Ontology Extraction from text documents by Singular Value Decomposition

ADMI 2001

Research students: Govind Reddy Maddi, gmaddi@cs.bowiestate.edu
Chakravarthi S Velvadapu, cvelvada@cs.bowiestate.edu
Faculty Advisers: Sadanand Srivastava, ssrivas@cs.bowiestate.edu
James Gil de Lamadrid, gildelam@cs.bowiestate.edu

Bowie State University, 14000, Jericho park road, Bowie, MD, 20715.

****

Abstract:

Due to the abundance of unstructured data available today, it has been interesting to research for finding an automated way to retrieve information, or to respond to a structured or an unstructured query. Domain specific information is useful in processing natural language, information retrieval and systems reasoning. We need more than a simple list of key words for all but very elementary processing.

It is very important to build document ontology to analyze a large collection of documents. Ontology of concepts, terms and the relations between concepts is more useful for complex reasoning. This type of ontology if automatically compiled, the user will be freed from all the tedious task of construction and also the ontology will be produced at a greater speed. Basically this project is about constructing a domain specific ontology, by reading the collection of related text documents, and extracting ontological information statistically. In particular, this system determines word frequencies, which are formed into a frequency matrix. Singular Value Decomposition is performed on the frequency matrix, and the resulting matrices are used to determine statistical relationships between documents and terms. These relationships are then used to build an ontological graph. The ontology graph can be viewed and manipulated with graphical user interface (GUI), or loaded into user code. On the whole our system provides a procedure of generating richer domain specific knowledge, with minimal user interaction.

1. Goal

The goal of this project is to construct a system capable of reading a standard text file documents, performing semantic analysis on those documents and generating a useful ontology for that set of documents with little human intervention. This process should be automated as much as possible. Methods used for extracting ontological information might include statistical methods and a small amount of user feedback. This project is coded in Java language.

2. Introduction

In our context, Ontology , is specification of a conceptualization. That is, ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. Document Ontology , is extracting concepts from a set of documents and identifying relationships of these concepts with different individual terms, i.e. individual words in that particular set of documents.

Domain specific ontology s are useful in systems involved with artificial reasoning, and information retrieval. Ontologies give such systems a vocabulary of terms, and concepts relating one term to another.

Crafting ontology is a tedious process. In process of relieving the user from this tedium, we have examined the automation of the process for constructing ontologies. Our system constructs a domain specific ontology from text documents, presented to it as a training set. The documents are read, processed, and a graph-structured ontology is produced. This ontological structure can be viewed and modified with graphical user interface (GUI).

In the process of achieving this goal, contemporary statistical methods of information retrieval such as Boolean, extended Boolean and vector space approaches have been studied [Green grass 1997]. Of all these approaches vector space approach has been found to be the most efficient method. It describes each document as a set of terms. This set defines the document space such that each distinct term represents one dimension in that space [Salton, 1989]. Each document in document space is defined by the weights, or in other words frequencies, of the terms that represent it.

Another vector space approach called Latent Semantic Indexing (LSI) [Derwester, 1990] attempts to catch term-term statistical references by replacing the document space with lower-dimensional concept space. LSI accomplishes this by using Singular Value Decomposition (SVD), a method of matrix decomposition. The effectiveness of SVD as compared to other techniques is described in [Harman, 1995].

In our present work we have concentrated on statistical analysis method, as compared to heuristic and rule based methods because of its simplicity and because it is based on fairly precise mathematical foundation.

This project describes a system of constructing a domain specific ontology, by reading a collection of related text documents, and extracting ontological information statistically, which is result of the procedures Pre-processing, Normalization, LSI by SVD, Graph construction and Graphical user interface (GUI) construction, which are very important and have been studied in depth.

Pre-processing is a process in which we extract the meaningful terms and count their frequencies while reading the input text file. Here several formatting procedures are performed on the document text, in order to make sure that the computed statistics are meaningful. These procedures are concerned with common roots and eliminating the marker words which are semantically insignificant.

Normalization is a process in which we calculate the normalized weight of each word that we have obtained from preprocessing. Here the preprocessed document is analyzed in terms of correlation of words, the frequency matrix known as Term-Document matrix is produced as result of the analysis.

This Term-Document matrix is then decomposed into three matrices like Term Vector (U), Singular Matrix (S), and Document Vector (V), using Latent Semantic Indexing (LSI) based on SVD. Then, after minimizing the size of all this matrices, the term matrix, which is one of the three matrices, is decomposed into vectors of related terms known as concepts. Concept is set of related terms.

Building document ontology is essentially building concept nodes and term nodes from the term matrix (U) and the document matrix (V), which we have obtained from SVD. A concept node represents a concept and contains information about its concept name, terms that belong to that concept, and their weights in that concept.

In graph construction, concept vectors are used to construct a bipartite graph, which is used to show the relationship between different terms and concepts, where the concept nodes are taken on one side and term nodes are taken on the other side.

To let the user easily interact with the system, a graphical user interface (GUI) is built. The GUI allows the user to easily view and manipulate the ontology. The GUI consists of two lists, one of which contains the list of concepts and the other contains the list of terms, and a panel that displays the bipartite graph drawn to show the relationship between terms and concepts by a line joining them and it also has many other functions.

4. Architecture Of Document Ontology Extractor

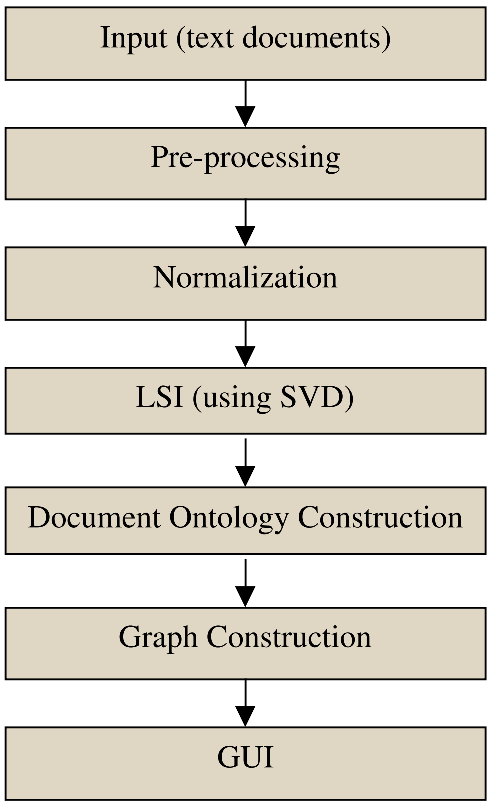

Figure 1. DOE Architecture

4.1 Input

The input for this system is set of text documents. The list of text documents, for which the ontology is to be built, is written to a text file and that text file is the input to our system. New files to this list can be added by typing directly onto input text file or from the Graphical User Interface (GUI).

4.2 Tasks to Perform

To accomplish this goal we should perform the following tasks:

- Pre-processing
- Normalization
- Latent Semantic Indexing based on SVD.

4.2.1 Pre-processing

Pre-processing is a process in which we extract meaningful terms and accurately count their frequencies while reading the input text file. Pre-processing involves the reformatting, or filtering, of a text document, to facilitate meaningful statistical analysis. In doing the Pre-processing on a text document the system will follow the following steps:

§ Numbers and punctuation marks standing alone should be cut; for example, "55", "720", "90", "4", " - "

§ It is necessary to cut punctuation marks at the end of the words; e.g., the program should not distinguish the words "school" and "school," "theaters" and "theaters," from each other.

§ The Program should not be sensitive to the case and all the words should be examined as low-case words; e.g., if the text contains words "school" and "School", the frequency of the word "school" in the output of the program should be two.

§ It is important to exclude "stop" words - articles, preposition, unions etc. - which cannot be used as descriptors of any document (words like "and", "the", "when", "though", "for" etc.). To achieve that goal, we prepare a "stop list" of such "noise" words (now approximately 250 words) in additional file. The program construct s a hash-table based on that list so that the current token in input file is compared with the content of this hash-table.

§ One more aspect of pre-processing is correct counting of the words in specific grammatical forms (irregular verbs, nouns of Latin origin etc.) We prepared one more auxiliary file which produced another hash-table to replace derivative forms with original ones;°e.g.°° "broke" - "break",°°° "broken" - "break",° "phenomena" - "phenomenon" .By using this procedure for the text containing words "break" and "broken" the program gives a frequency of two for the word "break".

§ And finally the last procedure of the pre-processing stage - "stemming". The objective is to eliminate the variation that arises from the occurrence of different grammatical forms of the same word (e.g., "president" - "presidential", "toured" - "tour", "worked" - "workers "etc.).

To illustrate the Pre-processing in short, consider the following text:

"Of 36 previously broken parts, 16 will probably break again. Rules like this worked before, and rules like this will probably work again."

After the Pre-processing the above sample text will be as,

"previously broken parts will probably break again rules like this worked before rules like will probably work again."

Using an article in *Washington Post* as an input textual file we tested the effectiveness of pre-processing procedures further. The article is the one In Appendix.

Output for the article in Appendix A, for the words occurring in the above text°for more than once after Pre-processing, is as in Table below.°

3 - opened°	2 - watts°	2 - conference°
3 - closed°	2 - make	2 - chief°
2 - prayer°	2 - today°	2 - age°
4 - try°	4 - help°	3 - white°
2 - constructor°	2 - disadvantaged°	2 - house°
3 - carry°	2 - skills°	3 - work°
2 - lake°	3 - high°	2 - force°
2 - emotional°	2 - jobs°	2 - say°
3 - relations°	2 - new°	2 - announce°
3 - kind°	2 - magic°	2 - initiative°
2 - develop°	2 - johnson°	2 - partnership°
2 - associate°	2 - star°	2 - watch°
2 - write°	5 - south°	2 - side°
5 - los°	3 - inner°	3 - take°
5 - angeles°	3 - cities°	2 - damaged°
3 - day°	2 - movie°	2 - economic°
4 - president°	2 - theater	2 - see°
9 - clinton°	3 - tour°	3 - banks°
3 - invest°	2 - transportation°	2 - loans°
2 - poorest°	3 - care°	2 - residents°
4 - area°	4 - academy°	2 - open°
2 - country°	2 - program°	2 - phoenix°
2 - recover°	4 - school°	2 - producer°
2 - dead°	2 - black°	2 - plant°
5 - riots°	2 - facility°	3 - tortilla°
5 - people°	2 - students	3 - blessed°
2 - visit°	2 - percent°	
2 - community	2 - go	

4.2.2 Normalization:

Normalization is a process in which we calculate the normalized weight of each word that we have obtained from preprocessing. In Normalization we are primarily concerned with establishing word correlation. We base this on word frequency. The first step in normalization is then to determine the number of occurrences of each term in the document, which is obtained from Pre-processing. We then calculate the weight of each term, by the following formula:

$$W_{i,k} = \frac{frequency_{i,k}}{\left(\sum_{j=1}^{n_k} frequency_{j,k}\right)}$$

where $W_{i,k}$ is the weight of i^{th} term in k^{th} document and n_k is the total number of terms in that document. This weight corresponds to term weight in a document. But this term weight should be normalized for the entire set of documents, not for only one document. This is called as normalizing the term weight.

In the next step of Normalization, the weights for each individual document are combined into normalized analysis of the whole collection of documents. To do this we must take into account the fact that a term may have a large weight simply because the document in which it occurs is small, rather than because it occurs frequently throughout the document collection. To eliminate this problem, the normalized weight of a term is calculated as

$$\text{NormalizedWeight}_{i,k} = \frac{W_{i,k}}{\sqrt{\sum_{j=1}^{n_k} W^2_{i,k}}}$$

This process is a fairly standard normalization for document length, as explained by Greengrass [3].The Normalized output of the words occurring more than twice, in the tested document of Washington post are shown in Table below.

Frequency - Word - Weight °

5°	°°° los°	° 0.17937425°
5°	°°° angeles°	° 0.17937425°
3°	°°° day°	° 0.107624546°
4°	°°° president°	° 0.1434994°
9°	°°° clinton°	° 0.32287365°
3°	°°° invest°	° 0.107624546°
4°	°°° area°	° 0.1434994°
5°	°°° riots°	° 0.17937425°
5°	°°° people°	° 0.17937425°
4°	°°° help°	° 0.1434994°
3°	°°° high°	° 0.107624546°
5°	°°° south°	° 0.17937425°
3°	°°° inner°	° 0.107624546°
3°	°°° cities°	° 0.107624546°
3°	°°° tour°	° 0.107624546°
3°	°°° care°	° 0.107624546°
4°	°°° academy°	° 0.1434994°
4°	°°° school°	° 0.1434994°
3°	°°° white°	° 0.107624546°
4°	°°° million°	° 0.1434994°
3°	°°° work°	° 0.107624546°
3°	°°° take°	° 0.107624546°
3°	°°° banks°	° 0.107624546°
3°	°°° tortilla°	° 0.107624546°
3°	°°° blessed°	° 0.107624546

4.2.2.1 Creating Term-Document matrix

At this stage we can represent a textual document as a set of meaningful normalized terms. The normalized term weights, collectively form the matrix **W**, where $\mathbf{W}_{i,k}$ **= NormalizedWeight**$_{i,\,k}$**.** This matrix is referred to as term-document matrix. A term-document matrix contains terms as rows and documents as columns.

In this step, we create term-document matrix that describes the occurrence of meaningful terms in each document of the collection. To create such a matrix, we keep track of different meaningful terms occurred in different documents. While reading new text document, new terms are added into matrix as rows, their frequencies are added in the respective columns. While reading new document the system should check whether these new terms are grammatical forms of some previous terms (i.e. eliminating stemming words).

The output of this process is a term-document matrix containing distinct, meaningful terms in the entire collection as rows, and documents in the collection as columns. For a collection of hundred documents, assume that we obtained two hundred and fifty different meaningful terms. For this collection, the term-document matrix would be a 250 X 100 matrix.

4.2.3 Latent Semantic Indexing

At this stage we wish to determine a set of concepts from the term-document matrix, where a concept is defined as set of related terms. We accomplish this by using a method called Latent Semantic Indexing (LSI), which primarily involves decomposing the matrix **W** using Singular Value Decomposition (SVD).

Latent Semantic indexing is a statistical method that links terms into a useful semantic structure with out syntactic or semantic natural language analysis and without manual human intervention. By using this method each document is represented not by terms but by concepts that are truly statistically independent in a way that terms are not.

Terms cannot be used as descriptors of a document, as it has certain drawbacks such as the assumption that the terms are independent. But some terms are likely to co-occur in different documents, which should not be considered as independent terms. LSI concepts are described as information rich by Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., Harshman, R. et al.[2].

Latent Semantic Indexing uses Singular Value Decomposition (SVD) to accomplish its goal. LSI is described thoroughly by Berry et al.[1]

4.2.4 Singular Value Decomposition

Singular Value Decomposition is well-known matrix decomposition method. It decomposes the matrix **A**, i.e. the **m x n** term-document matrix, with m terms, and n documents, as $\mathbf{A} = U*S*V^T$.

where **U** is **m x r** matrix, called the term matrix, **V** is **r x n** matrix, called the document matrix, and **S** is **r x r** diagonal matrix containing singular values of **A** in its diagonal in descending order. In this decomposition, the singular values **_i** corresponds to the vector $\mathbf{u_i}$, the i^th^ column of **U** and $\mathbf{v_i}$, the i^th^ row of **V**. Without loss of any generality we can assume that the columns of **U**, the rows of **V**, and the diagonal values of **S** have been arranged so that the singular values are in descending order, moving down the diagonal.

The decomposition of matrix **A** is as shown in the figure2.

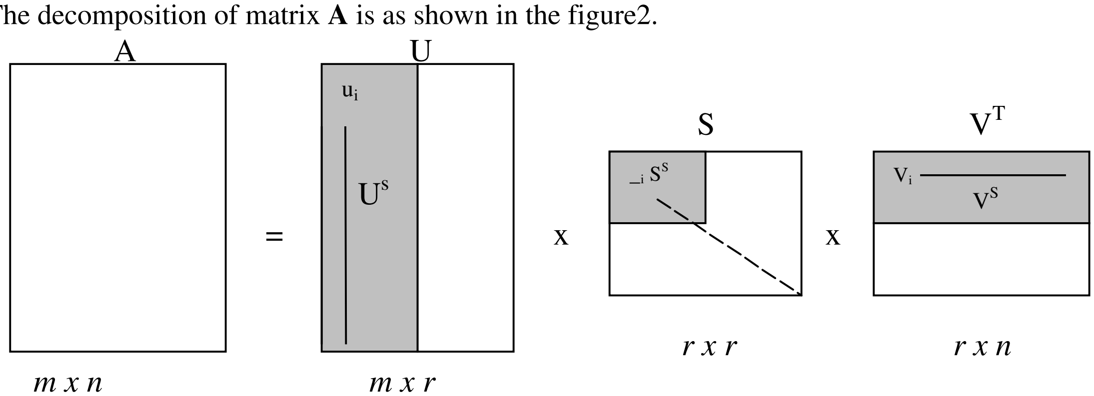

Fig. 2. Singular Value Decomposition and Latent Semantic Indexing

In LSI we form a new matrix $\mathbf{A}^s = U^s*S^s*V^{sT}$

Where $\mathbf{A}^s$ is derived from **A** by removing all but the largest **s** singular values, $\mathbf{U}^s$ is derived from **U** be removing all but the **s** columns corresponding to the remaining singular values, and $\mathbf{V}^s$ is derived from **V** by removing all but the **s** corresponding rows, where **s <= r**. As discussed by Nicholos and Dahlberg[5], and Derwester et al.[2], $\mathbf{A}^s$ is an approximation of **A**, with increasing accuracy as **s** approaches **r**. in our approach to LSI, we remove all singular values from **A** which fall below a threshold percentage of the largest singular value, _1.

The matrix $\mathbf{U}^s$ is an **m x s** matrix representing correlations between terms in the document collection. Each column of this matrix, $\mathbf{u_i}$, is a vector, which we consider to represent a concept. The elements of $\mathbf{u_i}$, is a vector, which we consider to represent a concept. The elements of $\mathbf{u_i}$, give the correlation of terms to the concept. The frequency of occurrence in the documents is represented by the value _i.

Detailed information about Singular Value Decomposition can be obtained from SVD-package Berry[1] and Spotting Topics with the SVD, Charles Nicholas and Randall Dahlberg et al.[5].

5. Constructing Document Ontology

Constructing document ontology is essentially building concept nodes and term nodes from term matrix (U) and document matrix (V), which we have obtained from SVD.

A concept node represents a concept and contains information about its concept name, terms that belong to that concept, and their weights in that concept. The name of a concept is generated automatically and is a hyphenated string of the five most frequent terms in that concept. Each column in document matrix (U) corresponds to a concept node.

A term node represents a term and contains information about its term name, concept that it belongs to, and its weight in different concepts. The name of a term is generated automatically and is simply the term itself. Each row in document matrix (U) corresponds to a term node.

6. Graph Construction:

Our ontology is a bipartite graph. We have two node types: concept nodes, and term nodes. Bipartite graph is used to show the relationship between different terms and concepts. In graph construction, concept nodes are taken on one side and term nodes are taken on the other side. Concept nodes are connected to term nodes, but are not directly connected to other concept nodes. Term nodes are connected to other term nodes, only by being connected to a common concept node. An example is shown in Fig.3.

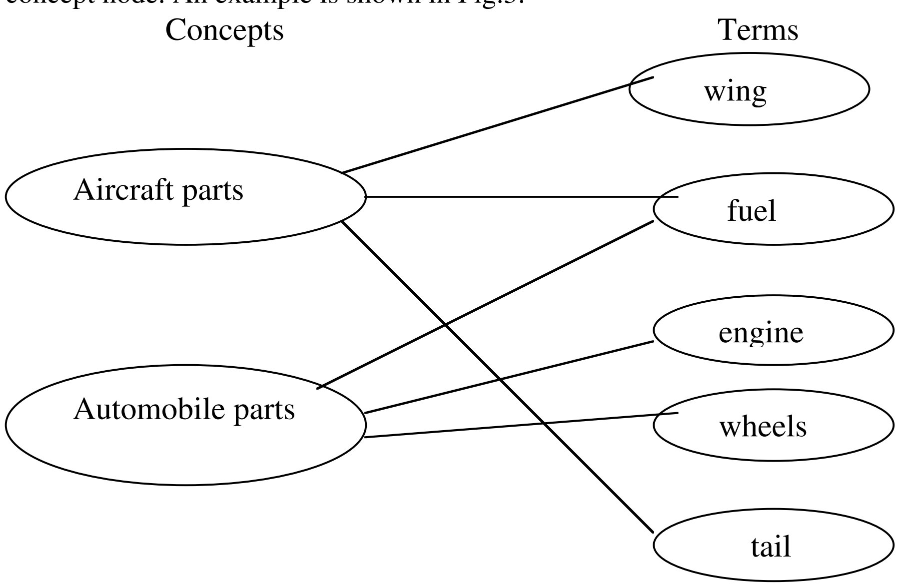

Fig. 3. Example, bipartite ontology graph.

The ontology graph is constructed from the matrix **U** and a list of term names. From the vector $\mathbf{u_i}$ we eliminate low correlation terms, by zeroing out elements in $\mathbf{u_i}$, which fall below a certain threshold percentage of the highest correlated term in $\mathbf{u_i}$. As an example if $\mathbf{u_i} = [0.4, 0.9, 0.7, 0.3]$ and the threshold is 50%, we would zero out all elements not within 50% of 0.9, resulting vector will be [0.0, 0.9, 0.7, 0.0]. This process is designed to eliminate terms that are, only weakly associated with the concept.

The resulting modified $\mathbf{u_i}$ vector becomes the template for the construction of a concept node. The concept node is connected to all terms with non-zero elements in the modified $\mathbf{u_i}$ vector, and term nodes are constructed only for terms, which are connected to some concept node.

All nodes are given names. Term nodes are simply named using the term name. Concept nodes are constructed automatically. Currently we generate a concept name consisting of a hyphenated string of the five most highly correlated terms in the concept vector. Since this does not produce excessively intuitive concept names, the user should have the ability to modify the concept name, to a more meaningful name, using the graphical user interface (GUI).

7. Ontology GUI:

The ontology GUI allows the user to easily examine and manipulate the ontology. The GUI contains different components like the bipartite graph display, the concept list, the term list, the current documents list, and also it has file, update and modify menus, which allows the user for doing different kind of things which are explained later in detail.

The graph displays a graphical representation of a bipartite graph to the user. Nodes arranged in two columns: the concept node column, and term node column. Connections between columns are shown as connecting lines. A GUI is as shown in Fig4.

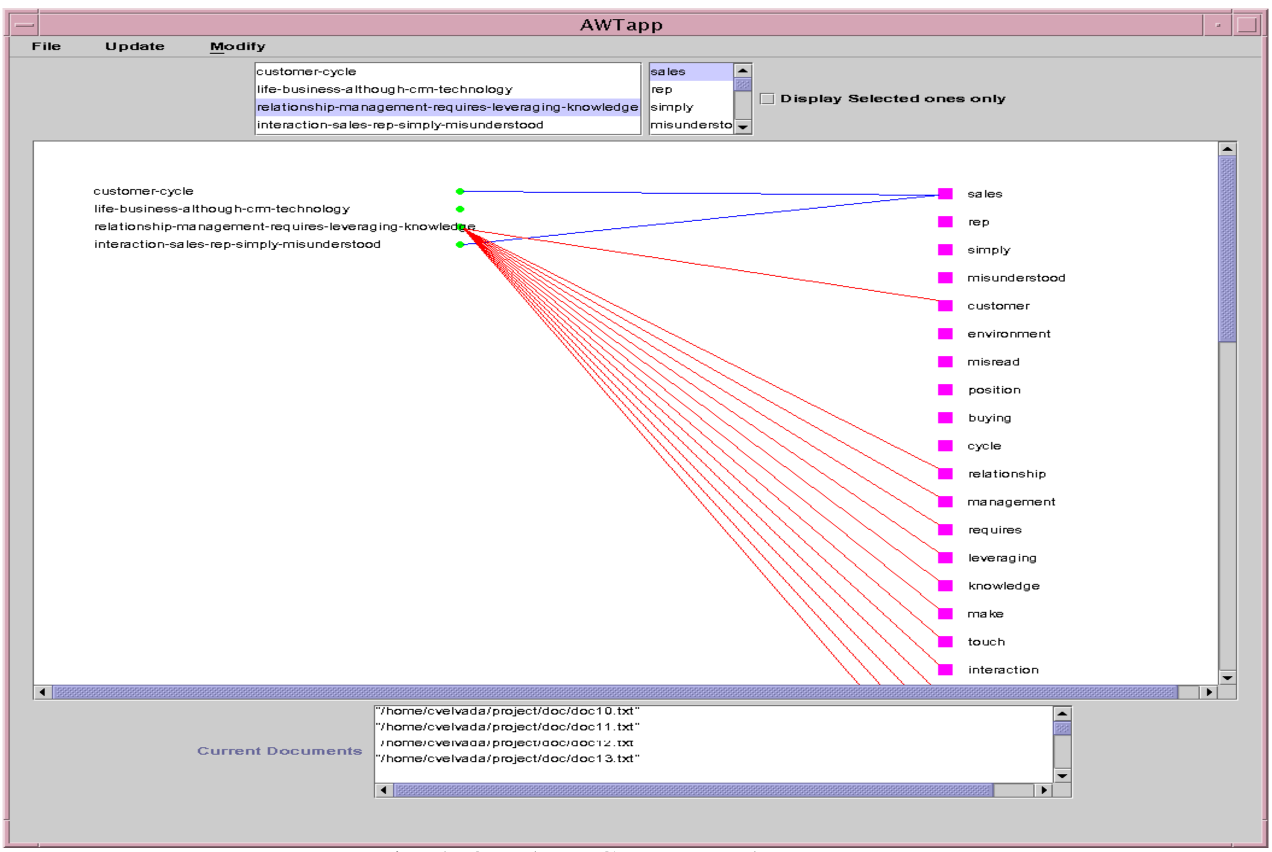

Fig. 4. Ontology GUI screen layout

Because of the potentially large size of the graph, a display of all connections would be too busy. While our system does allow the user to display all connections, the user can choose to display only a selected subset of edges. The user specifies this subset by selecting particular concepts and terms from the concept list and term list, respectively.

The GUI then displays only edges connected to the selected concepts and terms. This is shown in GUI in Fig.5

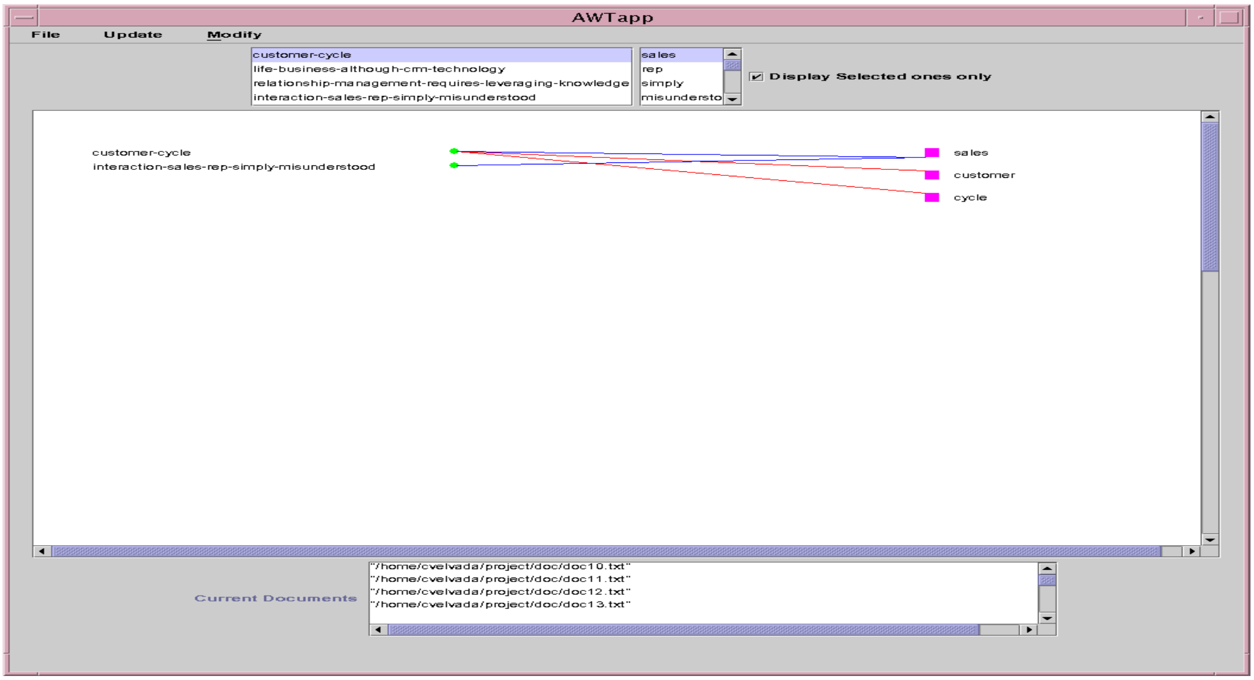

Fig.5. GUI. Displaying only the selected terms and concepts.

7.1 Different Menus in Ontology GUI

The different menus in Ontology GUI are the File menu, Update menu, and Modify menu; these menus are for doing different kind of operations.

7.1.1 File Operations menu:

As we know building ontology is a time consuming and expensive procedure, because it is involved with lot of mathematical computations. For avoiding all this overhead, when the user builds the same ontology over and over, the GUI lets the user save his work for future reference with some file operations such as New, Open, save, saveAs, Close and Exit. Each of the file operations is briefly described below.

New — It opens a new blank fresh ontology.

Open — It opens a previous ontology, which was already saved.

Save — It saves the changes made in already saved ontology, or saves a new built ontology for future references.

SaveAs — It saves the present ontology with any name for future references.

Close — It closes the present ontology.

Exit — It exits the user from the system.

The ontology files are saved with an extension .ont . The information such as names of the files

in

w

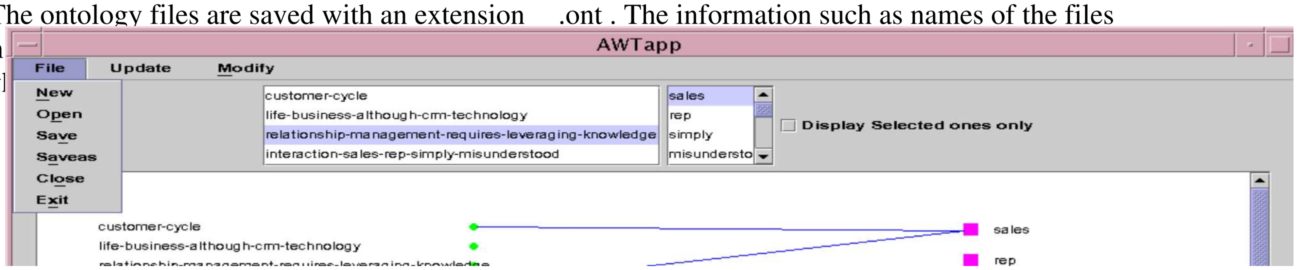

Fig. 6. Ontology GUI showing the File operations menu

7.1.2 Ontology Updates menu:

In the ontology Updates menu the users can add or delete documents to/from the present ontology and also the user can do different changes like ChangeSVDthreshold, and ChangeConcthreshold.

The user can change SVD threshold or concept threshold. SVD threshold controls **s** largest singular values that will be selected from **S**. the default value is 50% i.e. only those singular values, that are greater than 50% of the highest singular value, are selected. The user can change this default value. Concept threshold controls the number of terms related to a concept based upon the term weight. The default value is 50% i.e. only those terms, whose weight is greater than the 50% of the highest term weight in that concept, are selected. The user can change this default value.

The Document ontology may already exist for a collection of documents and at some time later the user may want to add some more documents to that existing ontology. To support all these operations, the Ontology Update menu in GUI has the following items.

Add — This command adds a new document to the ontology. The user has two options of adding new document viz. default build method and Fold-In method. The user can choose not to build right away and some time later can build ontology using default method or fold-In method from Update menu. This method throws an error message if the user tries to add a document that is already present in the collection.

To add a new document the user should,

- Click on Add menu item from Update menu
- The name of the file to be added should be selected from file chooser popup menu, which appears after clicking on Add menu item

- The build method type should be selected i.e. default build or fold-In build. The user can choose not to build right away. In this case, the file name is saved in a temporary file and the user can add these files in future by using default build or fold-In build.

Delete — This command deletes a document from ontology. Throws an error message if the user tries to delete a document that is not present in the ontology.
To delete a document from the ontology, the user should

- Click on Delete menu item from Update menu
- The name of the file to be deleted should be selected from file chooser popup menu, which appears after clicking on Delete menu item
- The user should select whether to build right away or not. If he/she chooses yes , the file will be deleted and GUI will be updated otherwise the file won t be deleted and GUI will remain unchanged.

ChangeSVDThreshold — This command changes the default value of SVD threshold and the user should choose between building right away with new threshold value and ignoring the change in threshold value.

ChangeConcThreshold — This command changes the default value of concept threshold and the user should choose between building right away with new threshold value and ignoring the change in threshold value.

DefaultBuild — This command builds a new ontology for the collection of files that are written in the input text file. The ontology Update menu in GUI is shown in figure 7.

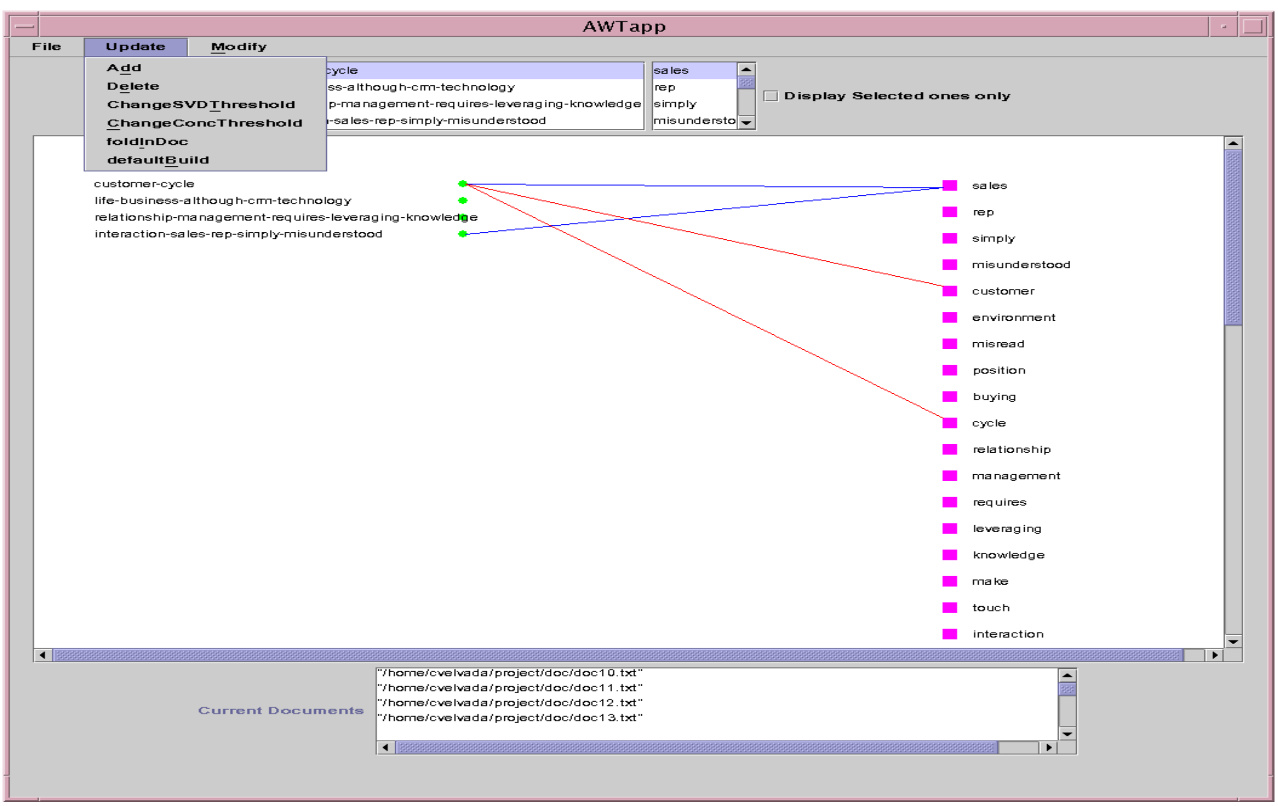

Figure 7 GUI, showing Ontology Update menu

7.1.3 Ontology Modifications:

The concept name is generated automatically and is a hyphenated string of the five highest frequent terms in that concept. The user may want to change this name, as the new name may give more meaning to that concept. The user may also want to delete a term from the present ontology. The user may also think that the Ontology would have been better without the last modification. To support these operations, Ontology Modifications menu has the following items

Rename — This command renames a selected concept

DelTerm — This command deletes a selected term

Undo — This command undoes the last modification and returns the system to the previous state

The modifications that can be undone are adding a document, deleting a document, changing SVDThreshold, changing ConcThreshold, renaming a concept and deleting a term. The modification menu item in GUI is shown in figure 8.

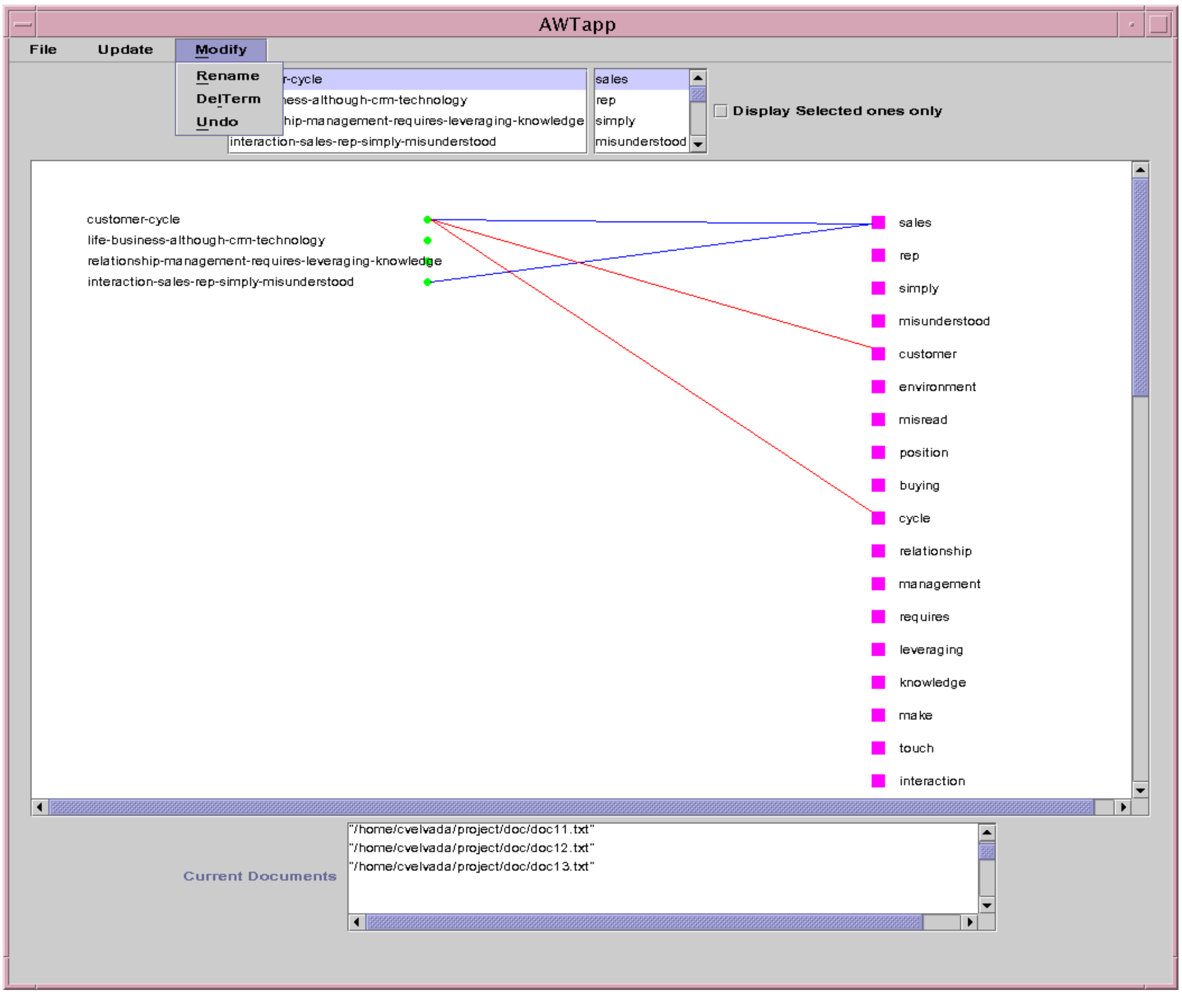

Figure 8 GUI, showing the modify menu

8. Future Directions

Making several improvements to our system would make it versatile. We are currently working on facilitating document based updates, as described by Berry et al.[1], which are Fold-In and SVD update methods and also we are trying to concentrate on developing the relations between the concepts.

10. Conclusions

Here, until now we have presented a system of Ontology Extraction from text documents using singular value decomposition, which constructs a domain ontology graph from a collection of text-based documents. The method used in this construction is statistical. It uses simple well-known matrix decomposition, and produces results whose validity is supported theoretically. Our system also allows the easy manipulation of the ontology. A system such as ours, which allows rapid construction of domain ontologies for use in systems requiring domain expertise, is a significant contribution.

11. Acknowledgements

This is a Bowie State University project in accomplice with University of Maryland, Baltimore County, and the National Security Agency, who are the sponsors for this project, to whom we are grateful. We sincerely thank our mentors Dr.Sadanand Srivastava, Dr.James Gil de Lamadrid for their valuable support.

REFERENCES

1. Berry, M. W., Dumais, S. T., O Brein, G. W. (December, 1995). Using linear algebra intelligent information retrieval.
2. Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., Harshman, R. (1990). Indexing by latent semantic analysis.
3. Greengrass, E. (February 1997). Information retrieval: an overview.
4. Golub, G. H., Reinsch, C. (1969). Singular value decomposition and least squares solutions. Hand book for Automatic Computation, Springer-Varlag, New York.
5. Nicholos, C., Dahlberg, R. (March 1998). Spotting topics with singular value decomposition, principles of Digital Document Processing, FIARO, St. Malo.
6. Berry,M.W., Do,T., O'Brien, Krishna,V., Varadhan,S., SVDPACKC: Version 1.0 User's Guide, Tech.Report CS-93-194, University of Tennessee, Knoxville, TN, October 1993.
7. Golub,G., Reinsch C., Singular Value Decomposition and Least Squares Solutions, in Handbook for Automatic Computation II, Linear Algebra., Springer-Verlag, New York, 1971.
8. Golub,G., Kahan.,W., Calculating the Singular Values and Pseudoinverse of the Matrix, SIAM Journal of Numerical Analysis, 2(3), pp.205-224, 1965;
9. Dr.Sadanand Srivastava, Dr.James Gil de Lamadrid, Yuriy Karakshyan, Document Ontology Extractor, CADIP 00
10. Dr.Sadanand Srivastava, Dr.James Gil de Lamadrid, Extracting an ontology from a document using singular value decomposition.
11. Golub,G., Luk,F., Overton,M., A Block Lanczos Method for Computingthe Singular Values and Corresponding Singular Vectors of a Matrix, ACM Transactions on Mathematical Software, 7(2), pp.149-169, 1981.
12. Chakravarthi S Velvadapu, Document Ontology Extractor, Applied research in Computer science, Fall-2001.

Appendix:

LOS ANGELES (AP) - For three days, President Clinton has stressed the need to invest in the poorest areas of the country. But here, on economically deprived turf that is still trying to recover from deadly race riots, he is arguing for investing in the poorest people themselves.

The president was visiting the community of Watts, scarred by riots nearly 30 years apart, to make the case today for private firms to help disadvantaged young people gain the skills they need for high-tech jobs in the new millennium.

Among those lined up to accompany Clinton was Magic Johnson, the former Los Angeles Lakers star who has revitalized South Central L.A. and other inner cities with multiplex movie theaters. Clinton was touring the Transportation Career Academy Program within Alain Leroy Locke High School, named for the first black Rhodes scholar. The facility helps prepare students for careers in transportation-related fields from urban planning to architecture. Since 1994, 1,800 students have participated in the academy, and 90 percent who graduate go on to college.

Later, Clinton was going to nearby Anaheim for the annual conference of the National Academy Foundation - where chief executives were huddling to discuss ways to connect employers with disadvantaged youth, especially those ages 16 to 24. The White House estimated that 10 million people in that age group are out of school, and 4 million of them lack a high school diploma.

"They have to pay attention to and care about the development of the work force," said deputy White House chief of staff Maria Echaveste. "They can't be competitive, they can't stay profitable, if they don't have a work force that is skilled and that is trained." At the conference, Clinton was to announce an 8 million initiative to help create "information academies" within inner city and rural schools. The initiative is a partnership between the Department of Labor and companies such as AT&T, Lucent Technologies and Cisco Systems.

That announcement closes out Clinton's tour, but he will remain in Los Angeles through Saturday to watch the U.S. women's soccer team compete for the World Cup.

Today's visit to Los Angeles' south side takes Clinton back to an area he first canvassed as a presidential candidate in May 1992, just days after riots in the wake of police acquittals in the beating of motorist Rodney King left 55 people dead and 720 buildings destroyed or damaged by fire.

Part of the complaint then - as it was in 1965, when 34 people died in Watts rioting- was the need for better access to jobs and social investment to eliminate the economic isolation of the inner city. The president will see a changed South L.A. After the 1992 riots, banks and federal redevelopment programs have made millions of dollars in loans and grants to local businesses, and many damaged areas have been rebuilt. And while residents welcome the government's help, there is a thriving spirit of free-enterprise and self-reliance.

The Baldwin Hills Crenshaw Plaza mall, boosted by a Magic Johnson movie theater and its proximity to affluent black neighborhoods, is booming with an occupancy rate of more than 90 percent. In other areas, retailers have opened new stores. Three banks, Washington Mutual, Wells
Fargo and Hawthorne Savings, partnered with Operation Hope to open banking centers where residents can apply for loans and take classes on managing their finances.

The president flew to Los Angeles from Phoenix, where he toured the facilities of La Canasta, a successful food producer, to highlight the needs of the Latino community on that city's south side. He strolled through the plant with owner Carmen Abril Lopez and watched as thousands of tortillas coursed past him on conveyor belts. While workers in white shirts and baseball caps removed the flawed ones, Clinton took a tortilla in his hands and inspected it, marveling at the fact that the plant produces and sells 840,000 tortillas each day.

"Our country has been really blessed by these good economic times," Clinton said. "But we know, as blessed as America has been, not every American has been blessed by this recovery. All you have to do is drive down the streets of South Phoenix to see that."