iSeS: Intelligent Semantic Search Framework

M. Jayarathne, D. Fernando, C. D. Arachchi,
I. Haththotuwa, S. Perera
Department of Computer Science and Engineering,
University of Moratuwa,
Moratuwa, Sri Lanka

S. Weerakoon
Zone 24x7 (Pvt.) Ltd.
Colombo, Sri Lanka
sajithw@zone24x7.com

***Abstract*—With vast amounts of data being produced, present world is overwhelmed with information and searching for appropriate content has turned out to be harder than ever before. Semantics, which typically focuses on the relationship between signifiers, such as words, phrases, signs and symbols, and what they stand for is now being used more and more in search engines to provide the user with more meaningful content. Further it is no more the case that users are interested in search results that the majority of users would agree to, but are more interested in results being personalized to them.**

In this research paper we present iSeS: Intelligent Semantic Search Framework, which is a search framework that a custom web site or an application can adapt. We focus on using underlying semantics of the content being indexed in providing more meaningful search results personalized to each user. We look into both latent semantic indexing and metadata extraction based methods for providing semantically rich search results. Collaborative filtering and how it is used to personalize search results is also explored in this paper.

Keywords-component; Semantics; Metadata; Search; Indexing; Latent Semantic Indexing

I. INTRODUCTION

Today the websites in the World Wide Web are becoming increasingly sophisticated and contain vast amounts of data. The busy schedule of humans today means that they would not like to or they would be unable to spend time browsing a particular website in search of some particular item. Therefore the easiest option has been to use a search engine to find the required information.

Even though sites like "Google" offer amazing search results and also personalized to some extent based on the past user behavior, in general individual sites lack efficient and relevant internal search engines for searching within its site. Almost all the search engines use text based search where it matches the query string with the text in the files. The search result is generated mostly based on the number of occurrences and this doesn't take the real meaning of the query string into account. The same applies to an application where the user is trying to find help details.

These search engines merely focus on parameters such as count of the search query in the document/webpage. However in today's context this is not merely enough. The user is in need of a personalized search as well as the relationship between the search results in order to make the relevant choice. This would save a significant amount of time for the user who will otherwise have to navigate into the website to find the relationship.

What the industry lacks is a platform integrated with semantic search and personalized aspects whereby the developer can directly plug in the framework after customizing it to the personal requirements.

II. BACKGROUND

This section carries a detailed description of related areas of study and the relevant concepts about the project's subject matter.

A. Searching based on Semantics of Data

Semantic search is the method of searching through documents considering more than syntactic level of keyword matching [1]. This is useful to conduct an intelligent search, unlike in keyword-matching based methods, where the search is uninformed and monotonous. A resulting document/data source that is very much related to a search query might not have even a single word or phrase common to it. But still, when the semantics of the search query is considered, that particular result might be the most relevant one. But a keyword based search fails to capture such results, since there are no matching keywords present. For searching methods which compare the actual meaning of data, all the steps which are related to the search should be aware of the semantics of data [2]. Semantic languages are used for keeping track of metadata/semantics of a data source. To develop a simple hierarchy of semantics for a web page or a local data file, simple concepts such as generalization, aggregation, association etc. can be used.

B. Latent Semantic Analysis

Latent Semantic Analysis (LSA) is a statistical technique for extracting and analyzing relations of expected contextual usage of words in documents. It is not a traditional natural language processing or an artificial intelligence based mechanism and takes only raw text parsed into words defined as unique character strings and separated into meaningful passages or samples as inputs. [3], [4].

In LSA first we need to represent the text as a matrix in which each row stands for a unique word and each column stands for a text document/passage. Each cell contains the frequency with which the word of its row appears in the passage denoted by its column. Next, the cell entries are

subjected to a preliminary transformation in which each cell frequency is weighted by a function that expresses both the word's importance in the particular passage and the degree to which the word type carries information in the domain in general.

Next, LSA applies singular value decomposition (SVD) [5] to the matrix. This is a form of factor analysis. In SVD, a rectangular matrix is decomposed into the product of three other matrices. One component matrix describes the original row entities as vectors of derived orthogonal factor values, another describes the original column entities in the same way, and the third is a diagonal matrix containing scaling values such that when the three components are matrix-multiplied, the original matrix is reconstructed. One can reduce the dimensionality of the solution simply by deleting coefficients in the diagonal matrix, ordinarily starting with the smallest.

C. *Personalization and Collaborative filtering*

Traditional search engines are optimized to provide search results which the majority of users would agree to. However the latest tendency in the field has been to provide search results personalized to each user. There are two flavors to search results personalization, whereby in the first flavor one user would get a search result different from another based on his/her preferences/profile. In this case personalization is tightly coupled to search functionality and in the other flavor personalization is done by reordering the search results returned by the search functionality.

1) *Collaborative Filtering*

Collaborative filtering is a promising approach in deriving a profile for the user based on the users who have historically had similar tastes [6] and is extensively used in recommendation systems. Collaborative filtering overcomes one of the major limitations in other approaches, which is data sparseness. It has been very successful in both research and practice, and both in information filtering such as search results filtering as well as E-commerce applications [7].

Item based collaborative filtering is a collaborative filtering algorithm presented by Badrul Sarwar et al in [8] and it tends to overcome a number of limitations in the traditional collaborative filtering algorithms. The first and the foremost challenge is to overcome is the scalability problems faced by other collaborative filtering algorithms. Item based collaborative filtering algorithm avoids the bottleneck of searching for neighbors among a large user population of potential neighbors by exploring the relationship between items first, rather than the relationships between users. Preferences/scores are computed by finding items that are similar to other items the user has liked. Since the relationships between items are relatively static, item-based collaborative filtering is able to provide results with less online computation.

On the other hand SlopeOne algorithm is one of the simplest forms of non-trivial algorithms for collaborative filtering based on ratings/preference values. However it has been proven that this simple algorithm is on par with more complex and computationally expensive forms of collaborative filtering algorithms. In [9], the authors highlight the need to keep the algorithms simple in order to make the processing efficient as well as to enhance the scalability, i.e. it is not recommended to compromise the simplicity of the algorithm for minor increases in accuracy. Further, SlopeOne can integrate new ratings without any delay and these new ratings change the output instantaneously. Moreover, SlopeOne performs well when the user is fairly new to the system, i.e. a user with few ratings receives valid outputs from SlopeOne algorithm.

III. METADATA BASED INDEXING

The index created using extracted metadata is very much different from a usual index created by a search engine. This index is a sort of a semantic map representing the information acquired by metadata extraction process. Four types of data are considered in creating this index. They are social tags, topic tags, entities and relationships. These entities were extracted using the OpenCalais web service [10].

Social tags try to determine how particular content would be tagged. As an example, a document about various sports should be tagged with the social tag 'Sports'. Topic tags are the most relevant social tags in the document which OpenCalais identifies as the topic for the content. Most of the time it only identifies one topic per document. Entities are the locations, people, countries, etc. OpenCalais defines a set of such data that can be extracted, which appear in the indexed documents. Without indexing these as plain text, they are indexed in such a way that the original meaning is preserved. Relationships are the relations between various entities.

Main algorithm used for metadata based indexing extracts the data from a given local/web repository such as a folder or web site. In the case of a local folder, the folder structure is navigated recursively to obtain the files. The main types of file formats include .pdf files, .docx files, .doc files and .odt files etc so that the user does not have to carry out separate implementations for different file formats. Extracted text is then submitted to OpenCalais web server to get the metadata extracted. For handling limitations of the web server, a long text stream is divided in to chunks of meaningful data before submitting, typically lesser than a length of 30,000 characters.

The tags extracted are then used to create the index. For efficiency purposes, the main thread spawns separate worker threads to create the index. Therefore, other than the main thread, the index uses a thread pool to perform the indexing. Using a thread pool instead of creating individual threads is more efficient since the overhead is less.

As the thread pool builds, it populates the index with the necessary data and finally shuts down. Since the indexing process is a onetime process, and the index built should be persistent, it is stored in a database to be used as the base for conducting the search. If needed this index can be updated later to add more files/web pages.

Since OpenCalais is remotely hosted as a web service, there might be network issues in transferring the data between the applications. The exceptions and error messages from the web server are handled appropriately at the connection point to make sure the framework can recover from such issues.

After metadata are extracted from OpenCalais, those tags have to be stored in a semantic manner in the database. Jena [11] is used for this purpose. Using Jena, metadata tags are indexed in a unique way which is convenient to perform a search preserving the semantics of metadata tags.

RDF [12] structures metadata in the form of subject-predicate-object which is called a triple. The same concept is used when creating this index as well. First a model for the whole data store is created using Jena. Then the model can be updated by inserting triples to the model. Figure 1 shows how the metadata tags are indexed as a graph.

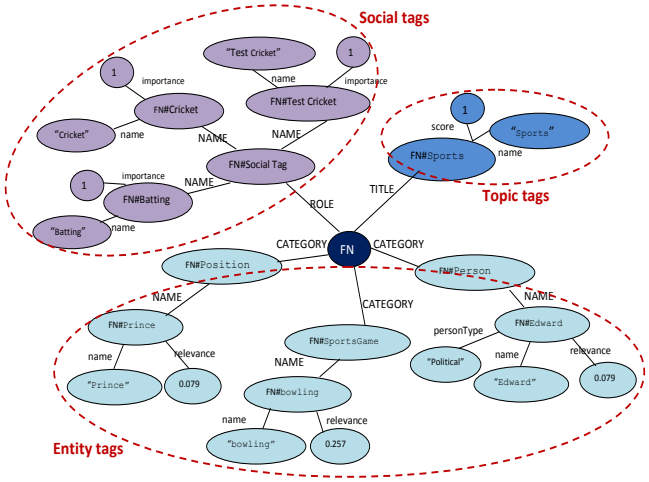

Figure 1 - Index diagram

In the model a graph is created for each file. The social tags of the file are added to the graph using the predicate VCARD.ROLE. VCARD is a constant class which consists of objects that represent all the definitions in the VCARD schema. Jena also enables users to create predicates with meaningful names. Then as shown in the graph, all the social tags have been added to the "Social Tag" node and their literal and importance values have also been added using predicates defined by the system. More important social tags have the value 1 and less important ones have the value 2. Likewise topic tags were also added to the model using the VCARD.TITLE predicate.

Names of all the nodes are in the URI(Uniform Resource Identifier) format. "FN" is the URL(Uniform Resource Locator) or the address to the resource and the string after "#" is the URN(Uniform Resource Name) or the name of the resource. Literals have only the string representation of the node and this was done intentionally for ease of querying and searching.

Entity tags are sent by OpenCalais after categorizing them in to predefined entities such as "Person", "Organization" etc. OpenCalais considers entity disambiguation when identifying and categorizing entities. Entity disambiguation is the process of resolving the identity of entities. For example "George Bush" and "George W. Bush" has to be identified as the same person. Those uniquely identified entities are given a relevance value which is based on the relevance of each entity to the document. It's a value between 0 and 1 where 1 is the value for most relevant entities. Also for some entities there are additional tags, such as "personType" and "nationality" for entities in the category of "Person".

The identified categories are added to the file using the property VCARD.CATEGORY and entities of each category is added using the predicate VCARD.NAME. Then for each entity the name, relevance and the additional tags are added with predicates defined by the system.

Likewise for each file, such a graph is created with social tags, topic tags and entity tags. This index is used to perform the metadata based semantic search.

IV. METADATA BASED SEMANTIC SEARCH

Search based on metadata can't simply use the keywords from the query and match them with keywords form the indexed data. This should consider the meaning of the data that are being search for, and also the meaning of the query as well. This can be done in several ways.

A. Conceptual Graph Matching

Semantic similarity can be considered as matching of objects and relations appropriately. This can be considered to be done using several criteria. They are; considering the similarity between nodes, considering the similarity between arcs (predicates) and/or considering the similarity between graphs [13]. These can be considered as pattern matching, and differ from granularity from each other.

Similarity between nodes can be determined with regard to the concept of semantic distance. Semantic distance is the distance between nodes. Therefore, the semantic similarity can be defined as the complement of the semantic distance.

Similarity between arcs is figuring out similarities between the relationships between nodes. Arc similarity can be compared by considering the type of the relationship, parent-relationships and child-relationships. But, it is not a simple procedure.

Similarity between graphs is the extent up to which a graph generated by the query can be matched with a sub graph of the index.

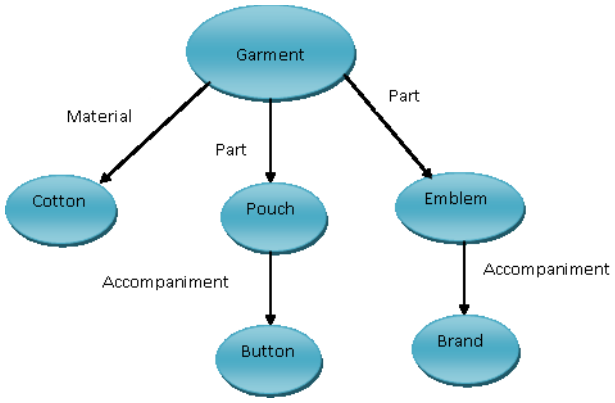

Figure 2 - Query Graph

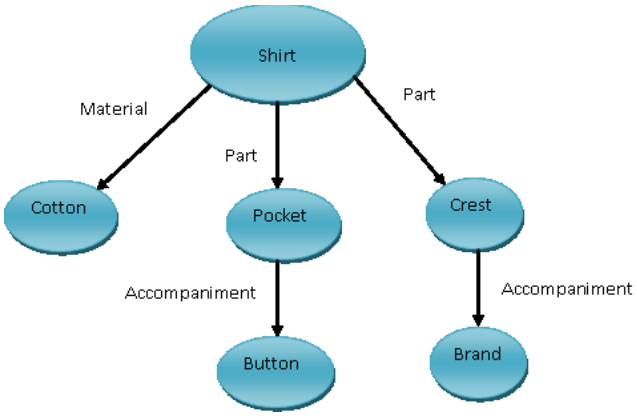

Figure 3 - Candidate result sub graph

B. *Complete and Minimal Answers*

Complete and minimal answers is another concept which is associated with RDF based data retrieval. This can be applied to a search based on an index built on RDF.

A complete answer is a result in the form of an RDF node which along with all its descendants contain all query terms. Ideally, the query terms should be the terms which are generated by a query analysis. However, if it's not feasible or if it's not possible to analyze the query, the query terms can be used as they appear.

A minimal answer is an answer such that,

$$\forall k \in Q: [v \text{ contains } k] \; OR \; [\exists u \text{ son of } v \; s.t \; u \text{ contains } k \; AND \; u \text{ is not complete answer}]$$

Here, k is the set of tags of the query, Q is the query itself and u, v are RDF nodes [14]. What is meant by this definition is that a minimal answer is a node, where the tags match the tags appearing in the sub tree rooted at the original node, and no node on this sub tree should be a complete answer to the query (with all tags in the query). Therefore, the search task is to get a set of such minimal answers, with a suitable score function to determine the most relevant results.

C. *Developed Algorithms*

iSeS framework uses several custom algorithms to deliver the required results. Specially for searching, the framework uses two such algorithms, which were designed and fine tuned by the iSeS team.

1) *Best Guess/ Entity Search Algorithm*

This is a typical algorithm which demonstrates the uniqueness of a search based on semantics rather than a search based on keywords only. This algorithm attempts to guess the best match for a given query. This does not give a list of most related documents as the final result of the search, but tries to suggest a set of entities which are related to a search query with the related scores. For an example, for an index created on cricketers, a query 'Sri Lankan bowler' might give the result 'Muttiah Muralitharan' as the best guess. But, since there can be only one or a very few results for a question type query; it is difficult to identify only the best matching result(s) separately. Instead this search algorithm defines a set of results, of which the first one is the best guess.

The implementation looks at the tags of an entity of the semantic map and tries to compare it with the search query. The entity which has been tagged with majority of tags related to the query can be selected as a good match. The algorithms require the user to specify a part of the query as the 'most important' part. Ideally this should happen automatically, but extracting the most important part of a query is another project by itself. Therefore, the easier option is to get the user to highlight a section of the query as the most relevant/important section.

After comparing the entity tags with the query, to get the best matching entity/ set of entities, it is important to specify a score to each entity. In determining the best guess(es), it is essential to take into account the relevancy of an entity to the document it's appearing in. For example, a sports document carrying a political figure might be less related to a query on politics than the same entity appearing in a document related to politics. Therefore, the scoring function needs to take this fact in to account other than the number of tags matched.

$$\begin{aligned} &Relevancy\ Score \\ &= \log\Big[\text{Relevancy determined by metadata extraction} \\ &* \text{metadata weight value} + \left[\frac{\text{matching tags in the entity}}{\text{total tags in the query}}\right] \\ &* \text{query tag weight value} + \left[\frac{\text{matching tags in the entity}}{\text{total tags in the entity}}\right] \\ &* \text{entity tag weight value}\Big] \end{aligned}$$

This function takes into account the relatedness of an entity to the appearing source document using the relevancy determined during metadata extraction and the proportion of the number of tags tagged in the entity to the total number of tags in the query. This approach enables to select the entities as the best results those which are most related to the source and the search query as well.

2) *Semantic Search Algorithm*

This is the semantic version of the usual keyword based search algorithm. Unlike being limited to comparing keywords in the search query and the index, the semantic nature of the search has been preserved. Thus, even though the indexed text does not contain any terms in the query itself, if that text source is relevant to the search, it might be returned as a result.

Initially the search was designed to query the RDF index each time the user enters a search query. But querying RDF index through Jena frequently was time consuming. Therefore we introduced three tables to the database which are being updated while the RDF index is created. These three tables contain the data required to do the calculations for the semantic search. This made the running time about 20 times faster.

The semantic search algorithm has two major steps. First the files are filtered by comparing the query tags and the social tags of files. Then the filtered files are assigned with a score and they are ranked based on that assigned score.

a) *Filter files based on social tags*

iSeS can be deployed in a file system or a web site which would be having hundreds and thousands of files in it. Since it is not viable to calculate scores to every file each and every time the user enters a query, there should be a filtering mechanism as the initial stage of the search algorithm. Social tag based filtering was used for this purpose. Social tags are the tags produced by the OpenCalais considering the context of the document. These literals might not be there in the text but would be the collection of words which best describes the context.

A file is selected as a filtered file if the social tags of that file match with at least one tag of the query. For this filtering social tags were chosen over entity tags since social tags are more semantically related to the document. Therefore this would filter the documents which are more semantically closer to the search query. These filtered set of files is then passed to the second step of the search.

b) Rank the filtered files based on score

Ranking of files is done based on a score assigned to each file. When assigning a score to files the entity tags are taken into account. This score has three components. One component is the similarity of pair of tags in a file that are matched with the query tags. Second component is the value assigned based on the number of tags matched in the document. Third component is a score based on the percentage of query tags matched with the tags in the document. In these calculations the "relevance" factor of each entity is given a very high importance. This is because relevance reflects how much that entity is semantically related to the document.

The logic behind the similarity calculation is to check whether the identified entity tags are closely related or not. This is quite closer to the proximity search in keyword based search where it gives a higher rank if tags are located close to each other. But in the semantic search the location of the tags is not considered but the semantic proximity is considered. If a considered node in the index is located near to another such node, those two nodes are considered as being semantically similar, and vice versa.

For each pair the "distance" is calculated first. Then the sum of "distances" is divided by number of pairs to take an average value. Similarity is the (1- ?distance). This "distance" value is the average of semantic remoteness and average relevance of the two tags. Figure 4 shows how to arrive at the remoteness value.

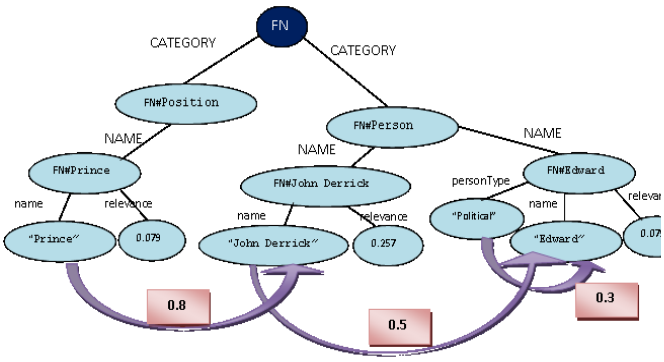

Figure 4 - Distance values

If the tags are the literals of same entity remoteness is 0.3, if in the same category remoteness is 0.5 and if they are in two different categories remoteness is 0.8. Similarity equation is as follows.

$$similarity = 1 - \frac{\sum(remoteness + average\,relevance)/2}{no.\,of\ pairs}$$

Since remoteness and average relevance values are always less than one, similarity is also less than one.

The second component of the score is the tag count score. This value increases when the number of tags matched to the entities of a file increases. Below is the equation to calculate this score.

$$tag\ count\ score = \frac{\sum relevance}{no.\,of\ tags}$$

Following equation is the score calculated for the third component which takes the percentage of query tags matched with the file in to account.

$$matched\ tags\ ratio = \frac{no.\,of\ matched\ tags}{no.\,of\ query\ tags}$$

Finally the score is calculated as follows.

$$score = \log_2(1 + (similarity + tag\ count\ score + matched\ tags\ ratio)/3)$$

This score is assigned to each file and ranked in the descending order of the score. Then according to the users preference the search results are sent either to the personalization module or to the user's application.

V. LSA BASED SEMANTIC SEARCH

We conducted a literature survey on the existing LSA implementations we can use for our project. Much of these implementations were proprietary. However the packages "S-Space Package" [15] and "Semantic Vectors" [16] are open source frameworks for developing and evaluating word space algorithms. These packages implement LSA and provide a comprehensive set of matrix utilities and data structures for extending new or existing models.

The packages are written in Java and define standardized Java interfaces for word space algorithms. Compared to "Semantic Vectors" and other existing frameworks, the S-Space Package supports a much wider variety of algorithms and provides significantly more reusable developer utilities for word spaces, such as tokenizing and filtering, sparse vectors and matrices, specialized data structures, and seamless integration with external programs for dimensionality reduction.

A. How LSA is being used in iSeS

The above mentioned LSA implementation is easily customizable to include word stemming and can be easily integrated into our requirements since not only can S-Space be used to get vector comparisons, but it can also be used to get similar words to a particular word of the corpus that is being used.

So by using this similar words interface we built an index where each significant word in the corpora is given a customizable number of similar words based on the LSA vector value for those words i.e. the more the vector similarity, the more the word similarity.

Then the iSeS framework would use the said words to compile a list of semantically close words for a given query and use those words to perform a normal keyword search on the index created using the Lucene keyword indexer to give the required search results. This keyword search is performed using the Lucene keyword based searching.

VI. PERSONALISATION

In providing users with personalized search results, two aspects were considered. Personalization model employed being the first is of high importance as it is key in making the personalization function highly generic. Personalization mechanism is the second aspect, which includes the algorithms used to personalize search results based on the personalization model chosen.

A. Personalization model

As highlighted, one of the major consideration is deciding on a personalization model was its genericness. Since iSeS is a framework, the models ability to be used in numerous scenarios was important. For example, personalization schemes such as those based on users' explicitly stated profile would not suit the purpose as they would restrict the use of iSeS to the application that collects such data.

The personalization model employed in iSeS is somewhat close to the model employed by recommendation systems. The model has users, items in search results and preferences for user-item pairs. In other words the model is a sparse matrix of preferences of users towards items in search results as graphically depicted in figure 5.

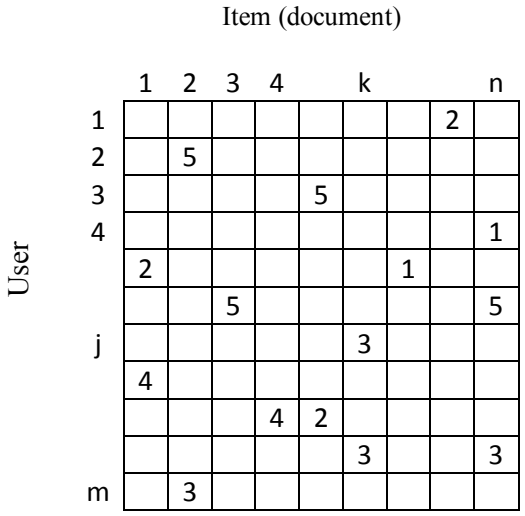

Figure 5 - User-Item matrix

The preference value can be anything with the notion of 'liking', such as explicitly stated preference of users towards items, number of times the user has visited the item etc. This quality has made personalization mechanism more generic, so that it can be applied in number of occasions.

B. Personalization Mechanism

iSeS uses collaborative filtering algorithms at the centre of its personalization module. Implementations based on Item-based collaborative filtering algorithm and slopeOne algorithm has been provided with iSeS as two algorithms present different advantages over each other. We reused Apache Mahout collaborative filtering library which comes with implementations of Item-based collaborative filtering algorithm and slopeOne algorithm.

With these algorithms, the personalization module infers preference values for each of the items in search results for the user who submitted the search query. These inferred preference values are stored with the search results and they are used to derive a final composite score, on which the ultimate ordering of search results is based on.

VII. RESULTS

The following results are the results obtained for the two search algorithms mentioned above, BestGuess Search and Semantic Search.

A. BestGuess Search

Under this technique, a set of queries which are related to a particular domain, but which do not have a specific answer were tested.

1) Example Domain: Business

The index for this domain was created using popular news site CNN's business news section on 03rd September, 2011.

TABLE 1
BESTGUESS RESULTS

Sample Query	Top Results
people related to economics extra tags: economics person	1. Tim Cook 2. Bernie Madoff 3. Steve Wozniak 4. Nizar Hani
world organizations extra tags: government, organization	1. European Union 2. International Monetary Fund 3. Natural Resource Defence Council
people related to entertainment extra tags: person, entertainment	1. Katt Williams 2. Josh Hutcherson 3. Melanie Brown 4. Jennifer Lawrence

All these results carry entities, people or organizations which are very much relevant to the query. These results are difficult to obtain using a normal keyword based searching mechanism, since the queries used might not have the actual mentioning of the entities related.

B. Semantic Search Algorithm

The same CNN's business news index was used to test the search queries for the Semantic Search algorithm. Semantic Search gives the address of the files as the search results. In keyword search also the search results are the addresses to files. Therefore in order to observe the accuracy of the Semantic Search, the results for the same search query for the same index from the Semantic Search and keyword search are compared in the Table 2 and Table 3.

1) Search query – "modern business concepts"
Extra tags: economy, technology

TABLE 2
SEMANTIC SEARCH RESULTS FOR THE QUERY "MODERN BUSINESS CONCEPTS"

Semantic Search results	Keyword search results
First result:	
Opinion: Apple rivals likely to prevent monopoly - CNN.com	When social media 'hinders' revolution - CNN.com
This web page does not contain the word "concept" but this comes as the first result since the content is about a modern business concepts.	Though this comes as the top result, the news is not about a modern business concept.
Second result:	
Steve Jobs: From college dropout to tech visionary - CNN.com	Tensions rising in Yemen; pro-government gunmen gather outside capital - CNN.com
This is about Steve Job and his business concepts.	Not related to business concepts.
Third result:	
When it comes to presentation, Mark Zuckerberg is no Steve Jobs - CNN.com	Libya's other wealth: Archaeological treasures - CNN.com
This article contains about Mark Zuckerberg and Steve Jobs and their business ideas.	This article does not relate to business concepts.

None of the above web pages have the keyword "concept". But in Semantic Search the relevant results are ranked at the top since the search is based on the semantics of the content.

2) Search query – "news related to entertainment"
Extra tags: entertainment, news

TABLE 3
SEMANTIC SEARCH RESULTS FOR THE QUERY "NEWS RELATED TO ENTERTAINMENT"

Semantic Search results	Keyword search results
First result:	
'Hunger Games' cast: Cheat sheet for stardom - CNN.com	Iran's nuclear plant connects to electric grid, the country says - CNN.com
This article is related to the movie "The Hunger Games" therefore this is a relevant result for the search query.	This article is about a nuclear power plant and not related to entertainment at all.
Second result:	
Katt Williams explains apology for Mexico remarks - CNN.com	Nice guys earn less, study finds - CNN.com
Katt Williams is a comedian therefore this article is related to entertainment.	This has no content related to entertainment.

These web pages also do not contain the keyword "entertainment" in their articles. But the Semantic Search has showed very good accuracy in predicting what the user needs to search.

VIII. Combining resutls

As described in earlier sections, iSeS framework gives a relevancy value for each and every item search results (i.e. webpage or document) based on their semantic closeness to the query as well as a preference value to the each item based on the expected preference by the user towards the item. iSeS then calculates a composite score to be used for the final search results ordering. This composite score is calculated as the weighted average of relevance and preference scores described earlier and the weights used for this purpose are configurable. At the moment, 0.75 and 0.25 are used as weights for relevance and preference scores respectively and since these weights are configurable, one can adjust the relative importance of semantic closeness to the query and the personalized nature of the search results.

IX. Conclusion

iSeS provides a framework to integrate a personalized semantic search engine to their website/data repository. We believe that it can remarkably improve the results of the search where the traditional keyword based searching fails in obtaining useful results. Further integration of personalization will be useful for the client who implements iSeS on their own application which has multiple users using the search feature. This remarkably cuts down the time and money to come up with a completely new search mechanism for their respective projects. It is the iSeS team's wish that this project bears a catalytic impact on searching based on semantics and will be even a tiny help to arrive at the concept of semantic content management in the near future.

X. Acknowledgment

We would like to express our sincere gratitude to Dr. Shehan Perera, the main supervisor of project iSeS, who was always behind us giving us valuable feedback and comments on the work. Mr. Sajith Vimukthi of Zone 24x7, the project co-

supervisor, was with the team right from the beginning, from the time of inception of the idea itself. Even along with his very busy schedule, he organized meetings with the team and voiced his opinions in moving forward. Also the team would like to thank the project co-coordinator Dr. Shantha Fernando for the support and guidance rendered throughout the project time period.

REFERENCES

[1] F. Alkhateeb, A. Alzubi, I. A. Doush, S. Aljawarneh, and E. Al Maghayreh, "Extracting authoring information based on keywords and semantic search," in *Proceedings of the 1st International Conference on Intelligent Semantic Web-Services and Applications*, New York, NY, USA, , p. 1:1–1:6, 2010.

[2] J. Peckham and F. Maryanski, "Semantic data models," *ACM Computing Surveys (CSUR)*, vol. 20, p. 153–189, Sep. 1988.

[3] T. K. Landauer, P. W. Foltz and D. Laham, "An Introduction to Latent Semantic analysis," *Discourse Processes*, vol. 25, no. 2, p. 259-284, 1998.

[4] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391-407, Sep. 1990.

[5] S. Hammarling, "The singular value decomposition in multivariate statistics," *ACM SIGNUM Newsletter*, vol. 20, p. 2–25, Jul. 1985.

[6] Konstan, J., Miller, B., Maltz, D., Herlocker, J., Cordon, L. and Riedl, J. "GroupLens: Applying Collaborative Filtering to Usenet News" *Communication of the ACM*, 40(3), pp. 77-87, Mar. 1997.

[7] Sarwar, B., Karypis, G., Konstan, J. and Riedl, J. "ItemBased Collaborative Filtering Recommendation Algorithms" in *Proceedings of the 10th international conference on World Wide Web*, Hong Kong, Hong Kong, 2001, pp. 285-295.

[8] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the tenth international conference on World Wide Web - WWW '01*, Hong Kong, Hong Kong, 2001, pp. 285-295.

[9] D. Lemire and A. Maclachlan, "Slope One Predictors for Online Rating-Based Collaborative Filtering," *In Siam Data Mining(SDM05*, p. 21--23, 2005.

[10] "Home | OpenCalais."[Online]. Available: http://www.opencalais.com/. [Accessed: 03-May-2011].

[11] "Jena Semantic Web Framework." [Online]. Available: http://jena.sourceforge.net/. [Accessed: 26-Aug-2011].

[12] "RDF - Semantic Web Standards."[Online]. Available: http://www.w3.org/RDF/. [Accessed: 03-May-2011].

[13] H. Z. Jiwei, H. Zhu, J. Zhong, J. Li, and Y. Yu, "An Approach for Semantic Search by Matching RDF Graphs," *In Proceedings of the special track on semantic web at the 15th International Flairs Conference(Sponsored by AAAI, 2002.*

[14] "Semantic Search - Algorithmic Problems Around the Web _8." [Online]. Available: http://www.docstoc.com/docs/75839792/Semantic-Search---Algorithmic-Problems-Around-the-Web-_8. [Accessed: 26-Aug-2011].

[15] D. Jurgens and K. Stevens, "The S-Space package: an open source package for word space models," *Proceedings of the ACL 2010 System Demonstrations*, p. 30–35, 2010.

[16] D. Widdows and K. Ferraro, "Semantic Vectors: a Scalable Open Source Package and Online Technology Management Application," in *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, 2008.