Semantic Metadata Extraction Using XemanticA Analysis

Sajith Vimukthi Weerakoon, Rohan Wickramasinghe, Erangi De Costa, Manisha Eleperuma, Shehan Perera and Sanath Jayasena
Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka.

***Abstract* - Identifying Metadata and understanding the semantics of them is vitally important and advantageous in analyzing, managing and utilizing information. This paper suggests a new technique which we call the XemanticA Analysis that is derived based on Latent Semantic Analysis (LSA). With this new technique the phrases, sentences or documents are evaluated with respect to their context and the system needs an initial training with respect to some known data. The technique which is proposed here has also been implemented and in this paper we introduce some of the important facts on the accuracy results and the outcomes of this technique and its implementation. This is a technique which can be used immensely in applications and there can be many extensions and enhancements that can be performed over the outcomes of this method.**

Keywords: Semantics, XemanticA Analysis, Metadata Extraction, Content Matrix (CM), Singular Value Decomposition (SVD), Latent Semantic Analysis (LSA)

I. Introduction

There are millions of documents that are being circulated on the Internet and other information repositories. These documents carry different kinds of information. It is indeed a very tiresome activity to identify the exact information that these documents carry, by manual means.

Extraction of information from documents and information repositories is vitally useful and at the same time this is identified to be a rather tedious task. Accurate information extraction leads to the better means of analyzing, managing, and using the information with ease and with a meaningful manner. Especially if we are aware of the semantics of given information, it will be invaluable [1]. Indeed, assessing the semantics of information will lead to an entirely new paradigm of computing.

Thus the demand and the research which is being carried out to create accurate techniques for information extraction and identifying the semantics of information is growing day by day. This paper introduces the XemanticA Analysis which is capable of assessing the semantics of a given word with respect to its context.

The Latent Semantic Analysis [2] is the precursor of XemanticA Analysis which has the core functionality of Singular Value Decomposition [3].

The XemanticA Analysis involves sophisticated Data Mining Techniques. There is a very huge computation taking place inside the implementation of this particular technique and it is identified and measured that the performance can be improved drastically by adding some minor changes to the implementation such as introducing parallel computing.

The paper demonstrates a mathematical proof into the XemanticA Analysis which is again derived using the mathematical algorithm of LSA and it also includes some results generated with respect to this technique using its implementation.

II. Background

Semantics is a branch of linguistics studying the meaning of words, the relationship between words and their meanings, and the individual meanings of words, as opposed to the overall meaning of a passage [4]. Semantics has the total power of revolutionizing the entire world of Information Technology.

The Semantic Web promises the wonders that can drive the mankind for a better tomorrow. This is an evolving development of the World Wide Web in which the semantics of information and services on the web is defined, making it possible for the web to "understand" and satisfy the requests of people and machines to use the web content [5].

One of the main barriers that is associated with moving towards the Semantic Web is the inability to identify and define the metadata in documents. As highlighted earlier, this process is of high importance as far as the context of information is concerned. There are certain attempts such as the DBpedia [6] which has attempted to identify the semantics of documents manually by defining the metadata by pure human interventions. But as the World Wide Web is an ever growing entity, it is indeed infeasible to create the metadata of all the web pages manually. The need for metadata extraction along with their semantics with some decent accuracy is a must. The XemanticA Analysis is a proposed precursor to achieve this target.

Latent Semantic Analysis (LSA) is a technique in natural language processing, in particular in vectorial semantics, of analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms [7].

The LSA uses the Singular Value Decomposition (SVD) as an internal technique. The XemanticA Analysis also uses SVD. Instead of the Occurrence Matrix which is used in LSA, XemanticA uses a new parameter called the Context Matrix (CM).

A. Latent Semantic Analysis

LSA uses a term-document matrix which is known as Occurrence Matrix, and this describes the occurrences of terms in documents. Columns represent documents and rows represent unique words and corresponding cell represents the occurrence of the word in the given document. When we consider the above matrix, we can consider as rows representing word vectors and columns representing document vectors. Then we use Singular Value Decomposition to decompose the matrix. SVD result in a word vector matrix (U), a document matrix (V) and a scalar matrix(S).

$$[U,S,V] = SVD\ (OM) \qquad (1)$$

By considering the n largest Eigen values in S we can get n dimension vectors omitting small Eigen values. Using vector operations or by plotting in a 2D graph (when n= 2) similar vectors belonging to similar category for an example words and documents, words and words, documents and documents can be assessed for their proximity. By using this algorithm we can extract related words, documents or word – document relations.

III. XemanticA Analysis

The XemanticA Analysis is the particular technique that we introduce through this paper. This technique is a degeneration of the Latent Semantic Analysis. There are two main parts in this, namely, the creation of the Context Matrix (Analogous to the Occurrence Matrix in LSA) and performing the Singular Valued Decomposition (SVD) over it.

In this paper we mainly focus on the creation of the Context Matrix as the SVD is considered to be a standard process.

IV. Context Matrix

Context Matrix is an indicator on the relevance of a particular sentence or a phrase to a given type of an entity. This is done by evaluating the patterns present in the data collection. The creation of the CM consumes a lot of computing power and it is considered to be one of the most sophisticated processes in XemanticA Analysis. There are several steps involved in the creation of the CM and these can be epitomized as given below.

1. Clustering the words which are associated with a particular word whose entity is known in the context of a document which is used to train the data. This is done using the Feeder.
2. The assessment of words which are associated with a given word in the particular text whose metadata need to be assessed using XemanticA Analysis.
3. Calculate the occurrences of the words in the associated words of each word and the inputs obtained through the Feeder and draft the Context Matrix.

Following parts will elaborate on each of these steps in length.

V. Feeder

In order to start the process of creating the CM, there should be a rich collection of words which are the most optimally associated with given entities. This task is done by the Feeder Algorithm (FA).

In order to run this algorithm there should be a collection of some text in which a word with its type of entity is known. Later in this algorithm the word whose entity type is known will be called as the known word and its entity will be called the type. Described below is the algorithm by words.

This algorithm also uses SVD as one of its internal operations. First of all it divides the whole document which is passed, into smaller chunks and then creates the Occurrence Matrix with respect to these chunks and unique words in the document. It then performs SVD over the OM and then gets the U, S, and V decomposed matrices. Here,

$$[U,S,V] = SVD\ (OM) \qquad (2)$$

With these we plot the graph with respect to three highest Eigen Values in the diagonal matrix S, take the respective columns in U and plot a graph by taking these columns as x, y and z axis and then analyze the relative positions of words. Thus we obtain a graph which is three dimensional.

We consider about the vectors of these relative points in the graph and then extract the set of words which are in the close proximity of the known word of the known type. This is done by drawing a sphere taking this point as the center and the radius of the sphere is decided with respect to the differences in the distances. This is assessed by a special parameter which is named as the Feeder Beta Factor.

The words which are inside this sphere drawn, will be extracted and stored in the file system or a database as the words related with the given type of entity.

This process is run with respect to several documents and at each time the stored data in previous processes will also be taken into account.

This raw set of words accumulated will be taken as immature at the very beginning and as it will only be used for the entity deciding once it passes the maturity test.

Maturity is tested by again drawing the words which are collected with respect to a given entity in a plot and checking whether more than 90% of words collected are within the Maturity Beta Factor. This is also a parameter which is used to draw a sphere taking the root of the graph as the centre of the sphere.

All the Beta Factors can be changed dynamically according to the accuracy that we want. These collected words are tabulated in the file system and then named after the type of entity. These words are assumed to be the set of words which are closely associated with the given entity. This closeness can be varied by adjusting the Beta Factor.

A. Genetic Algorithm to extract the words in the proximity

As an alternative method for the use of Beta Factors will be using a Genetic Algorithm to find the words which are suitable enough to be fallen in the proximity of a given entity type.

Here we consider all the relative distances of words as the universe set of solutions in the environment. We assess the relative distances of these words and assess the set of words which can be gone to the next level. The decision we make in figuring out the right fitness function decides the accuracy of the results.

During the implementation we have implemented a few fitness functions which are using different methodologies with respect to the relative distances of words.

After performing the process with any of the fitness functions that are suitable, we continue the process for a given number of generations, and through that, we can get the most optimal set of words which are in the close proximity of the known word. As it was detailed earlier we tabulate these words in the file system and use them at the time of Context Matrix generation as the feeds. These data which is stored in the database are called trained data.

The following topics will elaborate on the means in which this trained data can be used in metadata extraction in a real scenario.

VI. Identifying the Words Which Are Associated to a Given Word in the Text Whose Metadata Needs to Be Extracted

In the metadata extraction procedure of given text the first and the foremost step is to identify the words which are associated to each word in the text. This is again done using a similar technique described under the feeder.

First of all we degenerate text into smaller chunks and then create an Occurrence Matrix with respect to these chunks and unique words present.

After that we perform SVD over this OM and assess U, S, and V decomposed matrices. Again we plot the relative positions of each of the unique words given in the text as it was in the case of feeder.

Then we take each word as the pivot and then evaluate the words which are in the close proximity. This is also done in a similar manner described in the Feeder and a special Beta Factor which is called as the Extractor Beta Factor is used for the assessment.

The words which are associated with each word is then piled up together and used in order to create the Context Matrix.

VII. Creating Weights in the CM

Context Matrix is a matrix which is created by assessing the associated words for each word in a text with respect to the data trained and stored using the Feeder.

The following example will epitomize the whole process of creating the CM.

Let us assume that the following are the list of words which are taken as the outcome of the feeder.

```
entity{1} = 'UML Modeling Language
tool';

entity{2} = 'C++ object class
compilers  programming pointers
language';

entity{3} = 'orthogonal frequency
multiplex OFDM FDM PSK';
```

Let us assume that the text which is passed is as given below.

```
"C++ is a programming language
which can be used for the object
oriented programming. There are
many compilers which have been
written using C++ and in spite of
the number of languages which are
entering into the arena rapidly
C++ still has its popularity."
```

The list of words which are associated with the word C++ can be given as below.

```
C++ - C++,Programming,
      language, object,
      compilers.
```

	entity1	*entity2*	*entity3*
C++	*1*	*5*	*0*

This process is repeated for each word which is in the text and the outcome is known as the CM.

VIII. Analyzing the CM

As it was done earlier the CM is treated with SVD. Here we plot the graph between words and entities using their relative positions. (Entities are the piles of data created using the Feeder)

The Alpha Value is a user defined value which is customizable and this is used to draw a sphere centering each word in the graph. Then the closest entity which falls within the Alpha Value is proposed to be the entity type for the given word.

There are instances where there are no entities falling within the Alpha Factor. At such instances we assume that these words are belonging to some other entity type which is not within the trained data piles.

These words are then used to create other new types and this process is as described under the next topic.

IX. Enabling the Self Learning on Entity Types

This is one of the most important processes in the XemanticA analysis. If any of the words are not falling within the Alpha Factor as explained above, they are assumed to be belonging to a new type of words.

These words are piled up along with their associated words. These associated words are then stored as new entities. This storage is also a bit tricky.

We first assess whether each of these associated words for each word has more than 80% of overlapping with the associated words of another word. If this overlapping is present with the given percentage we pile up these associated words together and store them under a new entity.

These initial entities are considered to be immature and we assess these with the other immature words sets which are collected with the help of the Self Learning System.

Once these words sets become fairly decent, they are tested for their maturity. These new entity types can be given their names with the user intervention and later we can even automate this process.

X. Implementation

The XemanticA Analysis is implemented using Java based technologies. There were few prototypes which were implemented using MatLab and the final system is implemented using Java. There are few open source libraries which were used to perform some of the mathematical functions.

The system is developed in object oriented concept and thus we can change any implementation without adding drastic changes to other components.

The implemented version of the XemanticA Analysis is developed in such a way that it can be used to implement other applications which are directly dealing with information manipulation and analysing.

XI. Evaluation

The correctness of the XemanticA Analysis can be proven mathematically. The general proof for Latent Semantic Analysis [2] has been used in proving this. The implementation of the XemanticA Analysis was used in order to test some given input parameters. The obtained results are also given under this section.

A. *Proof of the XemanticA Analysis*

Let X be a matrix where element (i,j) describes the occurrence of term i in document j (this can be, for example, the frequency). X will look like this:

$$\begin{array}{cc} & d_j \\ & \downarrow \\ t_i^T \rightarrow & \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \end{array} \qquad (3)$$

Now a row in this matrix will be a vector corresponding to a term, giving its relation to each document:

$$t_i^T = [x_{i,1} \;\cdots\cdots\; x_{i,n}] \qquad (4)$$

Likewise, a column in this matrix will be a vector corresponding to a document, giving its relation to each term:

$$d_j = \begin{bmatrix} x_{1,j} \\ \vdots \\ x_{m,j} \end{bmatrix} \qquad (5)$$

Now the dot product $t_i^T t_p$ between two term vectors gives the correlation between the terms over the documents. The matrix product XX^T contains all these dot products. Element (i,p) (which is equal to element (p,i)) contains the dot product $t_i^T t_p = t_p^T t_i$. Likewise, the matrix $X^T X$ contains the dot products between all the document vectors, giving their correlation over the terms: $d_j^T d_q = d_q^T t_j$.

Now assume that there exists a decomposition of X such that U and V are orthogonal matrices and Σ is a diagonal matrix. This is called Singular Value Decomposition (SVD):

$$X = U\Sigma VT \qquad (6)$$

The matrix products giving us the term and document correlations then become

$$XX^T = (U\Sigma V^T)(U\Sigma V^T)^1 = (U\Sigma V^T)\left(V^{T^1}\Sigma^T U^T\right) = U\Sigma V^T V\Sigma^T U^T = U\Sigma\Sigma^T U^T \qquad (7)$$

$$X^T X = (U\Sigma V^T)^1 (U\Sigma V^T) = \left(V^{T^1}\Sigma^T U^T\right)(U\Sigma V^T) = V^T V\Sigma^T U^T U\Sigma V^T = V\Sigma^T\Sigma V^T \qquad (8)$$

Since $\Sigma\Sigma^T$ and $\Sigma^T\,\Sigma$ are diagonal we see that U must contain the eigenvectors of XX^T, while V must be the eigenvectors of $X^T X$. Both products have the same non-zero Eigen values, given by the non-zero entries of $\Sigma\Sigma^T$, or equally, by the non-zero entries of $\Sigma^T\,\Sigma$. Now the decomposition looks like this:

$$\begin{array}{ccccc} & X & U & \Sigma & V^T \\ & (d_j) & & (\hat{d}_j) & \\ (t_i^T) \rightarrow & \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix} & = (t_i^T) \rightarrow [[u_1] \ldots \; [u_l]] & \cdot \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_l \end{bmatrix} & \cdot \begin{bmatrix} \begin{bmatrix} v_1 \\ \vdots \end{bmatrix} \\ [v_l] \end{bmatrix} \\ & \downarrow & & \downarrow & \end{array} \qquad (9)$$

The values σ_1, σ_2, ... σ_l are called the singular values, and u_1 .... u_l and v_1 ....v_l the left and right singular vectors. Notice how the only part of U that contributes to t_i is the ith row. Let this row vector be called $\hat{t}_i$. Likewise, the only part of V^T that contributes to d_j is the jth column, $\hat{d}_j$. These are not the eigenvectors, but depend on all the eigenvectors.

It turns out that when you select the k largest singular values, and their corresponding singular vectors from U and V, you get the rank k approximation to X with the smallest error (Frobenius norm). This approximation has a minimal error. But most importantly we can now treat the term and document vectors as a "concept space". The vector $\hat{t}_i$ then has k entries, each giving the occurrence of term i in one of the k concepts. Likewise, the vector $\hat{d}_j$ gives the relation between document j and each concept. We write this approximation as

$$X_k = U_k \Sigma_k V_k^T \quad (10)$$

You can now do the following:

• See how related documents j and q are in the concept space by comparing the vectors $\hat{d}_j$ and $\hat{d}_q$ (typically by cosine similarity). This gives you a clustering of the documents.

• Comparing terms i and p by comparing the vectors

$\hat{t}_i$ and $\hat{t}_p$, giving you a clustering of the terms in the concept space.

• Given a query, view this as a mini document. and compare it to your documents in the concept space.

To do the latter, you must first translate your query into the concept space. It is then intuitive that you must use the same transformation that you use on your documents:

$$d_j = U_k \Sigma_k \hat{d}_j \quad (11)$$

$$\hat{d}_j = \Sigma_k^{-1} U_k^T d_j \quad (12)$$

This means that if you have a query vector q. you must do the translation $\hat{q} = \Sigma_k^{-1} U_k^T q$ before you compare it with the document vectors in the concept space. You can do the same for pseudo term vectors:

$$t_i^T = \hat{t}_i^T \Sigma_k V_k^T \quad (13)$$

$$\hat{t}_i^T = t_i^T V_k^{-T} \Sigma_k^{-1} = t_i^T V_k \Sigma_k^{-1} \quad (14)$$

$$\hat{t}_i = \Sigma_k^{-1} V_k^T t_i \quad (15)$$

XII. Further Enhancements of XemanticA Analysis

The technique which is elaborated here can be subjected to drastic enhancements. For an instance this same theory can be used to define ontology of a certain context. This is one such important issue that this can be solved using this technique.

The following is a brief description on how this scenario can be solved using XemanticA Analysis. Since we calculate the relative positions of each of the entity and word we can assess the relationships of words with respect to entities. Thus we can create hypothesis on the relationships between words in the context in terms of their semantics. This is then can be used to create ontology with respect to the data that is present.

With all these features the XemanticA Analysis can even attempt to build up the Semantic Web which is still a dream for many people.

XIII. Applications Developed Using XemanticA Analysis

XemanticA Analysis provides a solution to one of the dying problems that is prevailing in the world of semantics. This theory can then be used in a wide range of applications which are promising enormous innovation and effectiveness. These applications can vary from fast, accurate and precise search tools to sophisticated knowledge management tools.

The first application that is been developed using the implementation of XemanticA Analysis is known and the XemanticA Knowledge Desk Pro which is promising enormous accuracy and effectiveness.

Detailed under the next topic is the XemanticA Knowledge Desk Pro.

XIV. XemanticA Knowledge Desk Pro

XemanticA Knowledge Desk Pro is a tool which is used to semantically index files in a desktop and manage them effectively. This application provides functionality to perform searches over the indexed files and their semantic content.

This application also provides Graphical Interface which makes the task of the user easier and even pleasing. Figures below depict some instances of the interface of this application.

The figure below displays a magnified view of the entities extracted from a given document using the Knowledge Desk Pro.

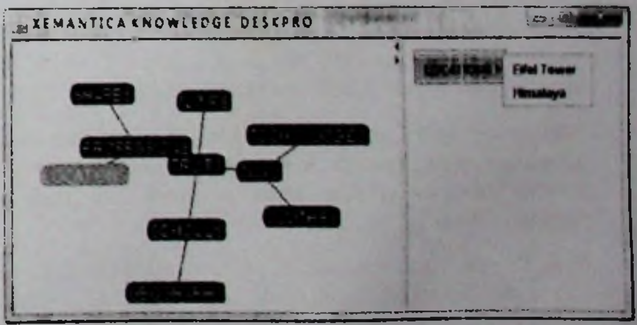

Figure 1 Extracted entities are displayed in a dynamic node graph

XV. Results and Accuracy

The implementation of XemanticA Analysis is tested with many sample inputs and records and it recorded an accuracy level which is above 90%. There are few enhancements that can be done over the implementation and it is expected that an accuracy level which is much closer to 95% can be obtained.

The accuracy of XemanticA Analysis can be determined by the results generated by XemanticA Knowledge Desk Pro.

XVI. Limitations

The main limitation is the higher computational power that this needs. There are few remedies that can be proposed in order to enhance the speed of the overall system and given below are some of them.

Introduction of Parallel Computing can be considered to be one of the main features that can be used to enhance the performance of the system. There are many instances where the parallel computing can be applied and it will help the system attain much higher performance.

Using of parallel computing with the essence of distributed computing will be another good remedy where higher performance can be attained. The computations can be distributed over many computers and be handled at a single center so that the system performs better and faster.

This technique is not very sound at applications which will need the real time processing. Thus some indexing technique as we have used in the XemanticA Knowledge Desk Pro should be used.

If the searches and other important functionalities can be performed over the index the system will act faster. Then again the storage will be one of the key issues. But yet considering the fact that only text is used we can presume that the space requirement will be base minimum.

XVII. Conclusion

The XemanticA Analysis proposed here is of very high importance as far as the information management is concerned. This technique is considered as a self learning system which can improve its scope with time.

There are many applications which can be developed using XemanticA Analysis and this is a technique which has already been implemented, tested and used in applications.

With many new enhancements that are promising to be added on top of XemanticA Analysis we can even think of moving towards critical areas in computing such as Semantic Web.

With the enormous capabilities of XemanticA Analysis we can make the information management task no longer a burden.

References

[1] Richmond H. Thomason, "What is semantics?" [Online Article].Available:http://www.eecs.umich.edu/~rthomaso/documents/general/what-is-semantics.html ,[Accessed: 28/12/09]

[2] Thomas Landauer, P. W. Foltz, & D. Laham . "Introduction to Latent Semantic Analysis", [Online Article]. Available: http://lsa.colorado.edu/papers/dp1.LSAintro.pdf [Accessed: 28/12/09]

[3] Wall, Michael E., Andreas Rechtsteiner, Luis M. Rocha (2003). "Singular value decomposition and principal component analysis". in A Practical Approach to Microarray Data Analysis. D.P

[4] Semantics http://en.wiktionary.org/wiki/semantics

[5] Semantic Web Nigel Shadbolt, Wendy Hall, Tim Berners-Lee (2006). "The Semantic Web Revisited". IEEE Intelligent Systems. http://eprints.ecs.soton.ac.uk/12614/1/Semantic_Web_Revisted.pdf. Retrieved April 13, 2007.

[6] "DBpedia",[Online Article] Available: http://dbpedia.org [Accessed: December 29, 2010]

[7] Melanie Martin (2002), *An Introduction to Latent Semantic Analysis*, [Online]. Available FTP: www.cs.nmsu.edu/~mmartin/LSA_Intro_AI_Seminar.ppt File: LSA_Intro_AI_Seminar.ppt [Accessed May 20,2010]

[8]"Latent Semantic Analysis", [Online] Available: http://en.wikipedia.org/wiki/Latent_semantic_analysis [Accessed: April 04, 2010]