Hypergraph-based Wikipedia search with semantics

G. Sudha Sadasivam*, K.G. Saranya and K.G. Karrthik

Department of Computer Science and Engineering,
PSG College of Technology,
Coimbatore – 641004, India
E-mail: sudhasadhasivam@yahoo.com
E-mail: saranyaa87@gmail.com
E-mail: kgkarrthik@rediffmail.com
*Corresponding author

Abstract: Wikipedia is a free, web-based encyclopaedia. This paper addresses the knowledge integration issue by computing semantic relatedness over a graph derived from Wikipedia by treating the articles as nodes and the links between the articles as the edges. Sentences with highest occurring keywords are extracted. These complex sentences are split into simple sentences and triplets with synonyms are extracted. A hypergraph structure is formed using hypernyms of the keywords to cluster the articles. Hypernyms extracted from the search query and keyword co-occurrences are used to extract relevant articles. Mapping the articles under the hypernyms category to an in-memory structure improves search efficiency and facilitates personalisation. The proposed work ensures the implied relationships between articles in the graph structure and maintenance of semantic relatedness between articles. Further, clustering the articles within the graph structure based on the hypernyms narrows down the search

Keywords: Wikipedia; hyper graph; semantics; hypernyms; in-memory; persistent graph.

Reference to this paper should be made as follows: Sadasivam, G.S., Saranya, K.G. and Karrthik, K.G. (2013) 'Hypergraph-based Wikipedia search with semantics', *Int. J. Web Science*, Vol. 2, Nos. 1/2, pp.66–79.

Biographical notes: G. Sudha Sadasivam is working as a Professor at the Department of Computer Science and Engineering, PSG College of Technology. She has authored five books and has published 40 papers in refereed journals. She has coordinated government and industrial projects in the area of distributed computing.

K.G. Saranya is working as an Assistant Professor in the Department of Computer Science and Engineering, PSG College of Technology. Her areas of interest include Semantic Web, personalised information retrieval and web services.

K.G. Karrthik is a student pursuing his ME in Software Engineering in PSG College of Technology. His areas of interest include high performance computing and NoSQL databases.

1 Introduction

Social networking sites, tagging systems, content management systems and wikis deal with inherently hierarchical or graph-shaped data that are deeply associative. Such recursive data structures are difficult to deal with in a relational database as it is not suitable to represent, store and manipulate complex dynamic and adhoc information (Cui et al., 2009). Hence, the need for new graph structures to represent Wikipedia database. Another issue with existing search structures is that they are either link-based or content-based. The paper proposes a graph-based persistent structure for Wikipedia database considering both the content and link relatedness of Wikipedia articles.

A graph database has full support for relationships to represent associative information. A graph database uses nodes, relationships between nodes and key-value properties instead of tables to represent information. Graph databases like Neo4j (2010) and key-value structures like Project Voldemort (2009) can implement graph structures. One of the problems faced by Wikis and social networks is handling large amounts of data. Hadoop scales the data grid and the compute grid. Data queries and aggregation can be carried out flexibly. It is a batch system that is suitable for sequential reads. It is not suitable for dynamic graph structures that require random write/read. Graph databases like Neo4j has far lower latencies for complex navigation problems. Hadoop (White, 2009) and other key-value stores are mostly concerned with relatively flat data structures, whereas Neo4j is concerned with deeper traversals. The proposed work aims at implementing a graph-based persistent structure for Wikipedia database using graph database and key-value stores. This aids to consider the implicit relationships between articles. Natural language processing is used to extract subject-verb-object triplets from sentences. This helps to maintain the semantic relatedness between the articles. Hypernyms are used to cluster the articles and thus represent the Wikipedia graph as a hyper graph. This approach helps to reduce the search space and improve search efficiency. Movement of the article clusters from persistent structure into an in-memory structure brings about time efficiency and personalisation.

2 Literature survey

The proposed work deals with parsing complex sentences, ontology construction, semantic graph construction, hyper graph construction and N-gram extraction. A literature survey was conducted in these areas. Extraction of background knowledge in the form of ontology can substantially help the search process (Ou et al., 2008). Processing of information in biomedical text to produce in simple syntactic constructs (Chidambaram, 2004) is domain specific. Complex sentence structures are broken into process able chunks. Information extraction is made easier by the inherent simplicity and dependencies between the chunks. In case of Wikipedia, the sentences were too complicated and provided more than one SVO pattern. We propose to split up this entries into two or more simpler sentences and then extract SVO patterns. Wikipedia articles had multiple subject terms and object terms. When multiple subject terms are encountered in a sentence, they are clubbed together as one subject term and then compared against a list of N-grams to see if this subject term has to be taken as one complete phrase. When multiple object terms are seen, each of the extracted SV terms are combined with each object term individually to provide multiple SVO patterns. SVO triplets (Rusu et al.,

2007) can be extracted from English sentences by generating parse trees. SVO knowledge can be used to construct a semantic graph. The hyper-graph model (Han et al., 1997) maps the relationship present in the original data in high dimensional space into a hyper-graph. A hyper-edge represents a relationship (affinity) among subsets of data and the weight of the hyper edge reflects the strength of this affinity. N-gram weigthtage could be done based on their position (Kumar et al., 1997). This method was modified from calculating weight values for N-grams to their frequency of occurrence in the given document. If they seem to occur more than a specified number of times (threshold value) in the document then they can be considered as valid n-grams in the given article.

A search data structure that uses content-based method inside link-based (Sudha and Karrthik, 2011) can be used to perform statistical search. To frame the graph structure the articles are considered as nodes and the link between the articles are considered as edges between the nodes. Each node in the graph has some properties. The name of the graph is one property, and the set of keywords that occur in it is the other. Every node has an edge to related articles. Relevance of related articles is also considered. This relevance is assigned to each edge based on keyword co-occurrence statistical information. The keyword entered in the search box is used to index into the graph. The articles that contain this keyword are fetched and taken as starting nodes to be traversed from inside the graph. Using the article's node, traversals can be made to neighbouring nodes and the search can be performed. This work does not include semantics; sentence-based searching and clustering articles. A similar approach was suggested in WikiWalk (Yeh et al., 2011).

The proposed work constructs the search graph as a complete hyper-graph with nodes clustered under certain categories, thereby each category of nodes forming a separate graph within the entire graph.

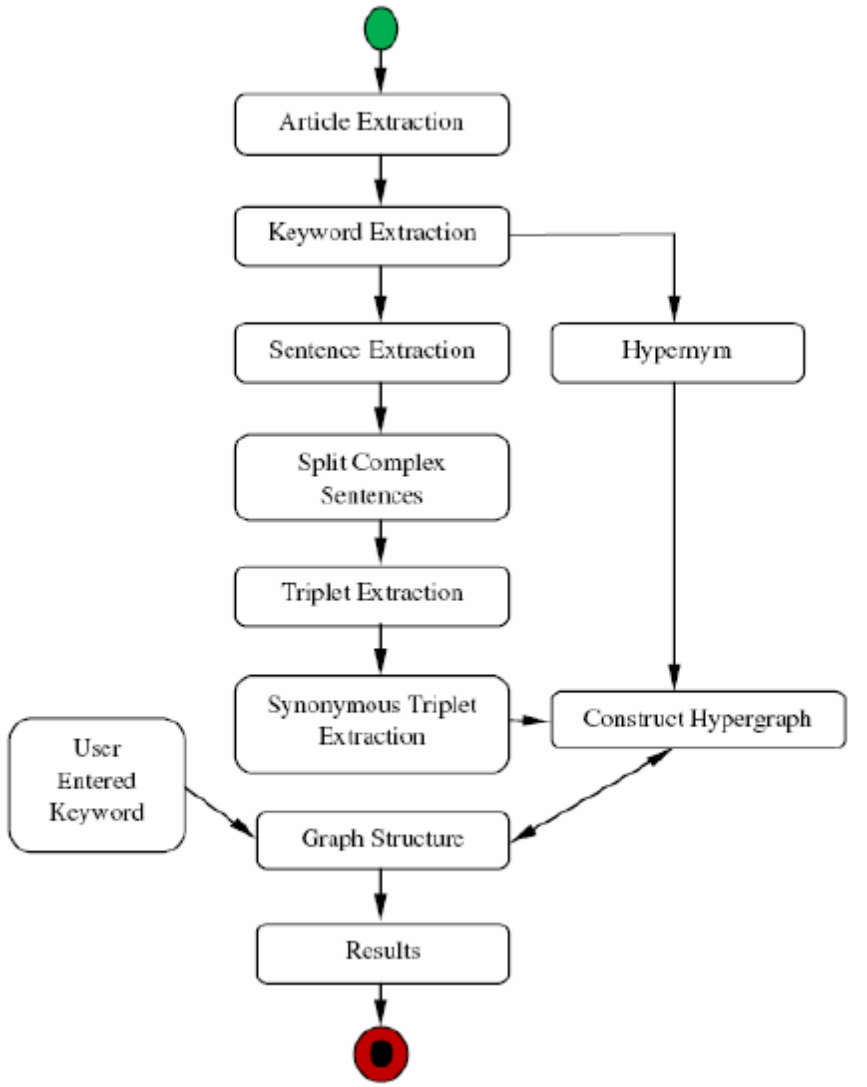
3 Proposed system

To construct the search hyper-graph initially a major category of articles is extracted from Wikipedia. Important keywords from each of the articles are extracted using Alchemy API (http://www.alchemyapi.com/). For each article, the sentence corresponding to the keyword that has the highest frequency of occurrence is extracted. Sentences that consist of a partial keyword are also considered. Complex sentences are split up into simpler sentences. Sentences which do not contain enough information to extract triplets are removed. Then for each of the refined sentences, Triplets (SVO) are extracted. The synonymous words for the subject, predicate and object terms in the extracted triplet are also fetched using Word net (Rusu et al., 2007) lexical database. These synonyms are used to generate all possible combinations of triplets. The hypernyms for the keywords is then used to cluster the articles. There may be more than one hypernym for the keywords extracted in the article, which means that a given article can be clustered under more than one hypernym cluster. Finally, the hypergraph is constructed from the triplets and hypernyms extracted. This resource description framework (RDF) hyper-graph structure acts as a catalyst for the searching of articles from the content-based Wikipedia graph structure.

During searching, the input sentence is first parsed to extract the triplets out of the search text. The hypernyms of the keywords in the sentence is used to fetch the cluster in the hyper-graph structure. This acts as a starting point from where the relevant articles are

fetched. Articles from all the hypernym clusters are extracted. This narrows down the number of articles searched. The retrieved articles are then searched for the SVO pattern from the given search sentence. This workflow is shown in Figure 1.

Figure 1 Proposed system (see online version for colours)



The steps involved in the implementation of the proposed system are as follows:

3.1 Article extraction

Articles are extracted from Wikipedia treating 'computer science', 'mechanics' and 'astronomy' as source pages through the wget command in Redhat Linux with a recursion depth of 2. A total of 25,652 articles were initially extracted from the Wikipedia site. Three different domains were considered (computer science, mechanics

and astronomy). In order to obtain the graph structure of Wikipedia, articles are treated as nodes and the links between the articles are considered as the edges. Each node contains the article name, keywords and the times at which they were last updated to the node. Links to other pages are treated as edges between the nodes. This structure is translated to a non-relational graph database in Neo4j.

3.2 Keyword extraction

Alchemy API has been used to extract keywords along with their relevance (0 to 1) from the Wikipedia articles through sophisticated statistical algorithms and natural language processing technology. N-grams of words have also been considered. Only keywords with relevance higher than the threshold are considered. The article title is added as a keyword to the file with a relevance of 1.

3.3 Sentence extraction

The sentences that contain the keyword with maximum frequency of occurrence are extracted from the article. Special characters (above ASCII 127) and text within braces that denotes notes from readers are removed. Hyphened (-) sentences and multiple sentences with ';' and ':' are split.

3.4 Handling complex sentences

Complex sentences contain multiple subject phrases and object phrases. Hence, SVO triplets extracted from these sentences become meaningless. Splitting is done by considering the tree bank generated by Stanford parser.

- In this parse tree, split the sentences at places where there is an 'S' depicting the start of a new sentence. Splitting at the 'S' gives two valid sentences in most of the problems.
- Sentences that have a 'W' word (like 'were', 'which', 'who', 'while', 'when', 'where') before 'S' should not be split.
- Sentences that have conjunctive words (like 'that', 'for', 'used', 'of', 'as', 'by', 'on', 'because', 'if') before the 'S' should not be split.

3.5 Triplet extraction

Stanford parse tree is used to extract these triplets. A sentence (S) is represented by the parser as a tree having three children: a noun phrase (NP), a verbal phrase (VP) and the full stop (.). The root of the tree will be S. The steps for triplet extraction are as follows:

1 The subject of the sentence is found by searching in the NP sub-tree. The subject will be found by performing breadth first search and selecting the first descendent of NP that is a noun. Nouns are found in the following sub-trees:
   - NN – noun, common, singular or mass
   - NNP – noun, proper, singular

- NNPS – noun, proper, singular
- NNS – noun, common, plural.

2 The predicate of the sentence is found from the VP sub-tree. The deepest verb descendent of the verb phrase will give the verb element of the triplet. Verbs are found in the following sub-trees:

- VB – verb, base form
- VBD – verb, past tense
- VBG – verb, present participle or gerund
- VBN – verb, past participle
- VBZ – verb, present tense, third person singular
- VB – verb, present tense, not third person singular.

Some subjects or objects might be N-grams in that particular document. The possible N-grams of a given document are first extracted and then each of the subjects and objects identified are compared with this list of N-grams and kept accordingly. Thereby longer looking subject and object terms can be eliminated.

3 The objects are found in three different sub-trees, all siblings of the VP sub-tree containing the predicate. The sub-trees are PP (prepositional phrase), NP and ADJP (adjective phrase). In NP and PP the first noun is looked for, while in ADJP the first adjective has to be found.

There are also some special cases while looking for these RDF triplets. Some sentences might consist of multiple subject terms in the noun phrase. In such sentences all the nouns from the first noun phrase have to be extracted and combined into one complete subject phrase. When a sentence has multiple objects all the objects are extracted and individually each of these objects will be combined with the SV terms to give multiple SVO triplets for the same sentence.

3.6 Extraction of synonymous triplets

For each of the subject, predicate and object, their synonymous words from the Word net database are taken and all the possible combinations of SVO triplets are generated. Thereby we get more semantic interpretation to the search sentence given.

3.7 Hyper graph construction

Neo4j graph database is used to construct the hypergraph structure. Here, nodes represent the articles and links represent the links between the articles are used to construct the graph structure. The following steps are carried out in construction of hypergraph:

1 *Link extraction:* It involves identifying three major types of links including infobox, categorical and content links.

2 *Generality filter:* Links are further factored out according to generality to eliminate irrelevant articles. A generality filter is used for this purpose. Articles that are irrelevant and too specific to the search are removed using generality filter.

Generality filter is based on the number of incoming links. One third of irrelevant articles are eliminated using the generality filter.

3 *Prioritising the links:* This is done by considering the linkage between two article nodes.

The relevance value between the article nodes are set as properties on the edges connecting the respective articles. The edges between the nodes have a weight that represents the fraction of keywords the two articles have in common. If article 1 (A1) has 'n1' keywords and article 2 (A2) has 'n2' keywords, and if they both have 'n' keywords in common, the fraction of common keywords between A1 and A2 is given by the formula:

$$\text{Relatedness} = \frac{2 * n}{n1 + n2}$$

The reciprocal of the fractions are now considered as edges and the Djikstra's shortest path algorithm is applied. Based on the path taken, link's priority is given as follows:

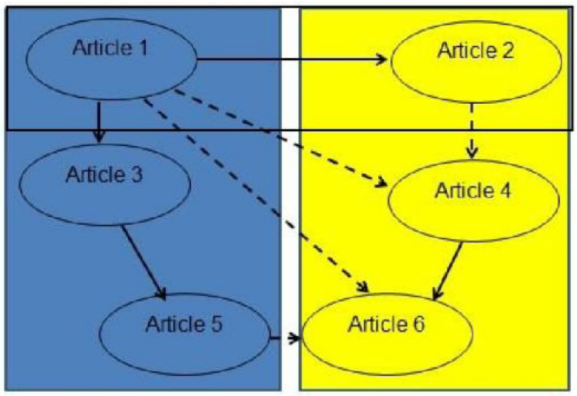
$$\text{Link priority} = \sum_{i=0}^{n-1} e_i \left(\frac{1}{2} * i \right)$$

where $e_i = 1/\text{Relatedness}$.

Now the articles are clustered within the graph structure, by extracting the hypernyms of the keywords in each of these articles. These hypernyms are used to cluster the article nodes. This hypernym category is set as a property to each of the nodes in the graph, and thus querying based on the hypernym would give all the articles in that particular category. This means that each of the articles under one hypernym can be seen as a separate graph inside this entire graph.

Each of the articles may belong to more than one hypernym category. Thus there might be plenty of linked hypernym-based graphs inside one big Wikipedia article graph. The articles contain as properties, the name of the article and the set of triplets that occur in them. Link weights are based on the keyword cooccurrence between the articles. A simple hyper graph is shown in Figure 2. Article 1 can belong to two categories (hypernyms).

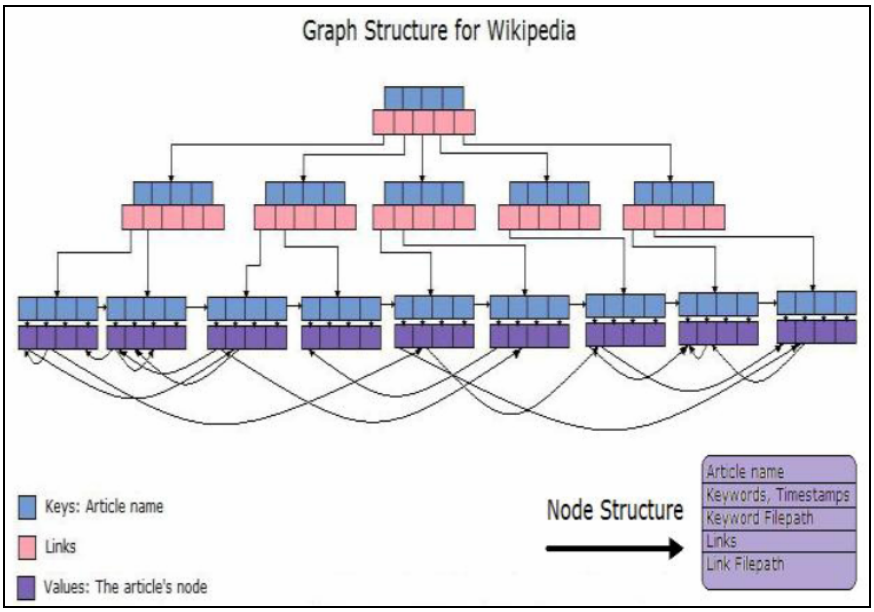
Figure 2 A simple hyper-graph structure (see online version for colours)



3.8 In-memory structures for personalisation

Categories related to the user are moved to in-memory structure. Two such in-memory structures have been constructed and evaluated. They are B+ trees and graphs. In B+ tree structure, one B+ tree is used to store keywords along with a hash table containing the articles in which it occurs and relevance. The keyword is the key and hash table (article, relevance) is the value. Wikipedia graph is implemented through another B+ tree (Figure 3). Each node in the graph is a leaf node in B+ tree. Each leaf node has an edge to related articles. Relevance of related articles will also be considered. Here the article name is the key and the node is the value.

Figure 3 In-memory B+ tree for Wikipedia graph (see online version for colours)



3.9 Search

Search procedure involves the following:

- split the complex search sentence into two or more simpler sentences
- extract the SVO triplets from the search sentence
- generate synonymous triplets
- extract the hypernyms of keywords from search sentences
- query the hyper graph database based on hypernyms
- fetch relevant articles by traversing the graph and only considering those links that have relevance above a particular threshold move into in-memory structure for personalisation.

4 Experimental results

The results were taken for different values of link priority threshold and at each threshold value. Four evaluation measures were calculated. Finally, the performance of the sentence-based search is compared with that of the content-based search. The performance of in-memory graph structure was compared to persistent graph structure.

4.1 In-memory structure

The in-memory structure consists of the articles under the hypernym category of the sentence given for search. All the articles under the hypernym category of the search sentence's hypernym category are taken from the persistent structure and these articles along with their links and link relevancies are loaded into the in-memory structure.

4.1.1 Link priority threshold variations

As shown in Table 1, a link threshold value of 0.8 takes precision to very low values and a threshold of 0.4 takes accuracy below 50. But a threshold of 0.6 strikes a balance between precision and recall. As seen there is a reasonable value of F-measure which shows the balance between precision and recall. Also the accuracy for this threshold is good. To get a good F-measure threshold should be maintained between 0.5 and 0.7.

Table 1 Metrics for different thresholds of link priority

Link priority threshold	*Precision*	*Recall*	*Accuracy*	*F measure*	*Time taken to search for triplets in the in-memory structure (in secs)*	*Time taken to fetch the hypernym-based articles from the persistent hyper-graph (in secs)*
0.8	40.0	76.92	97.12	52.63	5	10
0.7	64.0	72.72	97.60	68.08	5	10
0.6	84.0	70.0	97.92	76.36	6	10
0.5	70.0	55.0	93.13	49.41	6	10
0.4	52.94	37.5	82.23	43.90	6	10

4.1.2 Comparing in-memory structures

Table 2 depicts that the in-memory graph structure has a better accuracy and F-measure for similar values of link threshold. Although the graph structure has a delay in the search time, it provides better results. This marginal delay in search time may be because not all the nodes in the graph are at the same level whereas in a B+ tree all the article carrying nodes are at the same level (leaf). It is clear from the above comparison that the hypernym-based approach has a better performance in terms of accuracy, recall and precision. However, on the other side, it is slow in terms of time efficiency. This can take up to 5 seconds at the worst for graph structure containing some 800 article node.

Table 2 Comparing in-memory structures – B+ tree and graph

Methods	*Link priority threshold*	*Keyword threshold*	*Precision*	*Recall*	*Accuracy*	*F measure*	*Time (in sec)*
B+ tree (content-based)	0.55	0.5	32.0	26.0	73.0	28.6	1
Graph (content-based)	0.5	0.5	52.94	37.5	82.23	43.90	7.5
Graph (hypernym-based)	0.6	0.5	84.0	70.0	97.92	76.36	5

4.2 Using persistent structure

The results using persistent structure reveals that the accuracy is much higher than the accuracy values obtained using in-memory structure as the search space has been increased.

4.2.1 Link priority threshold comparison

As shown in Table 3, the optimal threshold that can be set for the links is 0.6 since only for that value we get a good F-measure.

Table 3 Metrics for persistent structure's use

Link priority threshold	*Precision*	*Recall*	*Accuracy*	*F measure*	*Time (in sec)*
0.8	40.0	76.92	97.12	52.63	25
0.7	64.0	72.72	97.60	68.08	27
0.6	84.0	70.0	97.92	76.36	27
0.5	84.0	35.0	93.13	49.41	30
0.4	52.94	37.5	82.23	43.90	35

4.2.2 Generality filter threshold variations

The generality filter plays a vital role in determining how many articles will be present in the graph structure. Table 4 provides a comparison of how the in-links and link threshold affects the performance of the persistent graph structure.

Table 4 Behaviour change of persistent structure depending on the in-links threshold

Generality filter threshold (in-links)	*# of files left*	*Link threshold*	*Search time in persistent structure (in sec)*
10	2171	0.7	75
20	1566	0.5	45
30	1281	0.4	40
45	980	0.3	30
60	815	0.2	30
100	430	0.2	25
150	258	0.5	25

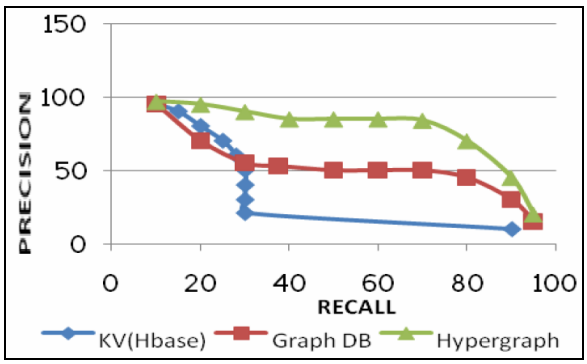
4.2.3 Comparing persistent structures

This persistent structures' comparison shows better accuracy and F-measure at the cost of search time. But in information retrieval, the F-measure and accuracy give a competitive edge over the optimal search time. Also it is seen that the hypernym-based approach is more productive in terms of search accuracy and precision, thereby ignoring the search time (Table 5). Figure 4 shows the precision recall curve for the persistent structures. It can be seen that Hypergraphs using hypernyms perform better when compared to the content-based key value and graph structures.

Table 5 Comparison of performance of persistent structures – Hbase and Neo4j

Methods	*Link priority threshold*	*Keyword threshold*	*Precision*	*Recall*	*Accuracy*	*F measure*	*Time (in sec)*
Hbase (content-based)	0.55	0.5	30.0	21.0	79.0	24.7	1.5
Neo4j (content-based)	0.4	0.5	52.94	37.5	82.23	43.90	2.0
Neo4j (hypernym-based)	0.6	0.5	84.0	70.0	97.92	76.36	25.0

Figure 4 Recall vs. precision for persistent structures (see online version for colours)



4.3 Comparison between in memory and persistent structures

It is clear from Table 6 that the in-memory search structure can perform better than the persistent structure in terms of search time. But this in-memory structure has space limitations. The time taken to move a category from persistent to in-memory is also shown. It can be seen that the time for search is minimised in spite of the time taken to move from persistent to in-memory structure.

Table 6 Comparison of performance of persistent and in-memory graph structures

# files	*# files moved to in-memory*	*Search time in seconds*			*Search time in persistent structure*
		In-memory search time	*Time for movement*	*Total search time*	
258	118	4	26	30	39
430	179	5	29	34	41
815	328	8	35	43	47
930	345	8	38	46	51
1,281	409	9	42	51	57
1,566	594	9	45	54	61
2,171	685	11	52	63	69

4.4 Comparison of content-based and hypernym-based search

The search results for the three methods namely, content-based, triplet-based, and triplet with hypernym are compared in Table 7. The results show that even at a higher value of link threshold taken, the triplet-based method performs better than the content-based method (F-measure value score). In the triplet-based method, the articles are not clustered according to their hypernym and so while searching for articles with the extracted triplets, we have to search in each article for the particular triplet. Thus, the search time increases to much as 10 minutes.

Table 7 Comparison of performance of content-based, triplet-based and hypernym-based persistent structure searches

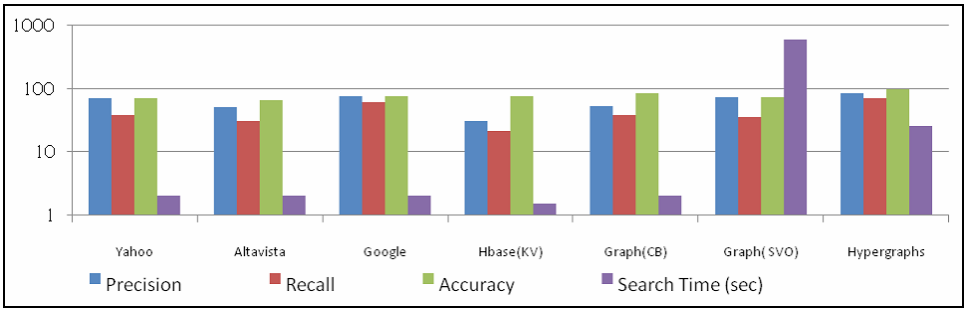
Methods	*Link priority threshold*	*Keyword threshold*	*Precision*	*Recall*	*Accuracy*	*F measure*	*Time (in sec)*
Neo4j (content-based)	0.4	0.5	52.94	37.5	82.23	43.90	2.0
Neo4j (triplet-based)	0.6	0.5	74.0	35.0	73.13	52.32	600.0
Neo4j (hypernym-based)	0.6	0.5	84.0	70.0	97.92	76.36	25.0

When articles are clustered by hypernym category, the search time is greatly reduced because we retrieve a set of articles that belong to the category of hypernym to which the search sentence belongs, and in those articles we look for the ones which contain the triplets in the search sentence. Thus, we eliminate unnecessary search through the other documents in the graph.

The search time for the content-based method is low. It is based on retrieving articles for particular keyword and then traversing the article graph based on the articles retrieved. Whereas in both the triplet-based and hypernym-based search methods, we need to extract the triplets from the given search sentence, find out the hypernym to which the search sentence belongs to and then retrieve all the articles from the graph which belong to the corresponding hypernym category. Then these articles are searched for the triplets that are extracted from the sentence. After the articles in which the triplets

occur are found out, the articles are taken as starting nodes and the graph is traversed to fetch the relevant articles. Thus, semantic interpretation to the search term increases in the search time of the hypernym-based method. Figure 5 compares the performance of the proposed approaches with existing search engines. It can be seen that due to semantics, precision recall and accuracy is improved, but search time also increases. This can be reduced using hypergraph-based approach. Thus, hypergraph-based approach not only improves precision, recall and accuracy, but also reduced search time.

Figure 5 Comparison of performance of existing and proposed approaches (see online version for colours)



5 Conclusions

Through the use of co-occurrence statistical information, it is concluded that the search yields statistically related results in addition to the regular search results with high accuracy. By using Neo4j, extension to the search space was provided increasing the level of accuracy. But searching directly from the persistent structure increases the search time as large number of nodes had to be traversed. Searching the articles based on the extracted SVO patterns ensures that sentences can be searched in an efficient way with semantics. The persistent hyper graph structure constructed using the hypernyms of the article ensures that only the relevant articles are fetched from the persistent graph structure. These articles can also be mapped to an in-memory structure to make sure the search is done even faster. The search system based on hypernym clusters in the graph structure has a better precision and recall than that of the content-based system. Querying using sentences is also facilitated in the proposed system. The search time was measured for both the in-memory hyper-graph structure and the persistent (Neo4j) hyper-graph structure. The in-memory structure yields better search time with the same precision and recall values for given set of article nodes as persistent hyper-graph structure. Hence a combination of both can be used to move personalised search information from the persistent structure to in-memory structure. This can improve search efficiency.

Acknowledgements

The authors acknowledge the support given by Dr. Sujatha Uppadhyaya, Director, Xurmo Technologies and Mr. Chidambaran Kollengode, Director, Cloud Computing and Big Data Analytics, Nokia R&D, Bangalore. This project is carried out as a consequence of PSG-Nokia Research collaborations on Big Data Analytics and PSG-Xurmo Research on Social Networking.

References

Alchemy API [online] available at http://www.alchemyapi.com/ (accessed 20/11/2011).

Chidambaram, D. (2004) *Processing Complex Sentences for Information Extraction*, December, MSc thesis, Arizona State University,

Cui, G., Lu, Q., Li, W. and Chen, Y. (2009) 'Mining concepts from Wikipedia for ontology construction', Paper presented in *IEEE International Conference on Web Intelligence and Intelligent Agent Technology*, 15–18 September 2009, Milano, Italy.

Han, E-H. (Sam), Karypis, G. and Kumar, V. (1997) 'Clustering in a high-dimensional space using hypergraph models', *International Journal of Scientometrics, Informetrics and Bibliometrics*, Vol. 2, No. 4, pp.370–376.

Kumar, N., Vemula, V.V.B., Srinathan, K. and Varma, V. (1997) 'Exploiting n-gram importance and wikipedia based additional knowledge for improvements in gaac based document clustering', Paper presented in *International conference on Research Issues on Data Mining and Knowledge Discovery – DMKD*, August 1997, California.

Neo4j (2010) *Neo4j: NOSQL for the Enterprise – Neo4j Manual* [online] http://docs.neo4j.org/chunked/milestone/ (accessed 15/12/2010).

Ou, S., Pekar, V., Orasan, C., Spurk, C. and Negri, M. (2008) 'Development and alignment of a domain-specific ontology for question answering', Paper presented in *6th International Conference on Language Resources and Evaluation*, May 2008, Morroco.

Project Voldemort (2009) [online] http://project-voldemort.com/ (accessed 25/03/2010).

Rusu, D., Dali, L., Fortuna, B., Grobelnik, M. and Mladenić, D. (2007) 'Triplet extraction from sentences', *Journal of Knowledge Creation Diffusion Utilization*, October 2007, pp.8–12.

Sudha, G. and Karrthik, K.G. (2011) 'Persistent graph structures for Wikipedia search', Paper presented in *IBM ICARE 2011, Next Generation Systems*, October 2011, Delhi, India.

White, T. (2009) *Hadoop: The Definitive Guide*, 2nd ed., O'Reilly Media, USA.

Yeh, E., Ramage, D., Manning, C.D., Agirre, E. and Soroa, A. (2011) 'WikiWalk: random walks on Wikipedia for semantic relatedness – TextGraphs-4', Paper presented in *Workshop on Graph-based Methods for Natural Language Processing, 2011*, Portland, Oregan.