general query such as "top medical schools US" while the other may choose to be more specific as in "best pediatric faculty America".

Methods used for clustering queries can be categorized into content-based, feedback-based and results-based approaches [8]. The feedback-based approach assumes that if users clicked on the same documents for different queries, these queries are similar [17, 20] and should be clustered. The results-based approach on the other hand is based on the principle that if two queries return the same result URLs, they are similar [4, 20]. The feedback-based approach and the results-based approach make use of only the result set of URLs returned by the user's query and hence suffer from the drawback that they entirely ignore the information contained in the query keywords. Moreover, the number of distinct result URLs is very large in a Web search engine which may lead to many similar queries not being grouped due to the lack of common URLs [4]. The content-based approach, also known as the *bag-of-words approach*, is based on the principle that if queries share a certain number of common keywords, they can be considered as similar. The success of the content-based approach in document clustering can be attributed to the fact that documents contain larger number of terms and hence, it is easier to find related documents. On the other hand, this approach

performs poorly in the case of queries primarily because of short query lengths. This limits the capacity to find similar queries which may express the same information need but have been framed differently using different keywords.

A method for improved text classification that incorporates linguistic knowledge into the bag-of-words approach using hypernyms from WordNet (http://wordnet.princeton.edu) was proposed in [14]. The authors use synsets (groups of synonymous words interchangeable in some context, eg. {mode, style, way, fashion}) from WordNet to replace nouns and verbs in documents. Using this approach, they were able to achieve a 47% drop in the number of errors as compared to the bag-of-words approach. The best performance was observed for cases where documents were authored by multiple authors employing very different terminologies like Digitrad, a public domain collection of folk song lyrics.

A similarity may be drawn between queries and Digitrad in that users use very different sets of keywords while searching [10]. Hence**,** it may be hypothesized that incorporating linguistic knowledge into query clustering can improve the quality of query clusters as compared to the content-based approach. In other words, queries which use different but synonymous keywords and are not clustered by the simple content-based approach would be clustered if the keywords were enhanced with synonyms. This paper thus uses lexical information from WordNet to generate query clusters, and compares the performance of this approach against our previous query clustering approaches (e.g. [8]). The synonymy relations from WordNet are used to replace the query terms with appropriate synsets. A new representation of query vectors in terms of the synsets is used to calculate the similarity between query vectors. These similarity measures are then used to generate query clusters. Finally, the quality of the generated clusters is measured using performance measures such as average cluster size, coverage, precision and recall.

The remainder of the paper is organized as follows. In Section 2, we review related literature. Section 3 describes the algorithm for generating clusters while Section 4 presents and discusses our experimental findings against various performance measures and contrasts it with the results obtained from other approaches. Finally, Section 5 concludes with a discussion of the implications of the results obtained and proposes areas for further research.

2. Related Work

The mining of query logs from search engines for the purposes of query reformulation has often been used in research on collaborative querying [5, 9]. These logs are a valuable source of information in that they provide indications for understanding the kinds of documents that users intend to retrieve using a set of particular query terms [5]. The information contained in the logs have then been used either for query expansion [5, 6] for suggesting terms related to users' queries to help in query reformulation [11] or for suggesting alternate queries to the user [9].

Using data/text mining techniques, these logs are processed to discover useful and interesting patterns in queries. For example, [20] used clustering on a set of queries from the Encarta encyclopedia based on two criteria: (a) query key words, also called content-based approach, and (b) clickthrough documents (feedback-based approach). It was found that a combination of the two approaches performed better than the individual approaches at low thresholds of similarity. The combined approach achieved an F-measure of more than 0.9 at a threshold of 0.6 compared with 0.8 for the individual approaches. Beeferman and Berger [1] use a criterion similar to the feedback-based approach in an iterative, agglomerative clustering of queries. Fu et al. [8] presented a comparison of the content-based approach, the results-based approach, and a hybrid approach combining the two. They concluded that the hybrid approach was seen to perform better than the two individual approaches. Besides clustering, [7] used association rule mining to discover similar queries. Here, each query session of a particular user is seen as a transaction. A problem faced in this approach is the difficulty in determining successive queries that form part of the same search session [7]. Cui et al. [5] proposed yet another method based on probabilistic correlations for isolating similar queries. The algorithm established correlations between the query terms and the clicked documents. Highly correlated terms were then used for query expansion.

The use of lexical and semantic information to improve information retrieval has been researched extensively. In particular, [14] used hypernym synsets from WordNet to represent text instead of the usual content-based approach for text classification. Their work concluded that classification using hypernyms

shows significant improvement in accuracy in cases where the documents are authored by multiple authors employing a wide range of terminologies. Besides text classification, linguistic knowledge has also been applied to query expansion. Voorhees [19] manually selected relevant synsets from WordNet for the task of query expansion. It was found that this technique improved retrieval effectiveness for queries that were short and incompletely formulated but not as effective for well-framed long queries. A different approach for query expansion, using knowledge of the *semantic domain* of the query terms, rather than synonymy and hypernymy relations, was used by [18]. They draw on knowledge about word co-occurrence from the word sense definitions in Wordnet to expand queries.

Although there is a fair amount of research on the use of semantic knowledge to improve document organization, its application to query clustering is limited. Our study presents an automated method for extracting and applying synsets to cluster queries. Since the query terms themselves are used to pick synsets, this approach also makes use of the important information conveyed by the content of the query.

3. Query Clustering Mechanism

This section presents the implementation of our approach to query clustering based on linguistic information from WordNet. The clustering process is divided into six phases.

3.1 Preparing the Dataset

The dataset used for our present work is a database of six months of queries drawn from the Nanyang Technological University (NTU) digital library. The queries cover a wide variety of subject areas such as various engineering disciplines, business administration, communication, etc. The query logs were preprocessed in the following ways, similar to [8] for purposes of comparison:

- A random sample of 35,000 queries were selected from the logs.
- Query terms were extracted leaving out additional information such as advanced options.
- Repeated queries were removed.
- Queries containing misspelled terms were discarded.
- Stop words such as “a”, “an”, “the”, etc. were removed.

After preprocessing, approximately 16,000 distinct queries were obtained. Before passing these queries to WordNet, numbers and special characters were eliminated since they would not be mapped to any synsets in WordNet. In addition, proper nouns such as names of countries and authors were themselves treated as WordNet synsets (eg. the query term “japan” was converted to the synset {Japan}. This was necessary as it was observed that proper nouns were either converted into higher level synsets such as {Asian country, Asian nation} for “japan” by WordNet or they would not be recognized and be eliminated. Proper nouns were detected with the help of WordNet as here, proper nouns are recorded with their first letter in the upper case, for example {Asia}. Part-of-speech taggers were not used as we noticed that many of these applications detect proper nouns by the capitalization of the first letter of the word, for example “Japan”. The taggers could not be applied to our queries since searchers typically do not pay attention to the case of letters when they type in their queries. Table 1 shows some sample queries.

Table 1. Sample queries extracted from query log

siphoning device, skin tissue, smart card introduction, synthesis for cobalt, thermal loading, thin film fabrication, thin films and Campbell

3.2 Extracting Synonyms from Wordnet

Figure 1 shows an example of an entry in WordNet. The first line of the entry gives the *sense number,* representing the different meanings of a word form. In the next line, the eight-digit number in the first part of the record is the unique identifier or *synset offset* of the *synset* {absent, remove}. This is henceforth referred to as synsetID. The part of the record following the “—“ symbol is a *gloss* or definition of the synset followed by some examples of usage.

A Java program using the JWNL API to the WordNet database (http://sourceforge.net/projects/jwordnet) was implemented to look up each query term and prepare a list of all its synonym synsets along with the

corresponding synset offset. The algorithm takes the preprocessed queries as inputs and submits each query term to the WordNet database to obtain a list of synsets with their synsetIDs. Multi-word terms are broken into their constituent words and each word is looked up separately. The algorithm in Table 2 shows our approach. Firstly, query terms are read from a query file. For each query term, we first check WordNet to see if a noun form of the word exists (step 3). The lookupIndexWord method searches for string q_t in file POS.NOUN (the part-of-speech noun word list of the dictionary). If the noun form exists, we call the function getSynset() which does the rest of the processing. The same procedure is repeated for the verb form (steps 6-8). For the adjective form (steps 9-12), we look for the satellite synsets of the word, since an adjective does not have a hypernym tree as in the case of nouns and verbs. If satellite synsets exist, the corresponding synsets and synsetIDs are written to the query database.

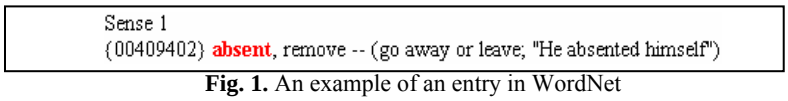

Fig. 1. An example of an entry in WordNet

Within the getSynset() function in step 14, we first find the number of senses in which the term is defined (step 20) since we need to collect synsets from each. Thereafter, we iterate over all the senses of the word to extract the first child of the hypernym tree (in the getHypernymTree method) which defines the synonyms of the word and retrieve the corresponding synset and synsetID in steps 16-18. These are written to the query database for further processing.

Table 2. Algorithm for extracting synsets from WordNet

```
1.     read qt;
2.     for each qt • Q
3.       word = dictionary.lookupIndexWord(POS.NOUN, qt);
4.       if (word != null)
5.          getSynset(qt);
6.       word = dictionary.lookupIndexWord(POS.VERB, qt);
7.       if (word != null)
8.          getSynset(qt);
9.       word = dictionary.lookupIndexWord(POS.ADJECTIVE, qt);
10.      if (word != null)
11.         while (satellite synsets exist)
12.            write synID, s;
13.
14.    function getSynset(qt)
15.      get n;
16.      for (i=1; i<= n ; i++)
17.         getHypernymTree for sense i up to level 1;
18.         write synID, s;

Key: Q = query; qt = query term; synID = synsetID; s = synset;
n = no. of senses
```

Following this phase, the query terms are replaced by the list of synsets, identified by their respective synsetIDs. Thus, while in the original content-based approach, the queries themselves form the feature set, in the new representation, each query is represented as a feature vector in terms of a set of synsetIDs. Each element of the feature vector represents a unique synset. Thus, if we define the set of queries as $Q = \{Q_1, Q_2 \ldots Q_i, Q_j \ldots Q_n\}$, then each query Q_j can be represented as:

$$Q_j = \{<s_1, w_{s1}>; <s_2, w_{s2}>; \ldots\ldots <s_i, w_{sj}>\} \qquad (1)$$

where s_i is a synset in Q_j, and w_{Sj} is the weight of the s_jth synset.

3.3 Weighting of Synsets

In our approach, two or more terms in a query may be mapped onto the same synset. This repetition enforces the context of a term's usage in a query. Consequently, an appropriate weighting scheme can be

used to assign greater importance to these repeating synsets over others. Here, the traditional TF-IDF scheme was used for weighting the synsets.

TF is the frequency of occurrence of a synset within a query and is computed as:

$$tf_{iQi}' = 1 + \log(tf_{iQi}) \quad (2)$$

Inverse query frequency, a concept borrowed from information retrieval, where it is better known as inverse document frequency, is calculated as:

$$iqf_i = \log\left(\frac{n}{qf_i}\right) \quad (3)$$

where n is the total number of queries and qf_i is the query frequency.

Finally, the synset weight is calculated as:

$$w_{si} = [1 + \log(tf_{iQj}')] * iqf_i \quad (4)$$

It should be noted that no distinction is made between parts of speech and word senses when picking synsets. This is due to the fact that, as mentioned earlier, queries tend to be usually short and hence, it is difficult to discern the context of usage of words. This approach is not a matter of concern in the case of longer queries, where many query terms map onto the same synset and increases the weight of the relevant synset. As for very short queries, the approach may cause some less relevant queries to be clustered. However, since very short queries are typically ambiguous and may not return a good set of results, it would be good to present the user with a range of query suggestions to help in query reformulation.

3.4 Similarity Computation

Similar to [8], the cosine similarity measure is adopted for this work because it takes into consideration the term weights and is popularly used in information retrieval. The numerator takes into account the common synset vector between two queries Q_i and Q_j, which we denote as C_{ij} and can be defined as:

$$C_{ij} = \{q : q \in Q_i \cap Q_j) \quad (5)$$

where q refers to the synsets common to both Q_i and Q_j. The cosine measure may now be expressed as follows:

$$Sim\_cosine(Q_i, Q_j) = \frac{\sum_{i=1}^{k} cw_{iQi} \times cw_{iQj}}{\sqrt{\sum_{i=1}^{k} cw_{iQ_i}^2} * \sqrt{\sum_{i=1}^{k} cw_{iQ_j}^2}} \quad (6)$$

where cw_{iQi} refers to the weight of i^{th} common synset of C_{ij} in query Qi and w_{iQi} is the weight of the ith synset in query Qi.

3.5 Clustering of Queries

The clustering algorithm puts two queries in the same cluster whenever their similarity value exceeds a certain threshold. Consequently, a cluster G is constructed for each query in the query set Q using the definition in Table 3 [15] where 1 < i < n, n being the total number of queries. Sim_cosine is the similarity value calculated for a pair of queries in the previous step. Threshold is the minimum value of sim_cosine which determines whether two queries should be placed in the same group. Thus, different thresholds will result in different clusters. We defined four similarity thresholds of 0.25, 0.50, 0.75 and 0.90 to measure the quality of clusters, to facilitate comparison with [8]. The quality of the clusters at these four thresholds is subsequently measured using different measures.

Table 3. Clustering Algorithm

for each Q_i $(1 \le i \le n)$
for each Q_j $(1 \le j \le n)$
if *sim_cosine* $(Q_i, Q_j) \ge$ *threshold*
then $(Q_i, Q_j) \in G(Q_i)$;

3.6 Evaluating the Quality of Query Clusters

Our quality indices measure how close the clusters obtained by the clustering algorithm are to those produced by manual/human clustering. These measures include average cluster size, coverage, precision and recall.

- **Average Cluster Size:** The average cluster size measures the ability of the clustering algorithm to provide recommended queries on a given query. It gives a quantitative measure of the variety of queries recommended to a user.
- **Coverage:** Defined as the percentage of queries for which the clustering algorithm is able to provide a cluster, coverage measures the ability of the clustering algorithm to recommend similar queries.
- **Precision:** Here, precision is defined as the ratio of number of similar queries to the total number of queries in a cluster:

$$\frac{\text{Number of Relevant Items Retrieved}}{\text{Total Number of Items Retrieved}} \quad (7)$$

In order to compute precision, a random sample of 100 clusters were selected and all 100 were manually examined (as in [8]) to see which of the queries in the cluster were actually similar. The overall precision was then computed as the average precision of all 100 query clusters.

- **Recall:** Recall is another performance metric popular in information retrieval and is defined as

$$\frac{\text{Number of Relevant Items Retrieved}}{\text{Total Number Relevant Items in Collection}} \quad (8)$$

Traditionally, recall would be defined as the ratio of the number of similar queries in the current cluster to the total number of all similar queries for the query set. This measure is difficult to compute as no standard clusters were available in the query data set. Hence, we used an alternative measure of recall known as *normalized recall* [20], defined as the ratio of the number of queries judged as correctly clustered in the 100 sample clusters for a particular threshold, to the maximum number of queries judged as correctly clustered in the 100 sample clusters across all thresholds. The number of correctly clustered queries within the 100 selected clusters equals the total number of queries in the 100 sample query clusters times average precision. The number of queries in 100 selected query clusters can be computed by average cluster size times 100. Thus, putting it together, the number of correctly clustered queries within the 100 selected clusters, n, may be defined as

$$n = \text{average precision} * \text{average cluster size} * 100 \quad (9)$$

Analysis of variance tests (ANOVA) were also conducted to reveal whether there were significant differences in terms of average cluster size, precision and recall across various thresholds. The chi-square test was used to measure the effect of thresholds on coverage, as the values were categorical.

In order to compute precision and recall, a sample of 100 clusters were manually evaluated by two evaluators. To examine the discrepancies caused by this subjective judgment, we also calculated the Kappa statistic to measure the degree of agreement between the two evaluators.

4. Experimental Findings

4.1 Results

By varying the similarity thresholds from 0.25 to 0.90, the average cluster size decreases from 93.09 to 10.42 (Table 4). A one-way ANOVA yielded a statistically significant variability in average cluster size across the thresholds, $F(3) = 29.03$, $p < .001$. Thus, at lower thresholds the clustering algorithm is able to provide a wider variety of queries for a query submitted by the user. As stated earlier, users tend to use a variety of words to express the same information need. Clustering with content words alone is unable to find queries that use synonyms or different forms of the same word, and hence the size of generated

clusters are small as in [8]. The larger cluster sizes in our present work can be attributed to enriching the queries with semantic information.

Table 4. Performance Results

	Threshold			
Performance Measure	**0.25**	**0.50**	**0.75**	**0.90**
Average Cluster Size	93.09	39.67	16.87	10.42
Coverage	0.99	0.98	0.93	0.86
Precision	0.65	0.80	0.86	0.93
Recall	1.00	0.52	0.24	0.16

In terms of coverage, even at a threshold of 0.90, the algorithm is able to provide clusters for a majority of queries (86.25%). At a threshold of 0.25, nearly all queries (99.56%) belong to one or more clusters, and hence, the probability of a user getting a recommendation for his query is high. A chi-square test for differences in coverage across thresholds was statistically significant, X^2 (3, N=48 000) = 4810, $p < .001$. The high coverage indicates that this approach is good at finding related queries for a given query.

desalination process	construction process and model
desalination process	thin film process
desalination process	Managing product and process development
desalination process	video process
desalination process	business process analysis

Fig. 2. Incorrectly clustered queries at threshold 0.25

The values of precision indicate an increase in precision from 65.40% at a threshold of 0.25 to 93.21% at 0.90. This indicates that at the threshold of 0.90 nearly 93.21% of queries have been correctly clustered. A one-way ANOVA yielded a statistically significant variability in precision across the four thresholds, F (3) = 20.88, $p < .001$. An examination of our clusters revealed an interesting observation which would also be applicable to the simple content-based approach. Most queries were very short and contained common and ambiguous words like “basic”, “process” and “design”. At higher thresholds of similarity, their effect is minimized by the more important concepts in the query like “malay” in the query “basic malay”. On the other hand, at lower thresholds, these words act as stopwords and tend to attract many irrelevant queries to the cluster. Figure 2 illustrates this phenomenon where the broad keyword “process” attracts queries which are not appropriate in the light of the core concept of the query, i.e. desalination. This affects the precision at lower thresholds to some extent. However, at the threshold of 0.50, this issue is mitigated as precision is good at 80%, which means that the chances of a user getting a good recommendation are high.

For the *normalized recall* computation [11], the maximum number of correctly clustered queries resulted for threshold 0.25. Hence, recall at this threshold was 100% [8]. This indicates that at this threshold, all the similar queries had been clustered. As can be seen from the table, there is a sharp drop in recall in the threshold range of [0.25, 0.75]. The smallest recall occurs at the threshold of 0.90. A one-way ANOVA yielded a statistically significant variability in recall across the thresholds, F (3) = 15.80, $p < .001$. Although the normalized recall is only an approximate measure of recall, it is able to give a fairly good indication of recall. Similar to the traditional recall measure, the normalized recall also varies inversely as the precision and gives an estimate of the percentage of similar queries that have been clustered.

Recall is good in the threshold range of [0.25, 0.50] but suffers at higher thresholds as compared to [8]. The low recall at higher thresholds can be attributed to two reasons:

1. The clustering algorithm was not able to relate acronyms with their corresponding expanded forms (eg “OOP” and “object-oriented programming”, “CAD” and “computer-aided design”). Hence, even though queries containing these terms should have been clustered, they were treated as different. Some form of acronym resolution would be required to resolve such errors.
2. A number of queries contained very technical terms which do not occur in the WordNet database (eg. “likert scale”, “OCR recognition”). Hence such terms were not provided with synsets, which resulted in inappropriate weighting of such queries. In the query “fabrication of CMOS” for instance, the term CMOS, which is the main subject matter of the query, is eliminated since it is not found in WordNet.

Thus, many queries such as "CMOS VLSI", "CMOS", and "analogue circuit design cmos" are not clustered. These would otherwise have been clustered using the simple content-based approach. This problem could be resolved by using an appropriate thesaurus.

4.3 Comparison of results with other approaches

The results of our present work and those obtained by [8] are presented in Figure 3 to facilitate comparison. Four clustering approaches have been plotted:

- Sim_syn: Our content-based approach using synonyms
- Sim_cos: Simple content-based approach [8]
- Sim_res: Results-based approach [8]
- Sim_hyb: Hybrid approach combining sim_cos and sim_res [8]

As can be seen from the figure, average cluster sizes are highest for sim_syn at most thresholds. At thresholds of 0.5 and above, the cluster sizes produced by this algorithm are the best. But at 0.25, the cluster sizes are too large to be of much value. In terms of coverage, sim_syn performs better with respect to the other approaches consistently, having clustered nearly all queries at the 0.25 threshold. This indicates that sim_syn has a better ability to provide recommendations for a given query.

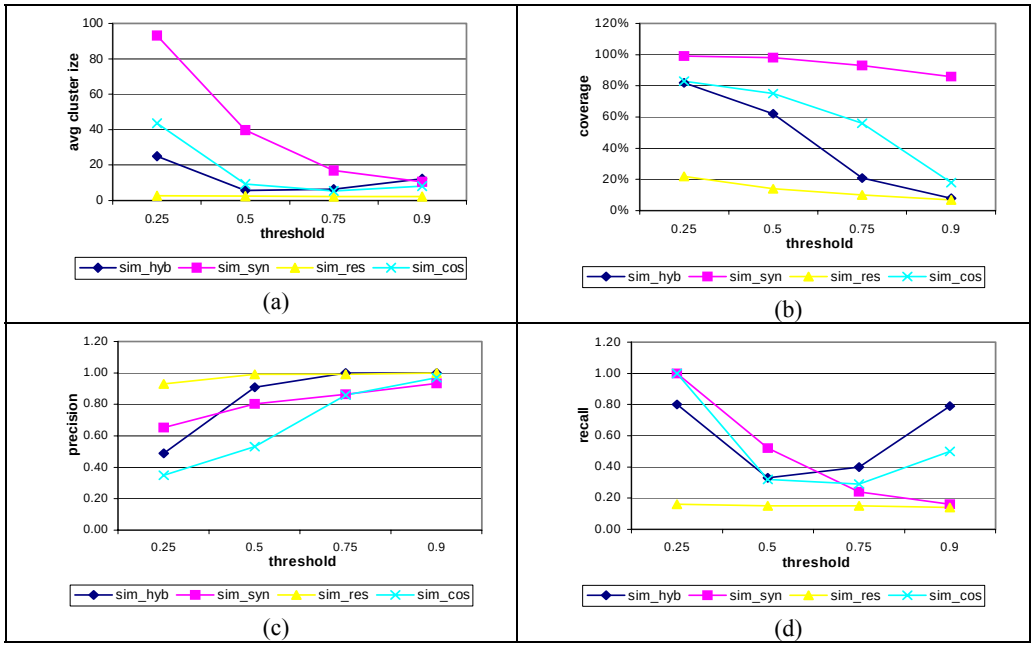

Fig. 3. Comparison of results with [8], (a) average cluster size, (b) coverage, (c) precision, (d) recall

In terms of precision, sim_res gives the best set of results (nearly 100% in the threshold range [0.50, 0.90]). Sim_syn performs only slightly better than sim_cos at the lower threholds. The low precision could be attributed to the reason stated in Section 4.1, which also applies to sim_cos. Sim_hyb performs best with respect to recall at the higher thresholds, where nearly 80% of similar queries have been clustered at threshold 0.90. At the lower thresholds, sim_syn performs best for recall.

The comparison of performance with the other approaches to query clustering indicates that the use of lexical information shows only slight improvement in the quality of query clusters as compared to the content-based approach. Since it uses the content of the query words to add lexical information, it is affected by many of the inherent weaknesses of the content-based approach. Even though in many cases the synonym based approach is able to find queries which would not otherwise have been clustered by the content-based approach, as shown in Table 5, the ambiguity of the content of the queries also helps to attract many irrelevant queries to the cluster. This greatly compromises on the precision of query clusters. Recall suffers mainly on account of the large number of technical terms in our queries which were not

adequately handled by WordNet. This synonym-based approach might be more useful in the case of longer queries, or where the context of the query is better understood through other means such as by examining result documents. This way, query keywords or phrases can be properly weighted to reflect their relative importance in the query. This will in turn help to select appropriate synonyms in order to increase the precision of query clusters. Each of the approaches discussed above perform well with respect to some quality measure. So, it is evident that by combining them in a suitable way, the most balanced set of results can be achieved. This is demonstrated by sim_hybrid, which combines the content-based approach and the results-based approach. It discounts the limitations of each of the approaches while also combining their strengths. Hence, the hybrid approach for query clustering is more robust as it can provide the best quality recommendations for a given query as compared to any other approach.

Table 5. Examples of clustered queries

Query	Related Query
engineering measurements fundamental	nanoscale science and technology
the econometrics of financial returns	Macroeconomics Blanchard
welding of ultra high strength steel	Soldering
procurement process	Internet purchase

5. Conclusion

We proposed an automated approach for query clustering using synonymy relations from WordNet and compared the clusters produced by this algorithm against the performance of clusters from other approaches discussed in [8]. The results of our experiments show that by integrating semantic and lexical knowledge in query clustering, although a wider variety of queries are able to find one or more clusters (refer to Section 4), the precision and recall of clusters is low. The clustering algorithm is able to find many queries that are related but are not identified by the content-based approach used in previous work (e.g. [8]) but does not perform as well as the hybrid approach, which combines the content-based and results-based approaches. Thus, it is evident that the users' information needs embedded in the content of the queries in the context of web search where queries are typically very short, may not be adequate to form good quality query clusters even when certain amount of lexical information is added. Yet the results suggest that better query clusters can be achieved when other available information like result URLs or information contained in result documents are taken into account together with synonyms. In other words, techniques for understanding the content of queries are required before lexical information can be applied accurately to queries.

In the light of our findings, we propose to extend our work, firstly, by applying word sense disambiguation or part-of-speech resolution on the query terms to enhance the precision of query clusters. Statistically determining word co-occurrence information from a large corpus is one possible way in which this can be done (e.g. [17]). Such co-occurrence patterns in queries can be used to pick out only relevant synsets instead of the complete list of synsets. We would also explore other query clustering approaches which can be used in conjunction with the synonym based approach to generate richer query clusters. It may also be possible to categorize queries according to genre or subject domain using available taxonomies such as Google Directory. This approach will allow users to choose optimally from their area of interest without forcing them to consider all forms of alternate queries.

Acknowledgments. This project is partially supported by NTU research grant number RG25/05.

References

1. Beeferman, D. & Berger, A. Agglomerative clustering of a search engine query log. *Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining*, (2000) 407-416.
2. Carpineto, C., De Mori, R., Romano, G. & Bigi, B. An information theoretic approach to automatic query expansion, *ACM Transactions on Information Systems*, *19*(1), (2001) 1-27.

3. Chien, S. & Immorlica, N. Semantic similarity between search engine queries using temporal correlation. *WWW 2005*, (2005) 2-11.
4. Chuang, S.L. & Chien, L.F. Towards automatic generation of query taxonomy: a hierarchical query clustering approach. *Proceedings of IEEE 2002 International Conference on Data Mining*, (2002).75-82.
5. Cui, H., Wen, J.R., Nie, J.Y. & Ma, W.Y. Probabilistic query expansion using query logs. *Proceedings of the eleventh international conference on World Wide Web*, (2002) 325 – 332.
6. Fitzpatrick, L. & Dent, M. Automatic feedback using past queries: Social searching? *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (1997) 306-313.
7. Fonseca, B.M., Golgher, P.B., De Moura, E.S. & Ziviani, N. Using association rules to discover search engines related queries. *First Latin American Web Congress (LA-WEB'03)*, (2003) 66-71.
8. Fu, L., Goh, D.H., Foo, S. & Na, J.C. Collaborative querying through a hybrid query clustering approach. *Digital libraries: Technology and management of indigenous knowledge for global access, ICADL 2003*, (2003) 111-122.
9. Fu, L., Goh, D.H., Foo, S. & Supangat, Y. Collaborative querying for enhanced information retrieval. *European conference on research and advanced technology for digital libraries (ECDL 2004),* (2004) 378-388.
10. Furnas, G.W., Landauer, T.K. , Gomez, L.M. & Dumais, S.T. The vocabulary problem in human-system communication. *Communications of the ACM, 30*(11), (1987) 964-971.
11. Huang, C.K., Oyang, Y.J. & Chien, L.F. A contextual term suggestion mechanism for interactive search. *Proceedings of 2001 Web Intelligence Conference (WI'2001)*, (2001) pp. 272-281.
12. Jansen, B. J., Spink, A. & Saracevic, T. Real life, real users and real needs: A study and analysis of users queries on the Web. *Information Processing and Management*, *36*(2), (2000) 207-227.
13. Lokman, I.M. & Stephanie, W.H. Information–seeking behavior and use of social science faculty studying stateless nations: A case study. *Journal of library and Information Science Research, 23*(1), (2001) 5-25.
14. Matwin, S. & Scott, S. Text classification using wordnet hypernyms. *Proceedings of the COLING/ACL Workshop on Usage of WordNet in Natural Language Processing Systems, COLING-ACL'98*, (1998) 45-52.
15. Raghavan, V.V. & Sever, H. On the reuse of past optimal queries. Proceedings of the Eighteenth International ACM SIGIR Conference on Research and Development in Information Retrieval, (1995) 344-350.
16. Silverstein, C., Henzinger, M., Marais, H. & Moricz, M. Analysis of a very large Altavista query log. Retrieved on February 12, 2003 from ftp://gatekeeper.research.compaq.com/pub/DEC/SRC/technical-notes/SRC-1998-014.pdf. (1998)
17. Turney, P.D. Word sense disambiguation by web mining for word co-occurrence probabilities. *Proceedings Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (SENSEVAL-3)*, (2004) 239-242.
18. Velardi, P. & Navigli, R. An analysis of ontology-based query expansion strategies. *Workshop on Adaptive Text Extraction and Mining at the 14th European Conference on Machine Learning, ECML 2003*, (2003) 210-221.
19. Voorhees, E. Using wordnet to disambiguate word senses for text retrieval. *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, (1993) 171-180.
20. Wen, J.R., Nie, J.Y. & Zhang, H.J. Query clustering using user logs. *ACM Transactions on Information Systems, 20* (1), (2002) 59-81.