Creating Usage Sensitive Hierarchical Knowledge Structures

Supervised by

Sean Colledge
Department of
Computer Science
University of Cape
Town
Rondebosch
Cape Town
South Africa, 7700
scolledg@cs.uct.ac.za

Dr. Anet Potgieter
Department of
Computer Science
University of Cape
Town
Rondebosch
Cape Town
South Africa, 7700
anet@cs.uct.ac.za

October 2007

Abstract

This research project consists of a system, which attempts to combine two methods of indexing documents and explores the idea that a correlation can be found between the data obtained from both methods. It then also attempts to discover any underlying usage sensitive knowledge structures that may exist in the data that was obtained. The project is divided into two sub-systems: The Automatic Document Indexing System and the Manual Tagging System. This report is only concerned with the Automatic Document Indexing System. Through the implementation of the Probabilistic Latent Semantic Analysis (PLSA) algorithm, popular in the field of Machine Learning, probabilities are obtained for the terms in a small number of short articles. The tag data obtained from the Manual Tagging System is then structured into term pairs and the differences in probabilities, from the PLSA data, between these term pairs is calculated with the hope that they share a correlation with the distances between the tags in the tag clouds of the Manual Tagging System. Due to time constraints however, the algorithm's accuracy could not be effectively evaluated, but the results that were obtained do show that there is very little association between the data of the two systems.

Table of Contents

List of Tables

List of Figures

1 Introduction

With more information being digitized and stored in digital libraries every day and communication networks reaching more users farther away, it is no surprise that a huge repository of textual data has become openly available to the public. The challenge that this creates is largely focused on how users can intelligently structure this huge mass of textual information and search through it to obtain only the relevant documents to a search query. This must of course also be done in a feasible amount of time.

Even though two documents on the web may be on precisely the same topic, they may use different terminology. Many techniques have been designed over time to try and overcome this problem of synonyms (e.g. car vs. automobile), spelling variations (e.g. Al Qaeda vs. Al Qaida), abbreviations (e.g. People's Republic of China vs. PRC), or valid ways of naming the same entity (e.g. Bombay vs. Mumbai). **[1]** Latent Semantic Analysis (LSA) has been a popular and successful technique in alleviating the challenges that the above has introduced and has been implemented by many researchers in this field **[1], [2], [3], [4]**.

In many cases a few small yet very significant changes have been made to the original algorithm, which have greatly increased the success of its outcome. This has given rise to Probabilistic Latent Semantic Analysis (PLSA).

An overall goal of this project is to achieve the semi-automatic construction of a usage sensitive hierarchical knowledge structure through the use of data obtained from Probabilistic Latent Semantic Analysis, combined with the data obtained from a *flat* tag cloud structure. The results obtained from PLSA on a domain of textual documents can be combined together with the underlying structure obtained from a user created tag cloud, and thus used to strengthen the associations between the terms of each document. These results can hopefully also show a correlation between the associations found by using the PLSA data as well as the data obtained from the Manual Tagging System using tag cloud structures.

As will be discussed throughout the rest of this research report, PLSA has been proven to be very successful in the use of automatic document indexing and will hopefully be a viable option in achieving the goals of this project.

2 Background and Related Work

2.1 Latent Semantic Analysis (LSA)

Latent Semantic Analysis is a technique in natural language processing whereby the relationships between a set of documents, and the terms they contain, are analysed, producing a set of concepts related to the documents and terms. It has many applications **[5]**, the most important being its application for Information Retrieval, which is the main focus of this paper.

LSA goes beyond lexical matching and addresses the well-known problem of using keywords alone to describe the entire contents of a document **[6].** It captures the underlying semantic structures, which better indexes the documents using the truncated singular value decomposition (SVD) of the term-document matrix. The collection of documents can be represented as a term-document matrix in the so-called *latent semantic space* in which the rows of the matrix correspond to terms, the columns to documents, and the cells to the number of times a given term occurs in a given document. Therefore, given n documents that contain m unique terms, we obtain an m x n matrix such that $a_{i,j}$ is the number of times that word i occurs in document j. In **[3]**, Papadimitrioum Raghavan, Tamaki & Vempala proposed a new theorem which performs the LSA pre-computation not on the original term-document matrix, but on a low-dimensional projection. This was shown to achieve great computational savings as well as no great loss of accuracy.

The success and effectiveness of LSA as a technique to improve Information Retrieval has been demonstrated empirically in a number of research articles and has shown a significant increase in retrieval precision **[1, 2, 7].**

A two stage variation of the original LSA algorithm was developed in **[1]**, and showed with successful results that it greatly improved the performance of the algorithm. They discovered that under a broad range of circumstances, a straightforward application of LSA fails to rank true aliases highly. Their new algorithm therefore would rather create a new set of pseudo-documents, which are documents created from each alias obtained from the original run of the LSA algorithm. LSA is then run on these new documents again.

Lastly, another popular use of the LSA technique, which has shown good results, is in the field of Semantic Image Retrieval **[5].** It allows for the comparison of a collection of data images, allowing a computer to determine which images are semantically alike. This technique does however use the branch of LSA called Probabilistic Latent Semantic Analysis, which will be discussed in the next section.

2.2 Probabilistic Latent Semantic Analysis (PLSA)

PLSA is a relatively new approach to automated document indexing, which uses a statistical latent class model for the factor analysis of the data. It is applied to a

training corpus of text documents through the use of the Expectation Maximisation algorithm **[8]**, and in doing so, is able to deal with domain-specific synonymy as well as with polysemous terms (terms with a diversity of meanings).

Although LSA has been applied with remarkable success in areas of research such as automatic document indexing, its lack of a statistical foundation has resulted in a number of deficits pertaining to the algorithm **[8].** Thomas Hofmann proposed a technique called Probabilistic LSA in **[8]**, which has a solid statistical foundation because it is based on the likelihood principle and thus provides better results than LSA for term matching in retrieval applications. It also defines a proper generative model of the data, which allows standard techniques from statistics to be applied to the model fitting, model combination and complexity control of the original LSA algorithm.

2.2.1 The PLSA Model

The conditional probability between documents *d* and words *w* is modelled through a latent variable *z*, which can be loosely thought of as a class or topic. The model is then parameterised by $P(w|z)$, the probability of a word *w*, and $P(z|d)$, the probability of the latent class *z*, and the words may belong to more than one class and a document may discuss more than one topic. The joint probability of a document *d* and a word *w* is represented as:

$$P(w, d) = P(d) \sum_{z} P(w \mid z) P(z \mid d) \qquad \textbf{(1)}$$

These parameters can be estimated using the Expectation Maximisation (EM) algorithm, which through its iterations can maximise the following log-likelihood function:

$$L = \sum_{d \in D} \sum_{w \in d} f(d, w) \log P(d, w) \qquad \textbf{(2)}$$

where $f(d, w)$ is the frequency of word *w* in document *d*. **[7, 8]** The EM algorithm starts with the Expectation-step, which estimates the probability that a word *w* in a particular document *d* is explained by the class corresponding to *z* through the following equation:

$$P(z|w, d) = \frac{P(w \mid z)\, P(z \mid d)}{\sum_{z`} P(w \mid z`)\, P(z` \mid d)} \qquad \textbf{(3)}$$

Next follows the Maximisation-step, where parameters $P(w|z)$ and $P(z|d)$ are re-estimated to maximise *L*:

$$P(w|z) = \frac{\sum_{d} f(d, w)\, P(z \mid w, d)}{\sum_{w`} \sum_{d} f(d, w`)\, P(z \mid w`, d)} \qquad \textbf{(4)}$$

$$P(z|d) = \frac{\sum_w f(d, w)\, P(z \mid w, d)}{\sum_{z`} \sum_w f(d, w)\, P(z` \mid w, d)} \qquad \textbf{(5)}$$

After this, a folding-in process is applied, which uses the EM algorithm in a similar way as before, the Expectation-step is identical and the Maximisation-step keeps all the $P(w|z)$ constant and only recalculates $P(z|q)$, where q is any new (test) documents that may be added. Only a small number of iterations of the EM algorithm is sufficient for folding-in **[7].**

2.2.2 Model Fitting with Tempered EM

Following the same steps as above for the PLSA algorithm, Hofmann had a slight change in his algorithm to the one mentioned above. In **[8]**, he introduces the EM algorithm and its iterations, defining it as a convergent procedure that approaches a local maximum of the log-likelihood in equation (2) as seen above. He states that it is naïve to assume that a model will generalise well on new data.

He therefore proposes a generalisation of maximum likelihood for mixture models – called *tempered* EM (TEM). A control parameter β is introduced, which modifies the E-step equation (3) as seen above according to the following:

$$P_\beta (z|d, w) = \frac{P(z)\,[P(d \mid z)P(w \mid z)]^\beta}{\sum_{z`} P(z`)\,[P(d \mid z`)P(w \mid z`)]^\beta} \qquad \textbf{(6)}$$

$\beta = 1$ results in the standard E-step, while for $\beta < 1$ the likelihood part in Bayes' formula (Formula 3) is discounted. The main advantage of the TEM algorithm in the context of document indexing is the avoidance of overfitting. In order to obtain the optimum value of β, the following scheme can be followed:

i. Set $\beta \leftarrow 1$ and perform EM until the performance on held-out data deteriorates (*early stopping*).
ii. Decrease β, e.g., by setting $\beta \leftarrow \eta\, \beta$ with some rate parameter $\eta < 1$.
iii. As long as the performance on held-out data improves, continue TEM iterations at this value of β.
iv. Stop on β, i.e., stop when decreasing β does not yield further improvements, otherwise go to step (ii).
v. Perform some final iterations using both, training and held-out data.

2.3 Examples

As a display of the results from a test run of the PLSA algorithm by T. Hofmann in **[8]** the following two tables would be obtained:

"plane"	"space shuttle"	"family"	"Hollywood"
plane	space	home	film
airport	shuttle	family	movie
crash	mission	like	music
flight	astronauts	love	new
safety	launch	kids	best
aircraft	station	mother	hollywood
air	crew	life	love
passenger	nasa	happy	actor
board	satellite	friends	entertainment
airline	earth	cnn	star

Table 1. Four factors from a 128 factor decomposition of the example corpus, listed with their 10 most probable words **[8]**.

"Bosnia"	"Iraq"	"Rwanda"	"Kobe"
un	iraq	refugees	building
bosnian	iraqi	aid	city
serbs	sanctions	rwanda	people
bosnia	kuwait	relief	rescue
serb	un	people	buildings
sarajevo	council	camps	workers
nato	gulf	zaire	kobe
peacekeepers	saddam	camp	victims
nations	baghdad	food	area
peace	hussein	rwandan	earthquake

Table 2. Four additional factors from the 128 factor decomposition of the example corpus **[8]**.

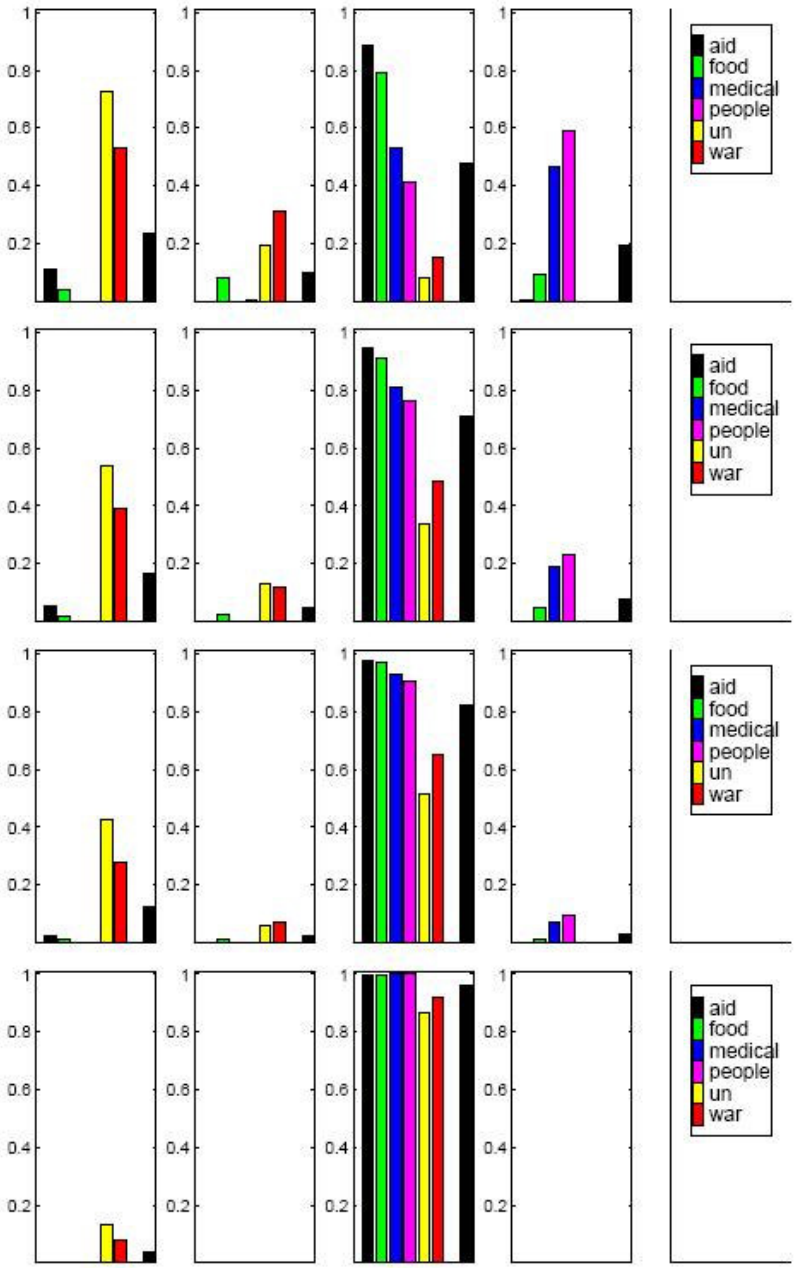


Figure 1. Folding in a query consisting of the terms “aid", “food", “medical", “people", “UN", and “war" **[8]**.

Figure 1 visualises the evolution of the posterior probabilities and the mixing proportions in the course of the EM algorithm’s iterations.

3 Specification of Solution

This project was proposed with an overall goal of achieving the semi-automatic construction of a usage sensitive hierarchical knowledge structure through the use of data obtained from Probabilistic Latent Semantic Analysis, combined with the data obtained from a *flat* tag cloud structure. The results obtained from PLSA on a domain of textual documents can be combined together with the underlying structure obtained from a user created tag cloud, and thus used to strengthen the associations between the terms of each document. It is also hoped that a correlation can be found between the automatic and manual process of determining the topics of a document. There are two main sections into which the project has been divided, namely the Manual Tagging System implemented by Ian Saunder and then the Automatic Document Indexing System implemented by the Author.

3.1 Manual Tagging System

This system involves user's manually entering data into a *tag cloud*. A tag cloud can be explained to be a two-dimensional space that contains terms relating to a specific article or set of text. These terms can be shifted around inside the tag cloud and its functionality is such that the more a term is tagged, the bigger the font size of the term becomes inside the tag cloud. They are commonly used on websites such as blogs where users would like to tag terms that explain the meaning of the blog as a whole. A typical example of a tag cloud is shown below.

Figure 2. A visualisation of a typical tag cloud.

For the purpose of this project, the users were asked to read an article obtained from the UnNews website (affiliated with uncyclopedia.org) and then to tag one term that exists in the article, to be a measure of the article as a whole. The users were required to move the tags that already existed in the tag cloud, such that terms that are linguistically related to each other were placed closer to each other and vice versa.

In doing so, data was obtained manually from users to determine what the users intuitively thought were the more common topics that describe each article.

It is with this in mind that a possible semi-automatic constructed ontology could exist in these tag clouds, where the distances between the terms in the tag cloud could be used as weightings for their association to each other in the ontology.

3.2 Automatic Document Indexing System

In this research project the popular automatic document indexing algorithm used in Machine Learning, called Probabilistic Latent Semantic Analysis, will be implemented. This algorithm was proposed by T. Hofmann in **[8]** and has become a popular means of determining the topic of a document without the use of manual intervention.

Once this algorithm has been implemented and the output that is produced by the PLSA algorithm is analysed and correctly structured, it will be compared together with the manual data obtained through the use of the tag clouds as well as the user entered data.

With this in mind, it was the goal of the research project to analyse and detect a possible correlation between the manual and automatic document indexing approaches and if a correlation exists, to determine how strong or weak the two approaches are relative to each other.

4 Design Specification

4.1 High Level Overview

The implementation consists of three distinct sections each written in its own programming language. The indexing component was implemented under the Java Platform. This follows a process of analysing a number of documents and producing a Term / Document Matrix through a number of techniques discussed in more detail later on.

The Probabilistic Latent Semantic Analysis (PLSA) component was implemented using Matlab. This was due to PLSA's mathematical and statistical complexity and the mathematical advantages that Matlab can provide for such a component.

And lastly, there is the linking component, which was also implemented under the Java Platform. This component is responsible for analysing the output obtained from the PLSA component as well as data obtained from the users when tagging the articles displayed on the website.

The figure in *Appendix A* shows a high level overview of the process mentioned above.

4.2 Indexing Component

The PLSA algorithm requires a number of operations to be performed on the raw text data of each article before the algorithm itself can be applied. These operations are performed by the indexing component. It would have been preferred to write the entire implementation in the same programming language, but the reason for it to be implemented in the Java platform is due to the fact that Matlab simply does not support the necessary operations that are needed to be executed by this component.

The component is required to determine the frequency of each word that exists in each article and then to set up a matrix with these frequencies in it. To automatically obtain only the article text from each web document and strip out all of the other unnecessary HTML tags, a HTML Parsing library is used. The Jericho HTML Parser **[21]** was chosen for this purpose. It is a lightweight HTML Parser that has all the functionality needed for the processing of the HTML documents that are used in this project.

Not all of the words in the articles are used, due to a process of removing all stop words (words that are more common and will have no bearing to the article as a whole, such as and, but and the) as well as through stemming. Stemming is the process of determining the base term for every term in the article, e.g. If the term found is "*cardiology*", then its base word would be "*cardio*".

Once these operations have been completed the data is saved in matrix format to a plain text file.

The following UML Class Diagram depicts the design and structure of this component.

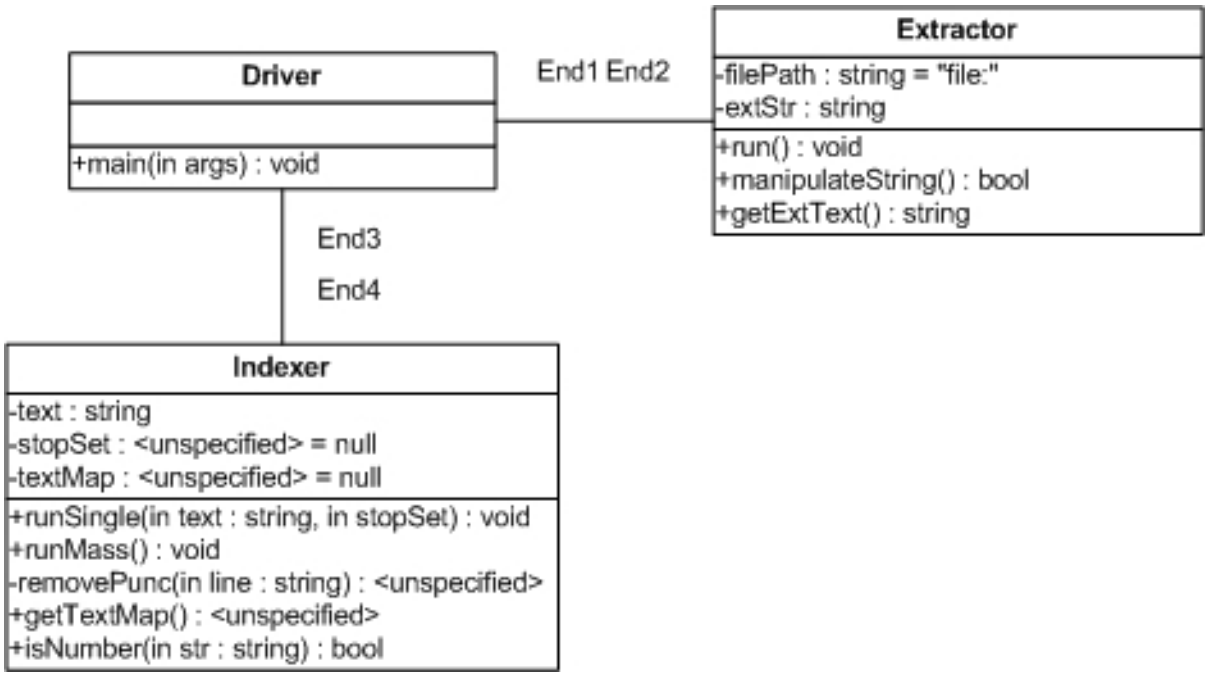


Figure 3. UML Class Diagram of the Indexing Component.

4.3 PLSA Component

This component is written as a Matlab script due to the fact that the algorithm can be implemented in a much smaller amount of code than in any other language such as Java. Matlab is also well known for its high speed in executing mathematical operations on matrices. This is very handy when the size of the matrix is directly related to the number of words in each article and the number of articles that are being analysed and it can become very large. This can increase the time needed to perform the algorithm exponentially.

4.3.1 High Level Overview of the PLSA Algorithm

The PLSA algorithm is an automatic document indexing algorithm which is used in the field of Information Retrieval to determine a number of topics that describe each article that was analysed, as a whole. There are a number of algorithms that exist to do exactly this, but what sets this algorithm aside from all the others is that it has a sound statistical foundation upon which it relies to produce as accurate results as possible.

It follows the Log-Likelihood Maximisation Estimation approach whereby it performs a number of iterations after which it calculates a new likelihood estimation at the end of each iteration. The aim being to continue with the iterations until the log-

likelihood function, shown below, reaches a local maximum. The algorithm then stops executing.

$$\mathcal{L} = \sum_{d \in \mathcal{D}} \sum_{w \in \mathcal{W}} n(d, w) \log P(d, w),$$

Figure 4. Log-Likelihood Function to be maximised.

Where: n(d, w) = the observed pair of document and term
P(d, w) = P(d) . P(w|d)
P(d) = the combined probability of the document column in the matrix
P(w|d) = the sum of each word probability and its dot product with its latent class probability

During each iteration an EM algorithm is applied to the already existing matrix that was calculated in the previous step. This EM algorithm consists of two steps, namely the E-step and the M-step. The E-step computes posterior probabilities based on estimates of the parameters used in the main PLSA algorithm and then the M-step where the parameters are updated for the given posterior probabilities that were computed in the previous E-step.

Once the local maximum for the Log-Likelihood function is reached, the current matrices are saved to a plain text file.

The figure below outlines a high level overview of the above mentioned process.

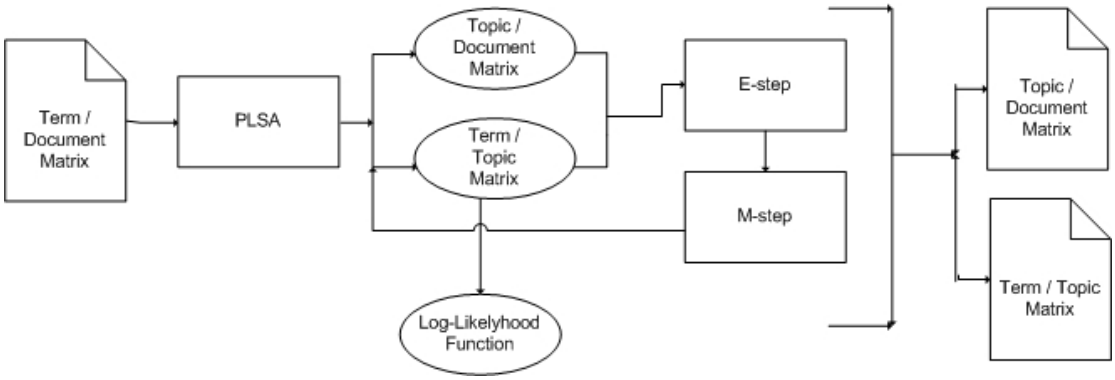


Figure 5. PLSA Component

4.4 Linking Component

This is the final component in the project implementation. It is responsible for reading in all the data that is produced by the PLSA algorithm and organising it into data structures in Java so that it can be easily used for analysing purposes.

It is also responsible for obtaining the data from the database that was produced by the users during the tagging process related to the other half of this project, which is implemented by Ian Saunders. This data is also organised into data structures after

which the data is combined and calculations are performed on the data so that graphs can be produced in Matlab using the output.

A UML Class Diagram that depicts the Linking Component can be found in *Appendix B*.

The figure below outlines a high level overview of the above mentioned process.

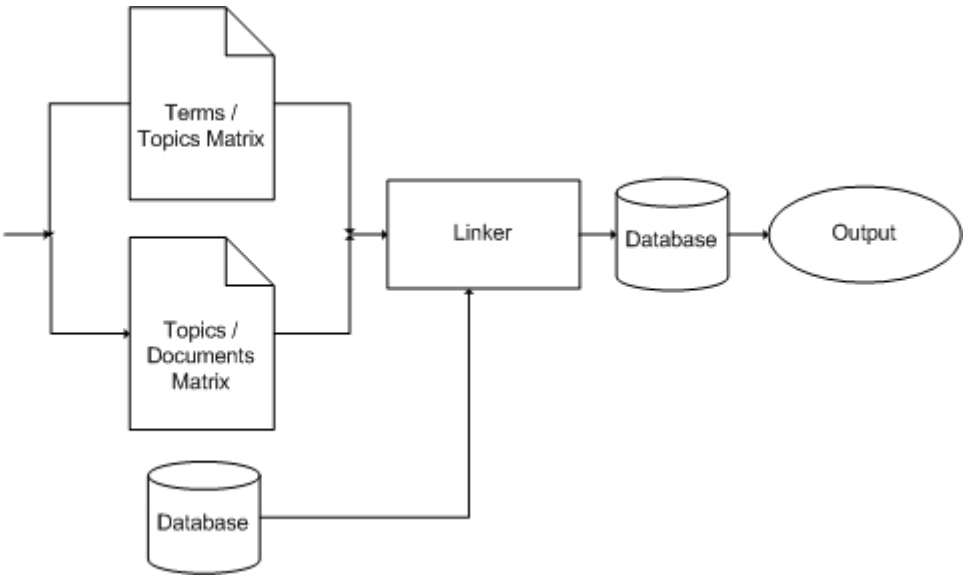


Figure 6. Linking Component

The Author and Ian Saunder agreed upon a MySQL database design that would be best suited for both parties. Prior knowledge of the database structure was needed so that the correct SQL queries could be implemented to obtain the data that was needed by this component.

A view of the MySQL database design can be seen below.

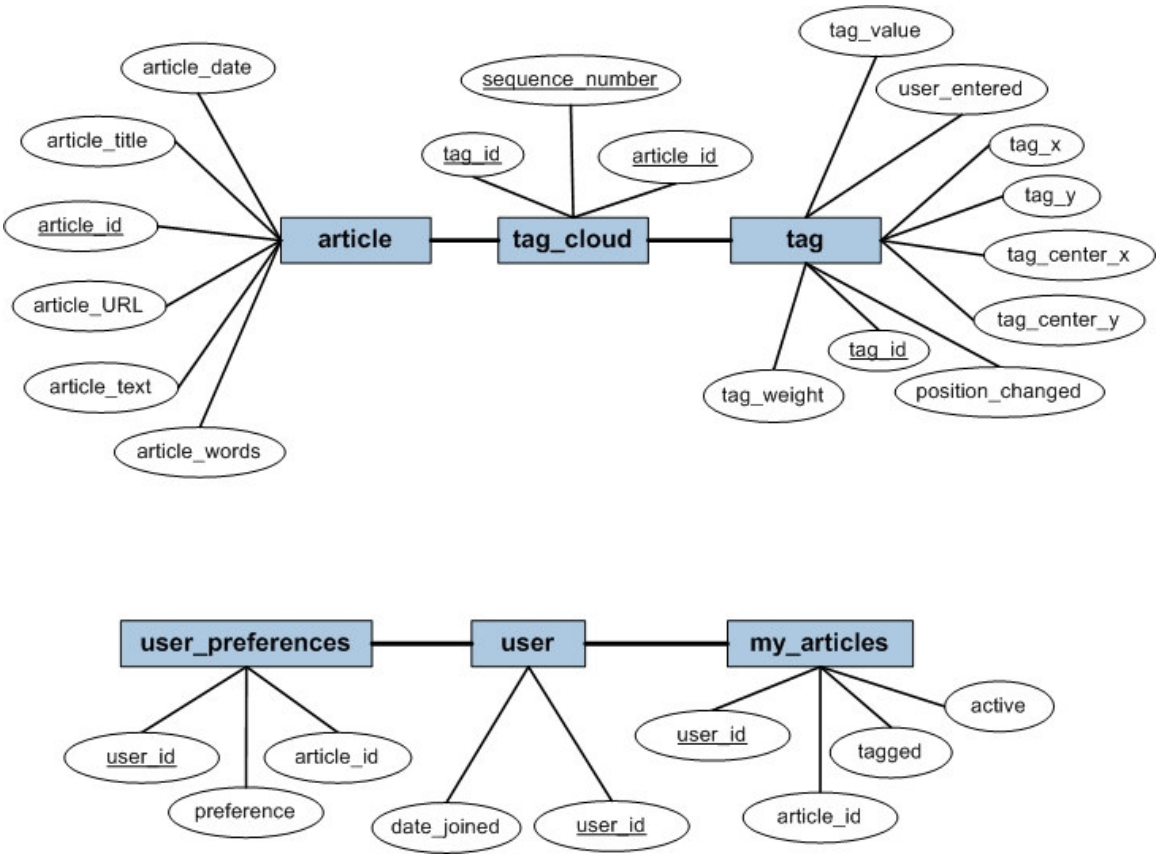


Figure 7. MySQL Database Design View.

5 Implementation

5.1 Extracting Tool

This is a small console application that was written in Java to be used for the purpose of purely extracting the text of a web article from inside all the other unnecessary tags of the web document. These include items such as HTML links, images, advertising blocks and menu items. It consists of one class, namely the Extractor class, which is called from the Driver program. The Jericho HTML Parser **[21]** is called from the Extractor class and then using the default text extracting methods together with a few chosen tag exclusions, a chunk of text is returned by the library.

The chosen tag exclusions are included by overriding a method from the Jericho HTML Parser as shown below.

```
public boolean excludeElement(StartTag startTag)
{
    //ignore all text inside tags with these class names
    //this type of text is not part of the article itself
    return "metadata".equalsIgnoreCase(startTag.getAttributeValue("class")) ||
           "thumbcaption".equalsIgnoreCase(startTag.getAttributeValue("class")) ||
           "thumb tright".equalsIgnoreCase(startTag.getAttributeValue("class")) ||
            "thumbinner".equalsIgnoreCase(startTag.getAttributeValue("class")) ||
            "magnify".equalsIgnoreCase(startTag.getAttributeValue("class"));
}
```

Figure 8. Code snippet showing the chosen exclusion tags

The driver program then calls the class method that takes the returned chunk of text and splits it into its relevant pieces. Now the articles that were used in the project have a few distinguishing features, which can be used to determine exactly which text pieces are in the article and which are not. Firstly, the date of the article appears at the top of each article just before the article text begins. So the class determines where this date is in the text chunk and removes all text prior to this point. It then moves onto a number of other items which randomly appear at the end of each article, they are listed in a certain order, as it is not always known which item will appear in which document, so a number of if statements were designed to catch either one of them and then remove all the text that appears after that point. The program is now left with a block of text that only contains the text from the article itself. This block of text is then written to a file in the same path as where the program was executed.

5.2 Indexing Component

The Indexing component consists of two main classes, namely the Extractor Class and the Indexer Class. The Extractor Class is the exact same class that is used by the Extracting tool mentioned above.

A graphical user interface (GUI) was implemented for this component, which asks the user to select three items. Firstly, the path of the Input Folder, which contains the web documents that need to be processed. All HTM and HTML files that exist in this folder are processed by the Indexer. Secondly, the path of the Output Folder, where the user would like the output of this component to be saved to. And lastly, the path where the Stop Word text file can be found.

All the parameters are passed back to the Driver program which will pass them on to the classes that need the relevant information.

For each web file in the Input Folder, the driver calls the Extractor Class, obtains the block of article text and then calls the Indexer Class which splits the text up into its individual terms and inserts them all into a Map data structure. All stop words that were loaded at start up, are removed from the structure. It also adds the text of every article to a separate String, which is used further on to obtain a list of all the terms from all the articles and produce one big Map, which becomes the terms Map. It is this map that is used for the storing of the term occurrences needed by the PLSA algorithm. The reason for using a Map data structure was because it has no duplicates and it uses a hash table for efficient referencing of its keys. Efficiency is needed because if many articles are analysed, the map can become incredibly big.

During the design phase it was opted to perform a process known as Stemming to each article. This process follows an algorithm whereby it changes each term in the article to be the base word of that term, so as to narrow down the number of terms that PLSA needs to process. It was decided during the implementation phase however, not to perform Stemming on the article data to avoid problems when comparing the data later on in the project. Because Ian Saunder is not performing stemming on the data that is displayed to the users and meant for tagging purposes, the terms that will be tagged may not necessarily line up with the terms that PLSA used in its processing. So stemming was therefore not implemented for the purposes of this project.

Lastly, the Indexing Component begins a process of building up the Term / Document Map, which is the main data structure that is output to a file and used by the PLSA algorithm. This map is structured as follows: Each key is a String value that refers to the id of the article that was processed. These ids were agreed upon beforehand for the sake of consistency. Each key then references another map data structure. Each of these maps are of the same size, and that is the number of terms that exist in all articles when put together. The keys of these maps are the String values of each term and they reference the frequency (number of occurrences) of that term as it existed in the relevant document of the main Map. A diagram depicting this structure is displayed further down as *Figure 9*.

The Indexing Component therefore outputs four plain text files to the Output Folder. They are named: “termdoc.txt”, “termdoc.html”, “terms.txt” and “docnames.txt”. The “termdoc.txt” file contains the Term / Document Matrix that is required by the PLSA algorithm. The HTML file is an easy to read table formatted version of the Term / Document Matrix. And finally, the last two files contain a list of the terms used and a list of the document filenames used, respectively.

Below is a summarised diagram of the structure of the Map data structure that is produced by the Indexing Component.

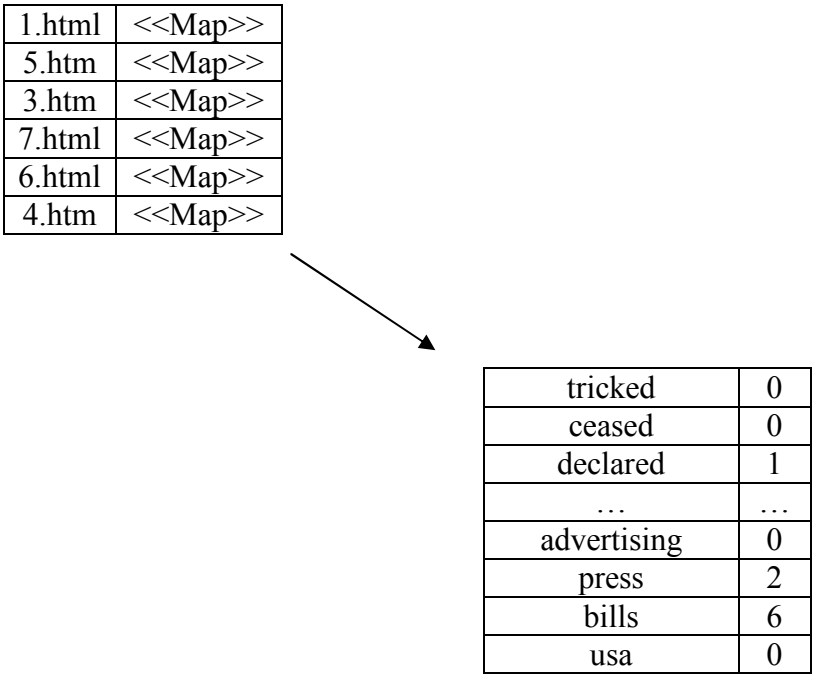


Figure 9. Structure of the Map that is produced.

5.3 PLSA Component

The implementation of the algorithm was split up into four small Matlab scripts.

5.3.1 Driver Script (run.m)

The driver script is responsible for loading the Term / Document Matrix that was produced by the Indexing Component and then executing the PLSA script with the correct parameters that are required. Once the algorithm has finished executing, the driver script will write the two matrices obtained from PLSA, to separate files. These two files are named td.txt for the Topics / Documents Matrix and wt.txt for the Terms / Topics Matrix.

The idea behind the PLSA algorithm is that it will be processing hundreds, if not thousands of documents and therefore only a selected number of topics are usually required to be produced. However in this project we are only analysing eleven documents and it was therefore decided to request the standard number of topics (128) from the algorithm as output. So the driver script therefore determines the number of terms that are in the Term / Document Matrix and request the same number of topics to be returned.

5.3.2 PLSA Script (plsa.m)

This script follows two main processes, it firstly calls the initialisation script, which performs the initialisation operations that are required and it then starts to iterate while calling the EM algorithm script.

5.3.3 Initialisation Script (init.m)

This script creates two matrices. One being the Topic / Terms Matrix and the second being the Topics / Documents Matrix, each of which are assigned initial random values in the interval [0,1]. Each matrix is then normalised according to the columns such that the sum of each column's values add up to one. The initial value of the Log-Likelihood Estimation function found below (*Figure. 10*), is then calculated according the current values in each of the matrices.

$$\mathcal{L} = \sum_{d \in \mathcal{D}} \sum_{w \in \mathcal{W}} n(d, w) \log P(d, w) ,$$

Figure 10. Log-Likelihood Estimation Function.

The PLSA script then continues by starting the loop and calls the EM algorithm script first. This is followed by the calculation of the new Log-Likelihood Estimation, which is compared to the previously calculated value. If the difference is less than 0.000001, the iterations are halted, otherwise the iterations continue until the stopping rule is reached.

5.3.4 Expectation Maximisation (EM) Algorithm Script (EM.m)

Parts of this script were obtained from the code written by Jakob Verbeek **[20]**. The use of the code is legally allowed as long as it is not for commercial use. Where necessary, the code was adapted so as to remain as close as possible to the theory that appeared in **[8]**. The script, which is called at the beginning of each iteration of the PLSA algorithm, performs the following steps: The Expectation (E)-Step involves the computation of posterior probabilities according to *Figure 11.*, based on the estimates that currently exist in the two matrices from the initialisation procedure.

$$P(z|d, w) = \frac{P(z)P(d|z)P(w|z)}{\sum_{z'} P(z')P(d|z')P(w|z')} ,$$

Figure 11. E-step posterior probability function.

The Maximisation (M)-Step then follows, at which point the parameters of the algorithm are recalculated using the new posterior probabilities calculated in the previous E-Step. These are calculated according to the equations in *Figure 12*.

$$P(w|z) = \frac{\sum_d n(d,w)P(z|d,w)}{\sum_{d,w'} n(d,w')P(z|d,w')},$$
$$P(d|z) = \frac{\sum_w n(d,w)P(z|d,w)}{\sum_{d',w} n(d',w)P(z|d',w)},$$
$$P(z) = \frac{1}{R}\sum_{d,w} n(d,w)P(z|d,w), \quad R \equiv \sum_{d,w} n(d,w).$$

Figure 12. M-step re-estimation equations.

By alternating between the EM-steps, we obtain a convergent procedure that approaches a local maximum of the log-likelihood in *Figure 10*.

The final output of these scripts is two matrices namely the Topics / Documents Matrix (TD) and the Terms / Topics Matrix (WT). The TD matrix has the original list of terms from the articles as its rows and then the requested number of topics as its columns. This matrix shows that a topic is made up of a number of terms and the terms with the highest probabilities therefore make up the specific topic in each column. The WT matrix has the requested number of topics as its rows and the number of documents as its columns. The topics with the highest probabilities are assumed to be the topics that describe what that document is about.

5.4 Linking Component

This component has three main classes, namely the PLSAReader Class, the DatabaseReader Class and the Linker Class. It also has a driver class, which is responsible for executing the selected methods in the correct sequence.

A GUI has been implemented to obtain the following necessary information from the user: The path to the Input Folder, which contains the files “td.txt” and “wt.txt” produced by the PLSA Matlab script, as well as the “docnames.txt” and “terms.txt” files that were produced by the Indexing Component. The path to the Output Folder, where the user would like the Linking component to save its output to. And lastly the information needed for connecting to the database, namely the hostname, port number, username, password and schema that contains the tables with the information obtained by the Manual Tagging System.

All the parameters are passed back to the driver program which will pass them on to the classes that need the relevant information.

5.4.1 PLSAReader Class

This class is responsible for reading in the data produced by the PLSA algorithm and then placing it into a Map data structure so that it can be easily referenced when needed by the Linker Class later on. It starts out by reading both the “td.txt” and “wt.txt” files into 2-dimensional arrays of type Double. It then analyses the TD array

and finds the topic row that has the highest probability for each document and saves the positions in a vector.

These topic positions are then matched against the WT array values and the column of term probabilities is taken from each topic column that was determined to be the topic of each document. These values are then saved to a Map data structure as follows: Each key represents a document id, which in turn references another map data structure. Each of these maps have a key element that is a string value representing each term and it references a double value which is the probability that the PLSA algorithm determined for that term to be the overall topic of that article. A diagram depicting this structure is displayed as *Figure 13*.

The final Map data structure that is produced and returned by the PLSAReader Class is shown below.

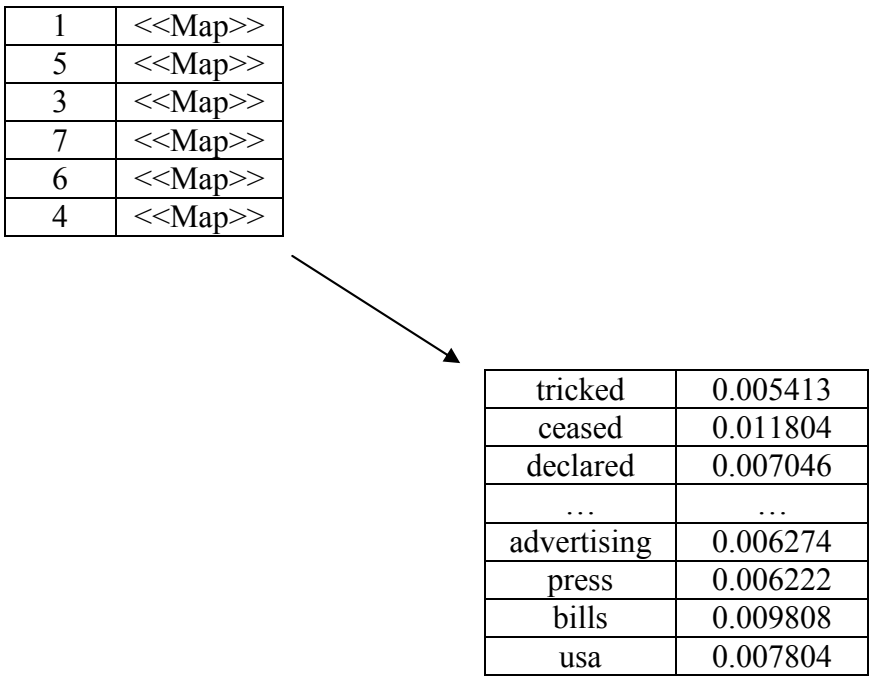


Figure 13. Structure of the Map that is produced.

The PLSAReader Class also outputs three files to the Output Folder, namely "Top10WordsDocs.html", "Top10ProbsDocs.html" and "MapTermsProbsDocs.html". These HTML files are easy to read table structures with the top ten terms found as well as their probabilities for each document as well as an overall Terms / Documents table similar to the one produced by the Indexing Component, but with the probabilities in place of the term frequencies.

5.4.2 DatabaseReader Class

This class connects to the MySQL database that is jointly used by both the Linking Component and the Facebook Application designed for the Manual Tagging System. Firstly a query is made to retrieve a list of all the articles that are available in the database. These values can be found in the "article" table and are then stored as the keys to a new map data structure.

Then for each article that was retrieved, the maximum sequence number is determined for that article, because we only want the data from the latest tag cloud. The reason for this is due to the fact that the Tagging System was designed so that it creates a new instance of the tag cloud every time a change occurs to it. The idea is to keep historical information and thus keep track of how the tag cloud has transformed since its creation. Once the maximum sequence number is retrieved, the query in *Figure 14* is executed to retrieve the data relating to every tag that has been tagged in that article.

```
SELECT tag.*
FROM tag_cloud
INNER JOIN tag ON tag.tag_id = tag_cloud.tag_id
WHERE tag_cloud.sequence_number = ?
AND tag_cloud.article_id = ?;
```

Figure 14. SQL Query to retrieve the tags for each article.

The tag information is saved to a vector, where each position references a specific item of the tag. These vectors are then added to the tag vector, which is added to the final map.

The final map data structure that this class returns can be seen below.

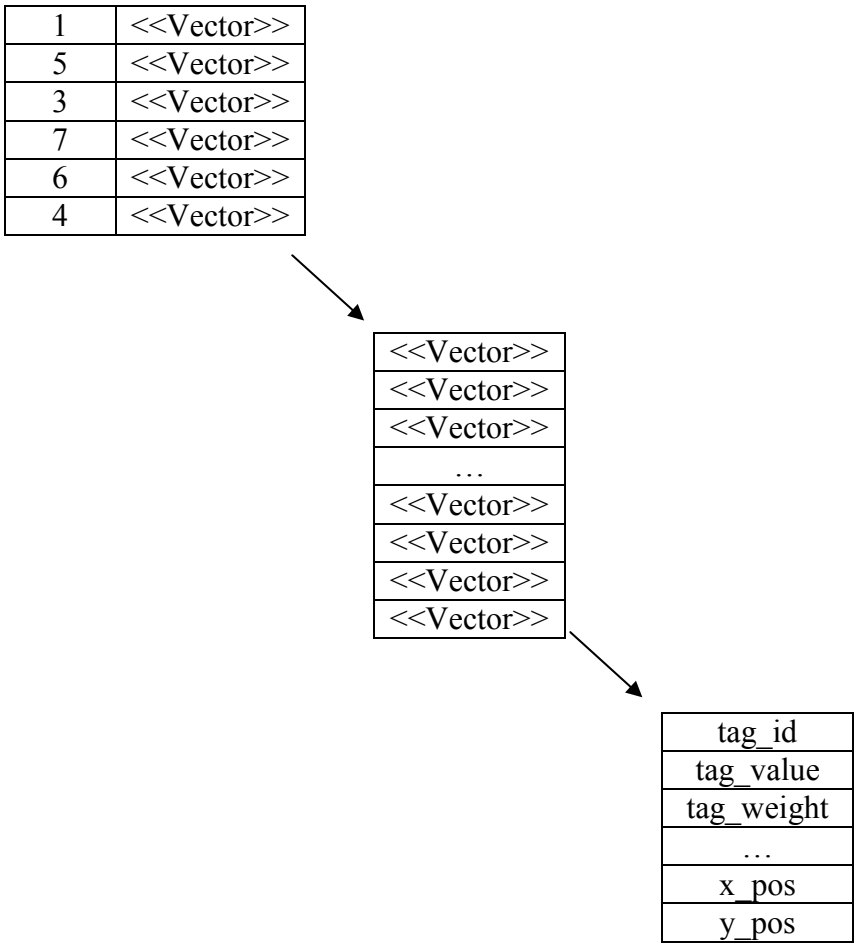


Figure 15. Structure of the Map that is produced.

In this structure the article ids are the keys to the map, each of whom reference a vector. Each of these vectors represents a tag for that article, stored in the form of another vector. These final tag vectors, as can be seen in *Figure 15*, store all the information related to a tag, such as its id, string value and weighting.

5.4.3 Linker Class

This is the class that performs the comparisons between the data obtained from the database as well as the data obtained from the PLSA algorithm. For each document in the map returned by the PLSAReader Class, it retrieves the vector of tags for that document obtained from the DatabaseReader Class as well as the map that contains the probabilities for each term in the document.

It then continues to run a loop that compares every term in the document that has been tagged by a user, with every other term that has been tagged. Each iteration of this loop performs two processes. Firstly a formula is applied to determine the Euclidian distances between each pair of terms using the data obtained from the Manual Tagging System, resulting in a conversion of the distance to a probability. And secondly, the difference in probabilities between the term pairs as obtained by the PLSA algorithm, from the Automatic Document Indexing System, is calculated. These are discussed further now.

5.4.3.1 Manual Data Distance Calculation

After deciding that the data is definitely not linear, it was chosen to represent the data using an Exponential Decay Function that converts the Euclidian distance between each tag pair as a probability. The function works as follows: For each tag cloud, the tag that has been tagged most often, is chosen and the distances between that tag and all its surrounding tags are calculated and the average weighted distance is determined from that. So therefore, if the tag cloud is cluttered with tags appearing close together, then the weighted average distance will be relatively small. And of course if they are spread far apart, then the weighted average distance will be relatively large.

The Exponential Decay function is therefore as follows:

$$e^{-\frac{dist}{avgdist}}$$

Figure 16. Exponential Decay Function.

The graphs obtained are all variations of the graph below, just with different shifts on the x-axis.

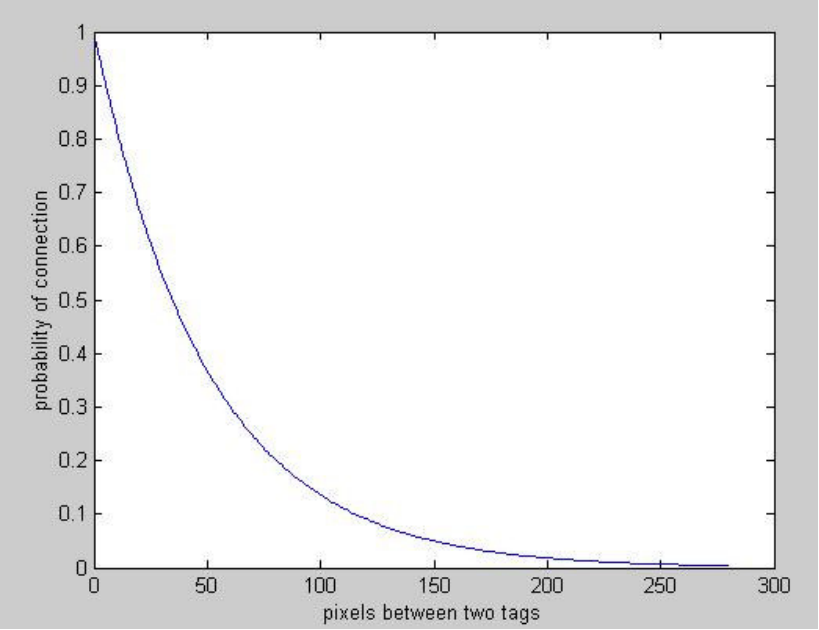


Figure 17. Exponential Decay Function Graph.

What the graph shows us is that if the tags were clustered together such that their distances are rather small, then it means that they are related, according to the users who tagged them as such, and therefore the higher their probability and vice versa.

5.4.3.2 Automatic PLSA Distance Calculation

The data produced by the PLSA algorithm is already in the form of probabilities as to the likelihood of each term being the topic of a specific document. Therefore, for the calculation of the distance between two tags using the PLSA data, it was decided to just get the absolute value of the one probability subtracted from the other, for each pair being compared.

Each of these values are then stored into file formats so that they may be used for the purpose of generating graphs and statistics for the findings of this project. The Linker Class outputs the following files to the Output Folder: "*MapTermsProbsDocs.html*", which contains all the probabilities that was produced by the PLSA algorithm, neatly laid out in an HTML table structure so that a user can easily find any probability that they need. "*Top10WordsDocs.html*", which neatly displays ten terms that were found to have the highest probabilities of being the topics of each relevant document. "*Top10ProbsDocs.html*", which is very similar to the previous file, except that it contains the probabilities of the top ten words for each document. And lastly, "*WordComparisons.html*", which contains table structures of every tag in each cloud compared to every other tag and the distances that were calculated between them, so that a user can easily reference any tag comparison that they need to.

The final output from this class is the creation of a table in a MySQL database, using the information that was retrieved from the GUI. The table contains the same data

that was output into the "*WordComparisons.html*", except that it is ordered according to a comparison id. The reason for creating a table, is so that the data can be stored in the same database as where the user data is stored by the Manual Tagging System and so that the OntoCreator Class can easily retrieve it through a SQL query and perform all the functions that it needs to.

5.5 OntoCreator Component

This class is responsible for producing an XML structured file that represents the data obtained from the joined processes of the PLSA algorithm as well as the user entered tagging data obtained from the Manual Tagging System. It is of a proprietary format designed by both the Author and Ian Saunder and represents the data in a neat format structured according to the following XML Schema:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:element name="word" type="xs:string" maxOccurs = "unbounded"/>
        <xs:complexType>
            <xs:sequence>
                <xs:element name = "related-word" type = "xs:string" maxOccurs = "1">

                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name = "value" type = "xs:decimal"
                                        maxOccurs = "1" />
                            <xs:element name = "probability" type = "xs:decimal"
                                        maxOccurs = "1" />
                        </xs:sequence>
                    </xs:complexType>

                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

</xs:schema>
```

Figure 18. XML Schema

6 Testing, Evaluation and Findings

Through the course of this project a number of evaluations were performed using the data that was obtained from both general users, experienced users as well as from the output of the implementation. These evaluation techniques will be presented and discussed in detail now and the results obtained will be evaluated as well.

6.1 Comparison of Tag Pair Distances from all Tag Clouds with PLSA Distances

The purpose of this evaluation was to determine if some sort of correlation existed between the data obtained from the output of the PLSA algorithm and the data obtained from the users and their activities, while using the implementation from the Manual Tagging System.

6.1.1 Collection of Data

The data collected from the Manual Tagging System was obtained as follows. Part of the implementation consisted of the design and implementation of a User Tagging System that was deployed as an application on the popular social networking website Facebook.com.

The reason for creating it as a Facebook Application was due to time constraints and the need to reach as many users as possible. The system contained eleven short news articles obtained from the popular news division of the Uncyclopedia.org website, called UnNews.

The users were asked to read an article and they were then provided with a tag cloud as it currently existed for that article. They were requested to tag one word from the article that they felt described the article as a whole, whether it already existed in the cloud or not, and then to shift the existing tags around such that, tags that are linguistically related to each other, are near to each other according to distance and vice versa.

On the date when the final data capturing took place, the website had a total number of 309 recorded tags from eleven different articles. These taggings were collected from the MySQL database as discussed in *Section 4.4* and once properly structured; they were ready to be used.

The second source of the data was the data output by the PLSA algorithm. This was the author's section of the project and as discussed in *Section 4.4* was nicely structured and ready for use. This is due to the fact that the PLSA algorithm is an automatic document indexing algorithm and only needed to be executed once using the eleven articles that were chosen to be displayed by the Facebook application. No user interaction was required.

6.1.2 How the data obtained was used

As it was already discussed in great detail in *Section 5.4.3.1* how the data obtained from the user taggings was converted from Euclidian distances between tag pairs to probabilities that reflect the association between each pair of taggings, it will not be mentioned here again.

However since the probabilities calculated from the data are a reflection of a linguistic association between the two tags, the data obtained from the PLSA algorithm needed to be used such that it too reflected this. This was achieved by taking the following into account.

Each term in the article has a probability assigned to it by the PLSA algorithm, which means that each term has that assigned probability of being the topic for the article that it appeared in. So if two terms had probabilities that were very much the same, meaning the PLSA algorithm determined them both to be very high possibilities of being the topic for the document they both appeared in, then it was assumed that these two words could quite possibly be linguistically associated to each other.

It was therefore decided to take each tag pair and get the probabilities of the two words and to get the abstract value of their difference. Therefore the smaller the difference calculated, the closer the association between the two tags and vice versa.

6.1.3 Graph Results

The above results were plotted on a scatter graph produced in Matlab and looked as follows.

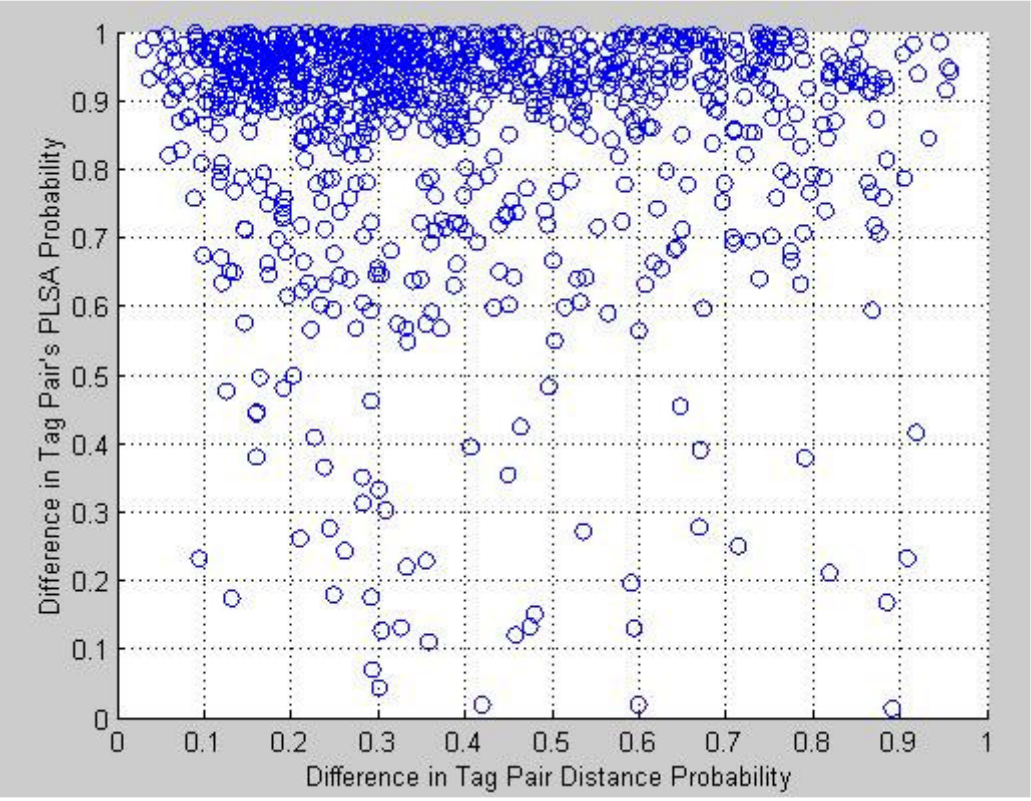


Figure 19. Graph plotting the distance probabilities between the tagged and PLSA data.

It was the hope of this research project that these values plotted together on the same graph would reveal some sort of pattern that would show a correlation between the automatic PLSA method and the manual user entered method. This was however not the case at all.

The data obtained from the PLSA algorithm were all rather low probabilities and even by calculating the differences between them for each term pair, the results obtained were still incredibly small. It was also difficult to scale the results such that they were on the same probability level as the data obtained from the Facebook Application, without actually ending up with results that were not a true reflection of what the PLSA algorithm had provided.

It was therefore decided to multiply each probability value by ten, to scale the values just a bit closer to those obtained from the Tagging System. Since a very small value reflected a stronger linguistic association, it was also decided to use the expression *1-probability* in order to obtain high probabilities to match those from the Tagging System.

This type of scaling however resulted in really high probabilities for most of the term pairs, which resulted in the graph being clustered near the top. It was therefore not a true reflection of term associations and could therefore not be used as such.

6.2 Comparison of Tag Frequencies from all Tag Clouds with PLSA Probabilities

The purpose of this evaluation was to determine if there was any significant correlation between the frequency of all tags, i.e. the number of times each tag was tagged in each Tag Cloud, and the data obtained from the PLSA algorithm. The reason for this was due to the fact that the PLSA algorithm relies quite a bit on the frequencies of each term in each document. It was the author and Ian Saunder's hope that a higher PLSA probability would relate to the same terms that were tagged numerous times by the users using the Facebook Application.

6.2.1 Collection of Data

For the purpose of this evaluation, the frequency of each tag was needed from the MySQL database. The way the database was designed according to *Figure 7* in *Section 4.4*, this was easy to obtain using the following query:

```
SELECT t.tag_value, COUNT(t.tag_value) AS freq, tc.article_id
FROM tag t, tag_cloud tc
WHERE t.tag_id = tc.tag_id
AND user_entered = 1
GROUP BY tag value;
```

Figure 20. Query for retrieving the frequency of all tags in all tag clouds.

The PLSA data already retrieved, was enough for the purpose of this evaluation.

6.2.2 How the data obtained was used

Because the probabilities from the PLSA data were already so small, it would not make sense to plot the frequency integer values directly to them. The frequency values were therefore divided by a ten to scale them down to a more reasonable value to match the PLSA data, which were multiplied by ten for scaling up purposes.

6.2.3 Graph Results

The above results were plotted on a scatter graph produced in Matlab and looked as follows.

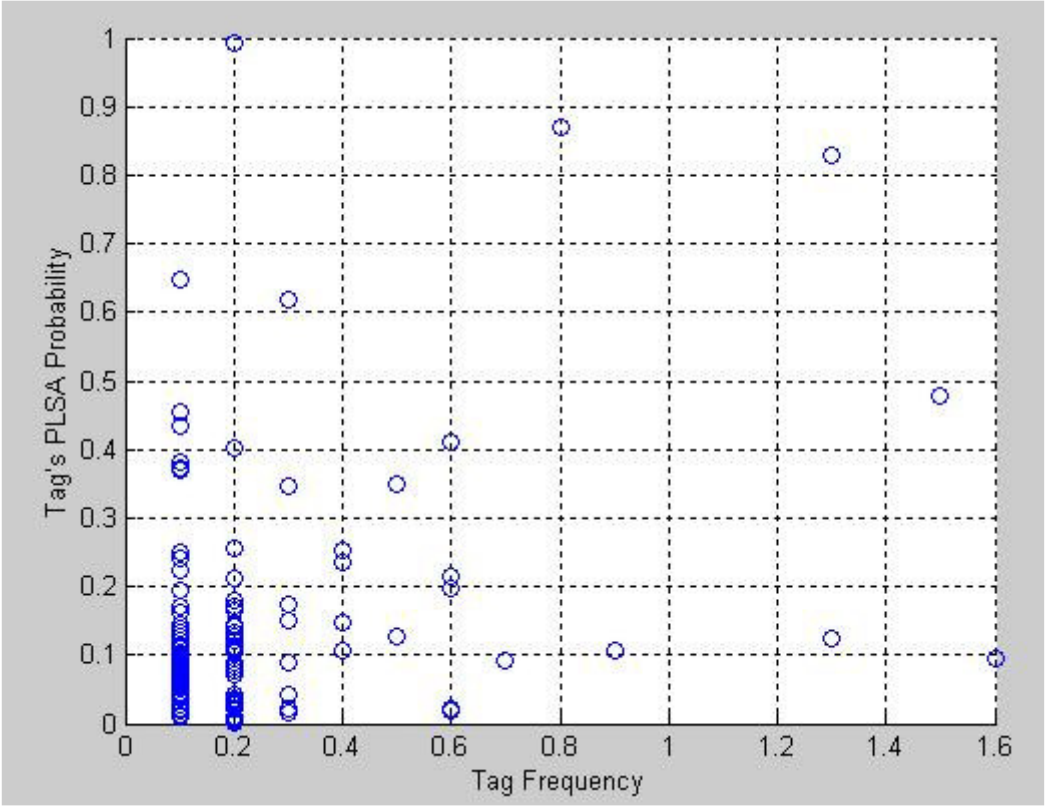


Figure 21. Graph plotting the frequency of each tag in each tag cloud against the PLSA probability obtained for each tag.

Once again there was no distinguishable pattern between the frequencies and the tag probabilities. The reason for the concentration of plots near the bottom left corner of the graph is due to the fact that many of the tags were only tagged once, maybe twice, hence the values falling mainly between the range [0.1, 0.2].

Once again the results were not ideal, as no pattern can be found according to the above graph that shows an association between the frequencies of the tags and their PLSA probabilities.

6.3 Using Data retrieved from Expert Users, to determine the accuracy of the PLSA algorithm

This evaluation was performed through a means of sending out a questionnaire to five Post-Graduate students, all male, in the Computer Science Department at the University of Cape Town who were currently in the process of obtaining their Masters in either Computer Science or Information Technology. Now although each of these students may not be experts in the field of language or document analysis, they do appear to at least have a certain level of credibility so that their answers in the

evaluation can be trusted to be honest. Also due to the fact that they are at a Masters level of education, places them in a bracket with an above average intellectual capacity to be able to complete what is needed of them for the purposes of this evaluation.

The questionnaire can be found in *Appendix E* and it asked each user to read each of the eleven articles that were chosen for the purposes of this research project. They were asked to fill in three terms that existed in each article, which they perceived as being a good reflection of what each article as a whole was about. For each document, the top ten terms that were produced by the PLSA algorithm, were matched against the three terms obtained from the users and averages and graphs were produced.

6.3.1 Graph Results

The graphs of the individual users can be found in *Appendix D* and the graph of the averages is displayed below.

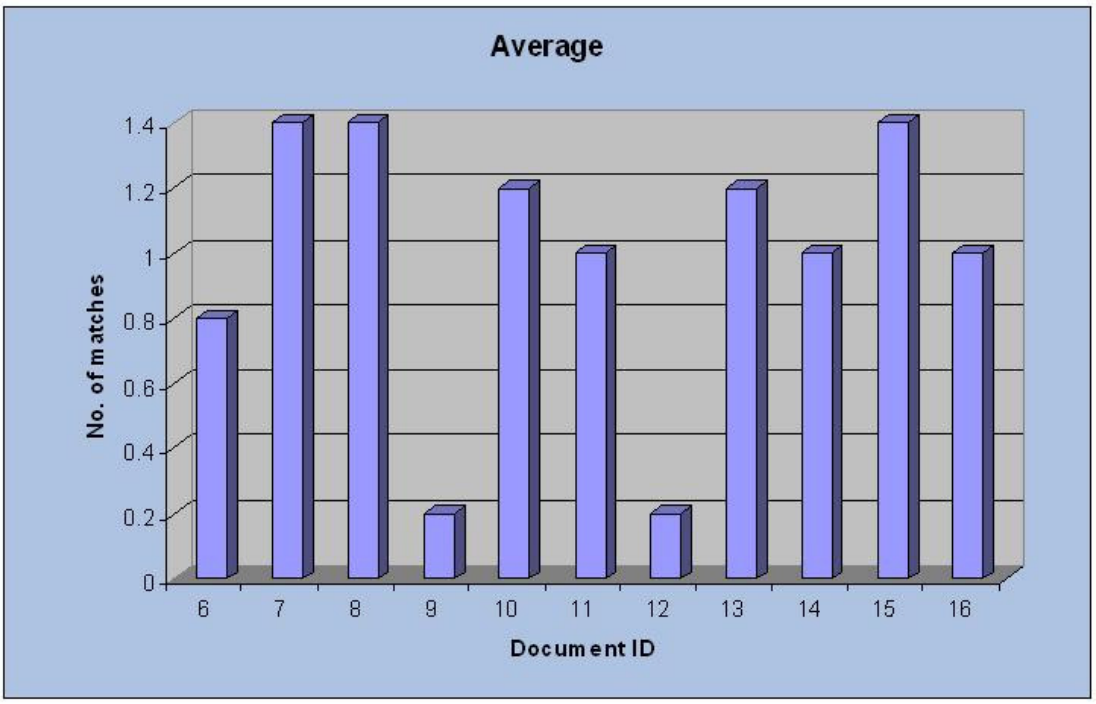


Figure 22. Graph displaying the average number of matches for each document.

What the above graph shows is that for any of the articles presented, at most only one or two of the terms obtained from the MSc students were found to be a match with the data produced by the PLSA algorithm.

Due to time constraints however, no more than five MSc students could be approached and asked to take part in the evaluation, and therefore these results are not ideal in any form. However to conclude based on the results that were obtained here, it does seem that the results from the PLSA algorithm that show which terms were found to be the topics of a document are generally not as accurate as how the user intuitively perceives the topics of a document to be.

6.4 Evaluation based on the three best Tag Cloud structures

This was the final evaluation that was performed for this research project. After the final data from the Facebook Application was collected, the structure of each tag cloud was analysed and the three best looking clouds were chosen for this evaluation, i.e. the clouds are structured well and the tags are spread nicely so that a user can easily understand its meaning.

The three chosen articles had the following titles: 1) "*A nice cup of tea and a sit down proven to cure all human ills*", 2) "*Doomed Astronauts Optimistic*" and 3) "*Gas stations begin giving away free gasoline*".

The tag with the highest frequency in each of these three clouds was taken and paired with every other tag in the cloud and a questionnaire, that can be found in *Appendix E*, was drawn up. The questionnaire was handed out to 50 anonymous users, both male and female, with no particular specialisation.

Each user was asked to rank the linguistic association between each word pair as described earlier, on a scale from one to ten. Where a rank of one would mean there is no association whatsoever and a rank of ten means there is a very strong association between the two terms. The data obtained from these users was captured into an excel spreadsheet. A number of statistical calculations were produced using this data, the results of which are discussed further down.

6.4.1 How the data obtained was used

The data from the questionnaire were divided by ten to obtain them as probabilities between the interval [0, 1]. It was then decided to take the previously calculated distance probabilities from the Tagging System that was mentioned in *Section 6.1* as well as the PLSA probabilities of the same tag pairs and graphs were produced displaying these three probabilities paired together.

6.4.2 Graph Results

The graph related to the "*Tea*" article is shown below.

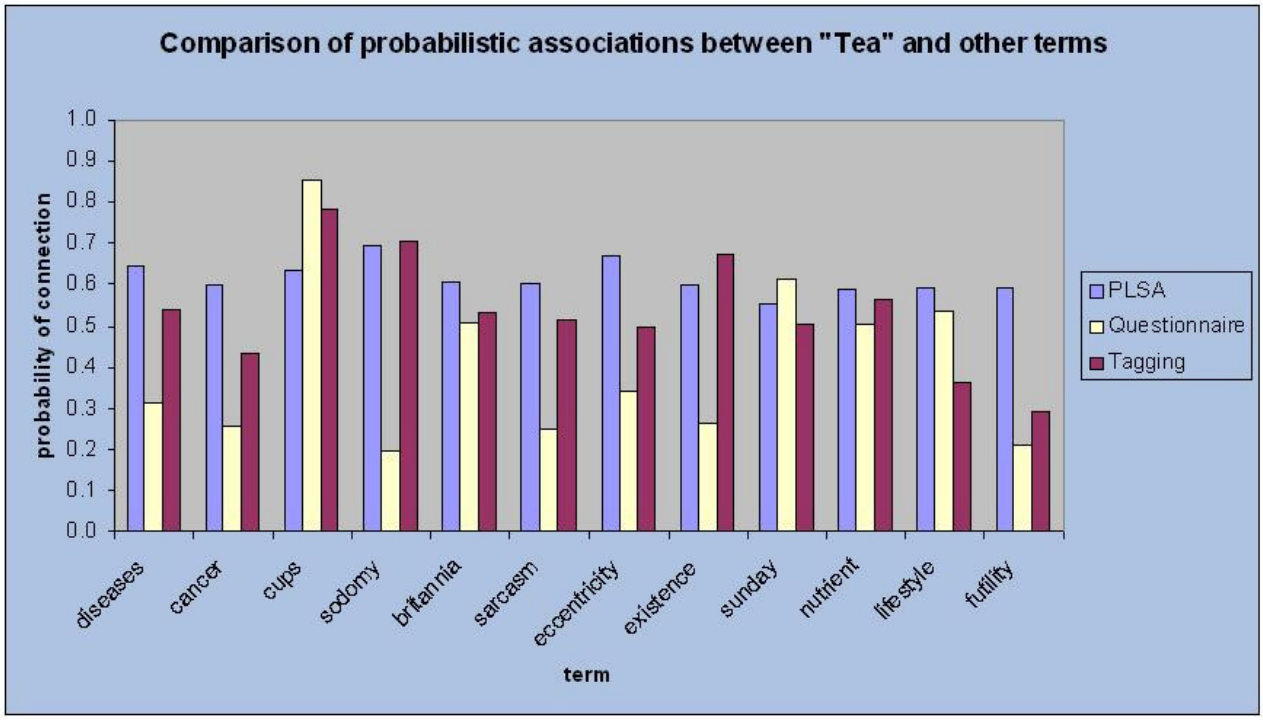


Figure 23. Graph representing Tagging, PLSA and Questionnaire data together for the article related to the term "*Tea*".

The results obtained show that for this article, the PLSA data and the user entered data from the Tagging System seem to be relatively matched up, and also having slightly higher probabilities when the data from the questionnaire assumed otherwise.

The graph related to the "*Astronaut*" article is shown below.

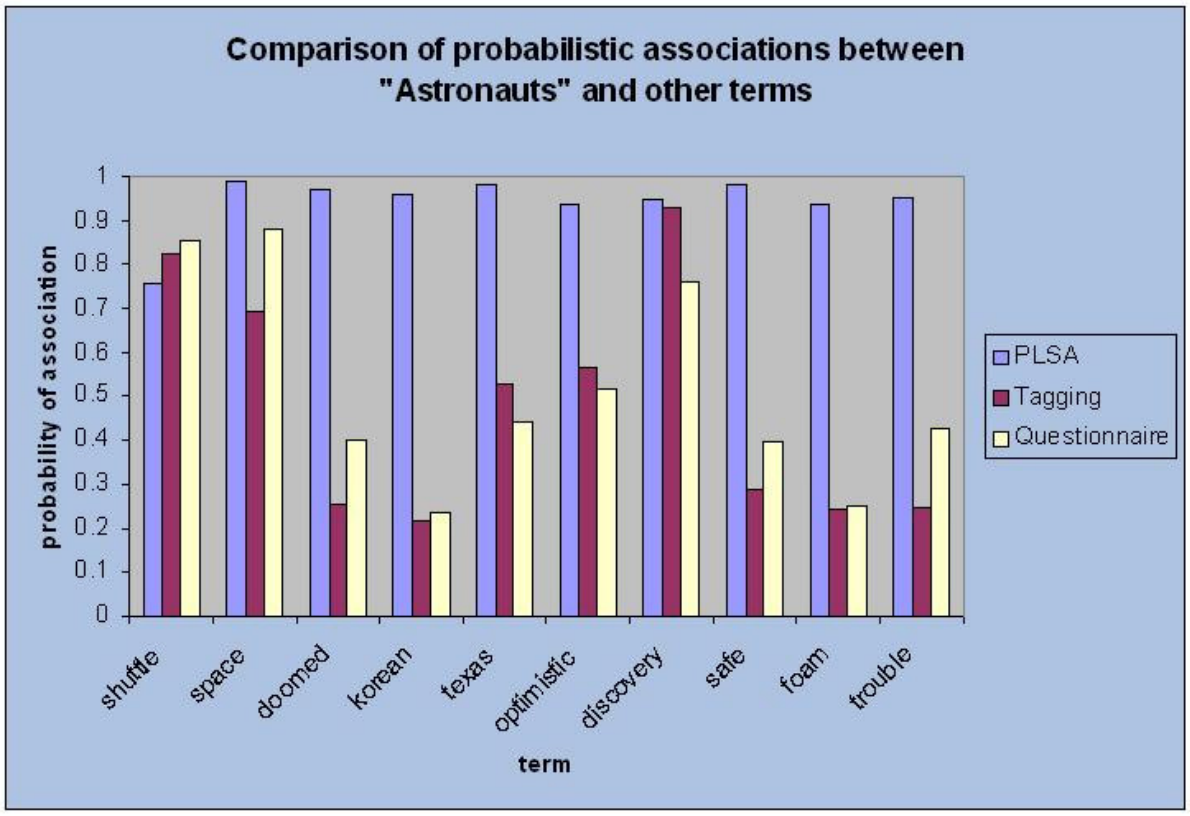


Figure 24. Graph representing Tagging, PLSA and Questionnaire data together for the article related to the term "*Astronaut*".

In this article the PLSA results are a lot higher than the previous article. It also shows that the data from the Tagging System is more accurately matched with the questionnaire data, proving that the PLSA algorithm was not as accurate in predicting the correct probabilities for this article.

The graph related to the "*Gas*" article is shown below.

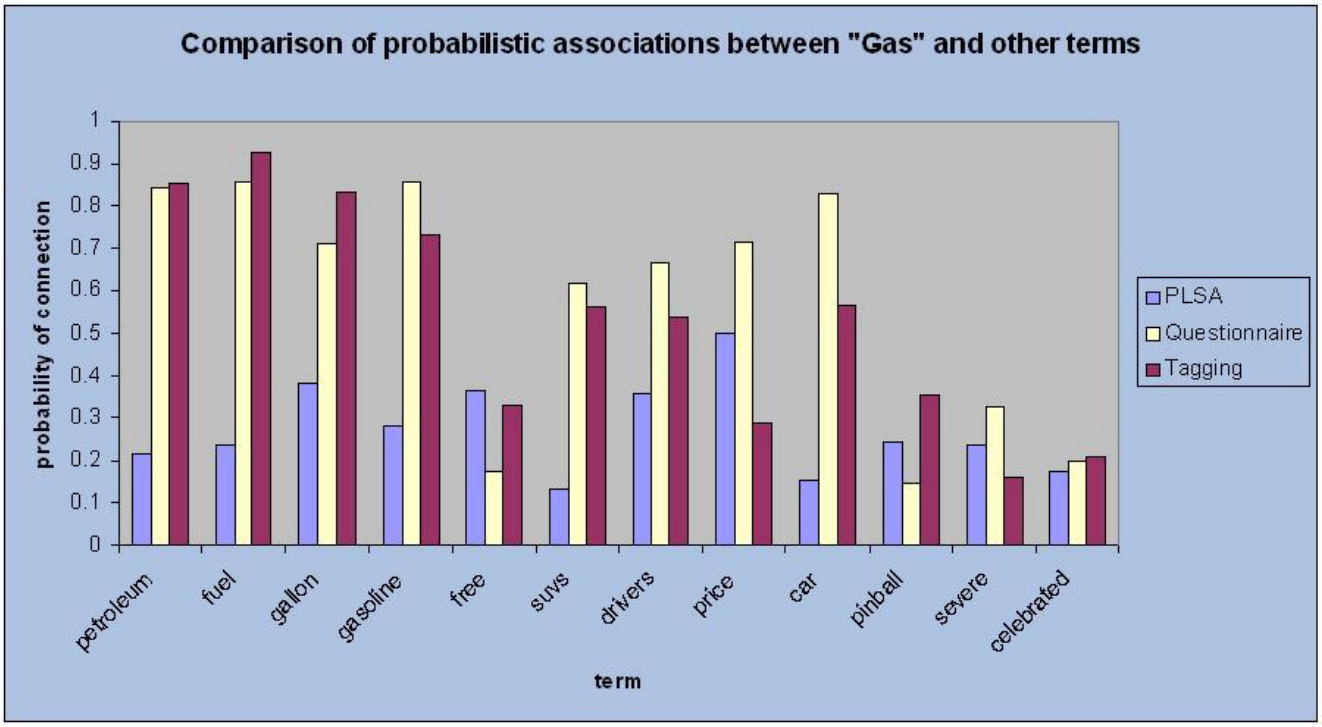


Figure 25. Graph representing Tagging, PLSA and Questionnaire data together for the article related to the term "*Gas*".

The opposite has occurred in this article, where the PLSA values are now much smaller than their Manual Tagging System counterparts and once again the manual data seems to be a better match to the questionnaire data than the PLSA data.

In conclusion to the above three graphs, it was determined that the PLSA algorithm can produce unpredictable values for an analysed article. When compared to user entered data based upon how users intuitively perceive the linguistic association between two terms, this method of using the difference between term probabilities from the PLSA data is not an accurate measure of the association between two terms.

6.5 Conclusion

The PLSA algorithm did therefore not perform as well as was hoped for the purpose of this project. Even though the results obtained from numerous studies, some that appeared in **[1, 2, 7, 8],** showed that PLSA outperformed many of its Document Indexing counterparts, it was still not a true reflection of what a user intuitively perceived the topics of a document to be. This may well be due to the fact that PLSA relies on the use of frequencies of terms in each document analysed, whereas users do not immediately take note of this when evaluating an article. The articles used in this research project were also very short articles; the reason for this was to avoid having users perform the lengthy and tedious task of having to read each article during a controlled experiment, and therefore not performing as is needed for the purposes of the project. The PLSA algorithm does however produce more accurate results when used to evaluate a large number of lengthier documents **[8]**.

7 Conclusion

7.1 Summary of the Research Project

After performing the necessary evaluations needed, it can be concluded that the PLSA algorithm did not performed as well as was hoped by the developer. This was due to the fact that only short articles were used in the Manual Tagging System so as to attract user interest and to hold their attention while performing the evaluations. The PLSA algorithm does produce better results when it is performed on a much larger number of documents as well as when the documents are lengthier than those used in this research project.

Secondly, it can be concluded that by obtaining the difference between two term probabilities from the PLSA data, the result can not be used as a true reflection of the linguistic association between these pairs of terms.

Lastly, when evaluation takes place requiring users to enter data based upon how they interpret an article, while taking numerous factors into account, such as length of the article, the already existing tags in the tag cloud, time that the user has available and the usability of the system they need to use, the data obtained cannot be controlled in any way. This allows users to enter any random data that they see fit, which can alter the outcome of the experiment. As was seen in the evaluation in *Section 6.1*, the data entered by the users during the experiments could not be associated directly with the PLSA data obtained from the Automatic Indexing System. This was due to the accuracy of the PLSA algorithm as well as incorrect data obtained from the users, but it also proved that what the users intuitively perceived to be the topic of a particular article, was mostly not the same as the results produced by the PLSA algorithm.

7.2 Lessons Learnt

The developer realised that time management is an important factor for the successful outcome of any project. Due to not properly managing time in this research project, the required number of evaluations needed to produce more accurate results, could not be obtained.

Effective communication between the partners of a research project is incredibly important, as joint decisions need to be made regarding the design of systems that will be used by subsystems of both sections of the project. The developer feels that this was successfully achieved and no problems were encountered due to this aspect.

8 Future Work

This section presents possible work that could be implemented to extend or improve the outcome of this research project.

8.1 Improvement of the PLSA Algorithm

In *Section 2.2.2*, a variation of the original EM algorithm used during the iterations of the PLSA algorithm, is presented. This variation of the algorithm was not implemented in this research project, however it has been proven that the use of this algorithm will improve the accuracy of the PLSA data, because it avoids the problem of overfitting the data **[8]**. With these new and improved results, a possible stronger connection may be observed between the PLSA data and the data obtained from the Manual Tagging System.

8.2 Correct construction of an Ontology

Due to the lack of the correct data required to create a proper Ontology in the true sense of the term, it may be possible to extend this project, exploring other alternatives to obtain associations between the PLSA data and the Manual Tagging System data. With these, the PLSA data can be used to adjust the weightings that represent the associations between term pairs, thus resulting in a proper ontology design.

8.3 Visualisation of the Ontology

The use of Hypergraphs can also be used for the purpose of visualising the ontology created in *Section 5.5*. Another option could be to represent the ontology using Topic Maps, which extends the project to another area of research currently being explored today.

9 References

[1] Oates, T., Bhat, V., ShanbHag, V. *Using Latent Semantic Analysis to Find Different Names for the Same Entity in Free Text*. Proceedings of the 4th international workshop on Web information and data management WIDM '02. Nov 2002.

[2] Foltz, W. *Using Latent Semantic Indexing for Information Filtering*. Proceedings of the ACM SIGOIS and IEEE CS TC-OA conference on Office information systems. Cambridge, Massachusetts, United States. 1990.

[3] Papadimitriou, C., Raghavan, P., Tamaki, H., Vempala, S. *Latent Semantic Indexing: A Probabilistic Analysis.* Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems. Seattle, Washington, United States. 1998.

[4] Cai, L., Hofmann, T. *Text Categorization by Boosting Automatically Extracted Concepts*. Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval. Toronto, Canada. 2003.

[5] Shah-hosseini, A., Knapp, G. *Semantic Image Retrieval Based on Probabilistic Latent Semantic Analysis*. Proceedings of the 14th annual ACM international conference on Multimedia. Santa Barbara, CA, USA. 2006.

[6] Ding, C. *A Similarity-based Probability Model for Latent Semantic Indexing.* Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. Berkeley, California, United States. 1999.

[7] Brants, T., Chen, F., Tsochantaridis, I. *Topic-Based Document Segmentation with Probabilistic Latent Semantic Analysis*. Proceedings of the eleventh international conference on Information and knowledge management. McLean, Virginia, USA. 2002.

[8] Hofmann, T. *Probabilistic Latent Semantic Indexing*. Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. Berkeley, California, United States. 1999.

[9] Bestgen, Y. *Improving Text Segmentation Using Latent Semantic Analysis: A Reanalysis of Choi, Wiemer-Hastings, and Moore (2001)*. Computational Linguistics, Volume 32 , Issue 3. 2006.

[10] Yan, R., Hauptmann, A. *Probabilistic Latent Query Analysis for Combining Multiple Retrieval Sources*. Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. Seattle, Washington, USA. 2006.

[11] Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R. *Indexing by Latent Semantic Analysis*. Journal of the American Society for Information Science. 1990.

[12] Hofmann, T. *Unsupervised Learning by Probabilistic Latent Semantic Analysis*. Machine Learning Journal, 42(1), pp.177.196. 2001.

[13] Alaniz, A., Pimentel, M., Guerrero, J. *Latent Semantic Linking over Homogeneous Repositories***.** Proceedings of the 2001 ACM Symposium on Document engineering. Atlanta, Georgia, USA. 2001.

[14] Dumais, S., Furnas, G., Landauer, T., Deerwester, S., Harshman, R. *Using latent semantic analysis to improve access to textual information.* Proceedings of the SIGCHI conference on Human factors in computing systems. Washington, D.C., United States. 1988.

[15] Gong, Y., Liu, X. *Generic text summarization using relevance measure and latent semantic analysis*. Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval. New Orleans, Louisiana, United States. 2001.

[16] Landauer, T., Foltz, P., Laham, D. *An Introduction to Latent Semantic Analysis*. Discourse Processes, v25 n2-3 p259-84. 1998.

[17] "*The MathWorks – Accelerating the pace of engineering and science*" [Online] Available at: http://www.mathworks.com/access/helpdesk/help/techdoc/index.html. Accessed: 13 September 2007

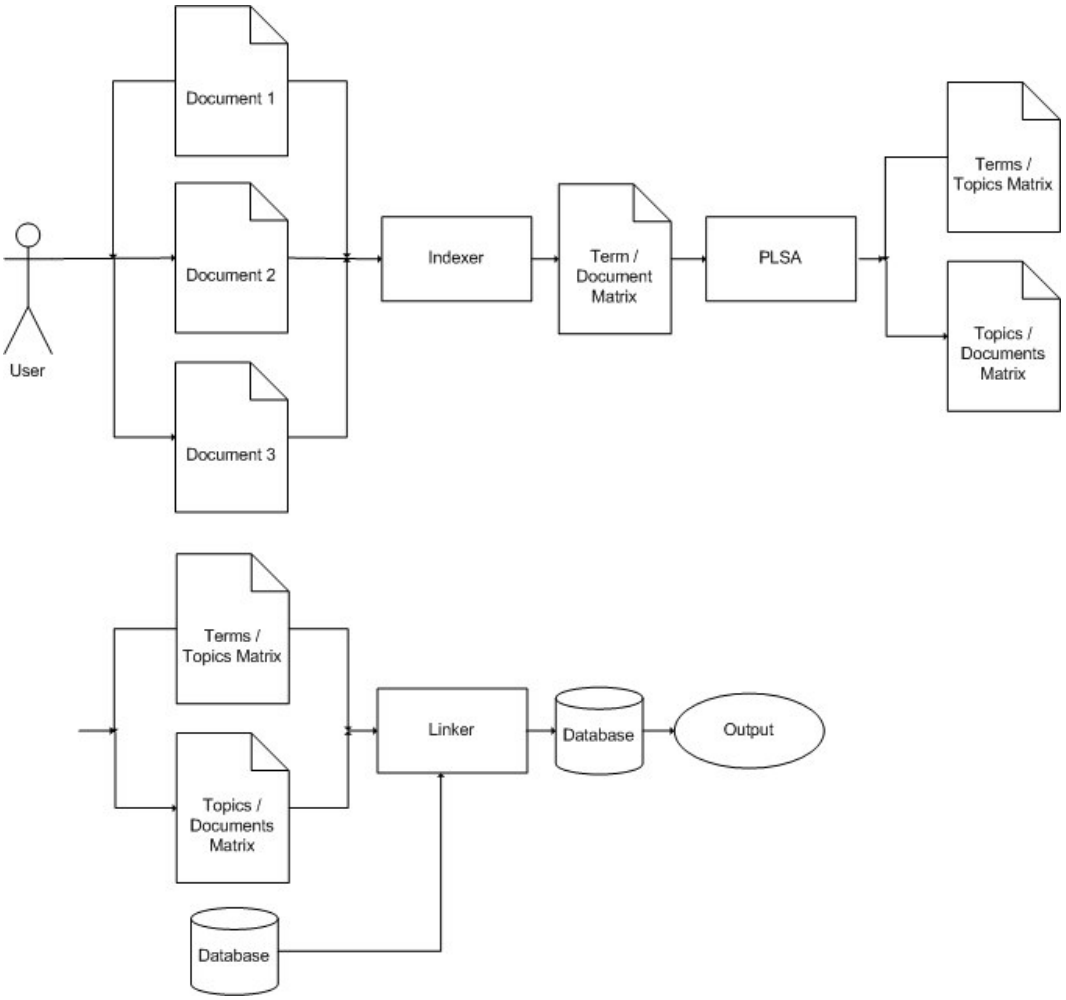
[18] Hahn, BD., *Essential MATLAB for scientists and engineers Third Edition*, Maskew Miller Longman, Cape Town, 2002.

[19] Underhill, L., Bradfield, D., *Introstat Second Edition*, Juta & Co., South Africa, 1994

[20] Verbeek, J., *Probabilistic Latent Semantic Analysis*, [Online], Available at: http://lear.inrialpes.fr/~verbeek/software, Accessed on: 13 September 2007

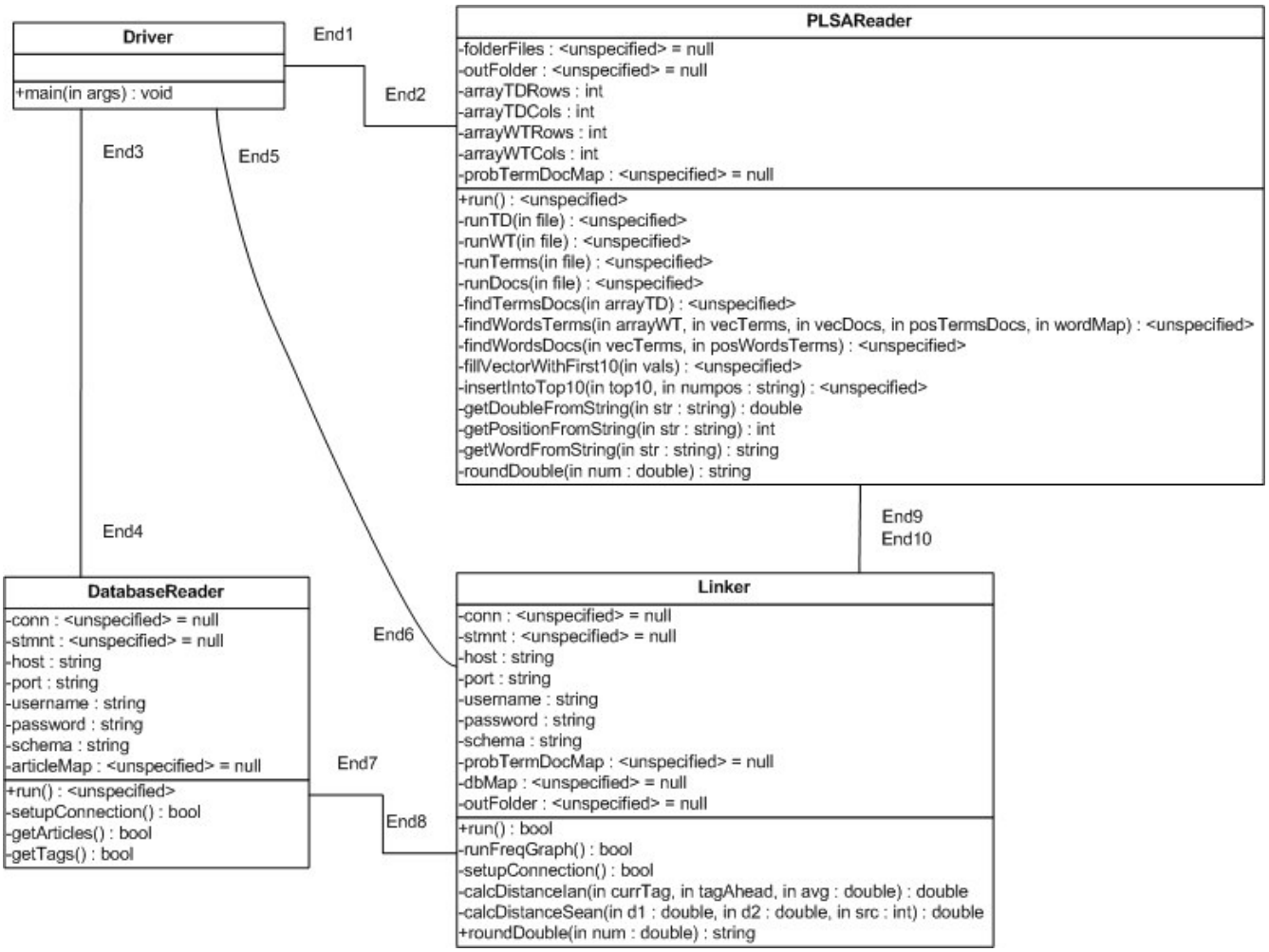
[21] Jericho HTML Parser Java Library, [Online] Available at: http://jerichohtml.sourceforge.net/doc/index.html. Accessed on: 31 August 2007

Appendix A



Appendix A. High level Design Process

Appendix B



Appendix B. UML Class Diagram of the Linking Component

Appendix C

Final Tag Cloud States

Value	Frequency	Position
alonso	13	93, 111
car	2	106, 135
emotions	1	145, 84
champion	3	64, 54
mclaren	2	54, 65
current	1	78, 58
hamilton	2	48, 51
wailing	2	111, 112
dirty	2	121, 113
squatting	1	142, 72
defeat	1	97, 144
rookie	2	43, 66
turd	2	150, 75
curling	1	20, 123

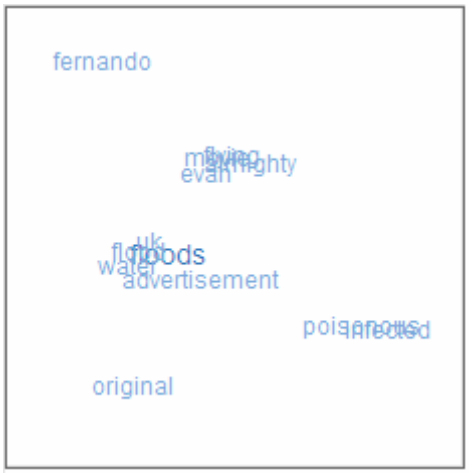
Value	Frequency	Position
alonso	13	93, 111
car	2	106, 135
emotions	1	145, 84
champion	3	64, 54
mclaren	2	54, 65
current	1	78, 58
hamilton	2	48, 51
wailing	2	111, 112
dirty	2	121, 113
squatting	1	142, 72
defeat	1	97, 144
rookie	2	43, 66
turd	2	150, 75
curling	1	20, 123

Value	Frequency	Position
tea	15	117, 104
diseases	1	136, 56
cancer	1	156, 46
cups	2	136, 111
sodomy	1	101, 80
britannia	1	84, 63
sarcasm	2	62, 97
eccentricity	2	85, 56
existence	1	117, 137
sunday	1	150, 151
nutrient	1	146, 66
lifestyle	2	58, 165
futility	1	10, 123

Value	Frequency	Position
loser	6	113, 133
virgin	6	120, 117
spelling	2	92, 117
nyarlathotep	1	90, 143
champion	3	144, 68
tool	1	119, 130
prostitute	2	48, 21
condemnation	3	92, 161
booty	1	44, 14
cheerleaders	2	154, 51
sexually	2	57, 23
target	1	80, 101

Value	Frequency	Position
facebook	16	78, 155
student	1	109, 153
theft	3	112, 92
accused	1	149, 89
blondes	1	84, 66
elfin	1	72, 48
forbidden	1	118, 68
face	1	84, 134
condemnation	3	148, 67
boston	1	91, 136
furious	1	154, 145
offenders	1	26, 123

Value	Frequency	Position
bush	13	102, 85
president	2	126, 110
americans	2	132, 105
books	1	66, 154
washington	1	128, 118
chimp	4	90, 52
bubbles	1	126, 58
free	4	51, 83
extra	1	91, 156
read	2	54, 149
leader	1	155, 128
nuclear	2	179, 21
natural	1	174, 176
remarkable	2	174, 173



Value	Frequency	Position
floods	7	77, 124
animal	1	17, 123
advertisement	3	93, 138
fernando	1	47, 29
poisonous	1	175, 161
almighty	2	120, 80
uk	3	72, 118
flood	2	68, 124
evan	1	100, 84
infected	1	190, 162
water	1	62, 131
original	1	60, 189
flying	1	112, 76
movie	1	104, 77

Value	Frequency	Position
astronauts	9	92, 85
shuttle	3	91, 101
space	1	63, 95
doomed	1	187, 24
korean	1	168, 188
texas	2	42, 67
optimistic	2	53, 112
discovery	1	86, 85
safe	1	55, 182
foam	2	33, 188
trouble	1	193, 28

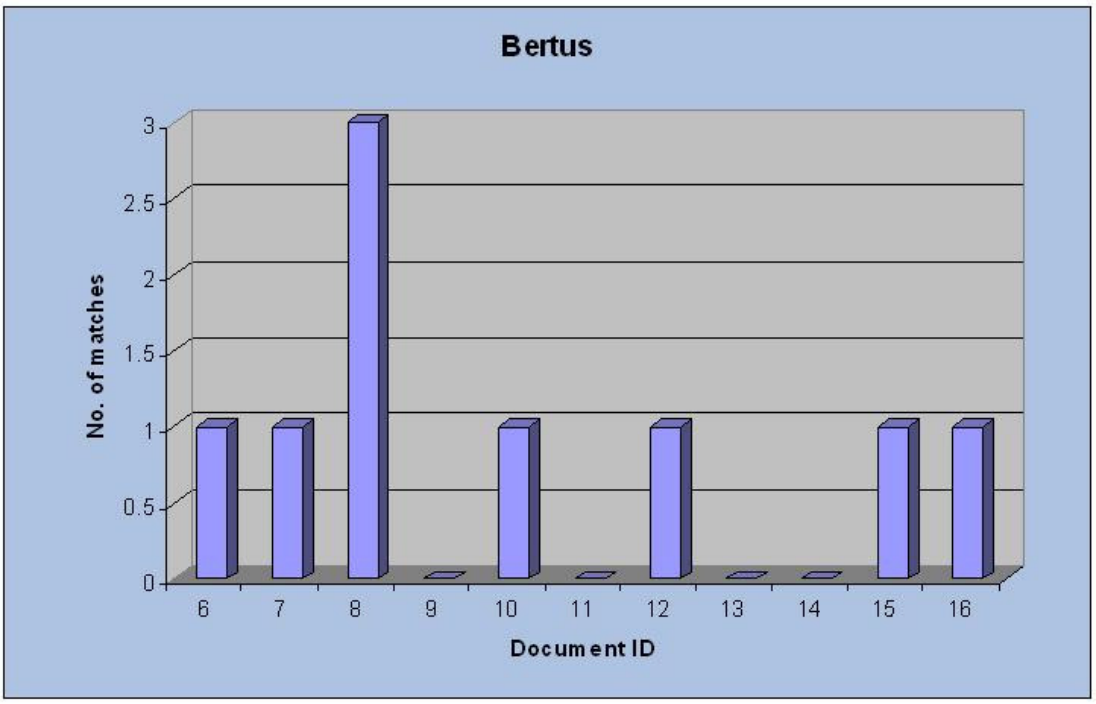
Value	Frequency	Position
gas	8	76, 123
petroleum	1	89, 123
fuel	1	70, 121
gallon	1	82, 109
gasoline	3	102, 124
free	4	157, 78
suvs	2	31, 106
drivers	1	41, 85
price	1	161, 65
car	2	34, 101
pinball	1	148, 171
severe	1	190, 22
celebrated	1	170, 214

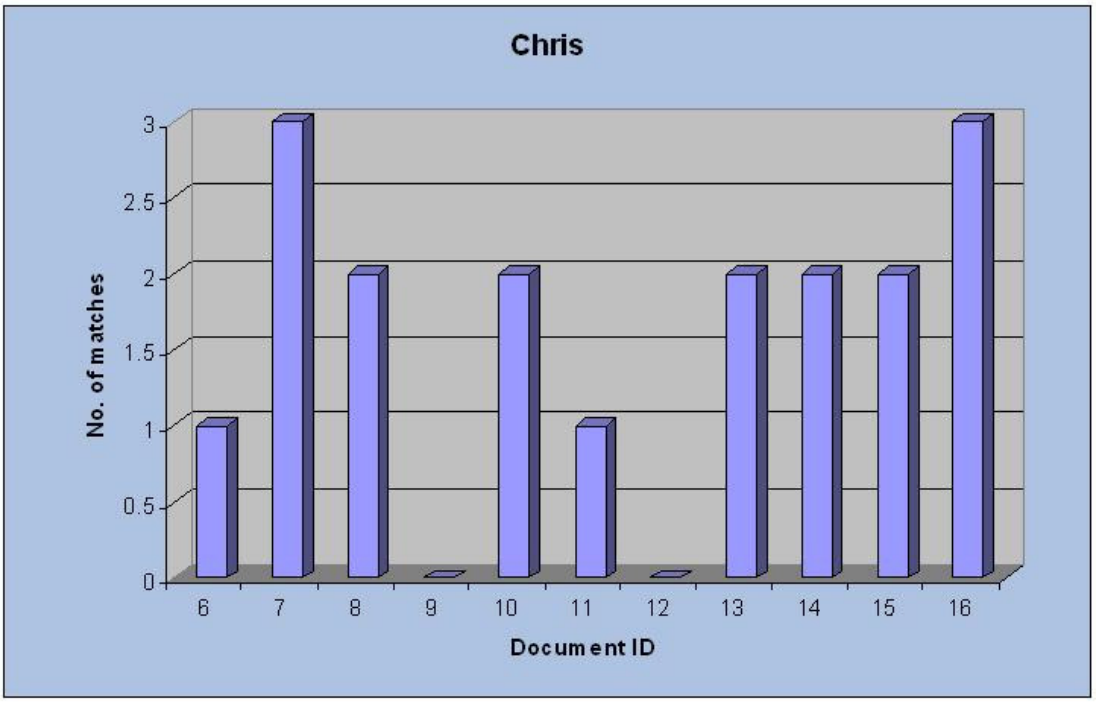
Value	Frequency	Position
happiness	6	133, 105
australia	4	72, 66
penalty	1	173, 128
potato	6	128, 116
customs	1	70, 112
sydney	1	60, 61
parcel	2	73, 137
rugby	2	140, 100
ireland	1	84, 74
outlawed	1	161, 58
outlandish	1	27, 123

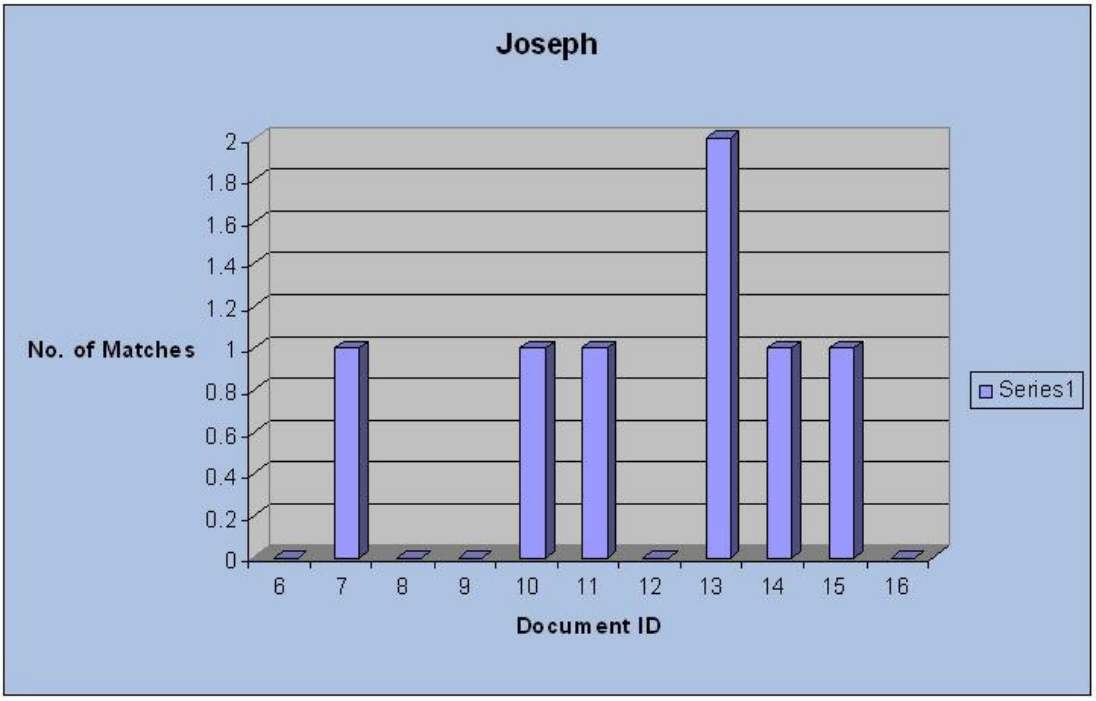
Value	Frequency	Position
microsoft	6	85, 99
monkeys	3	165, 161
bill	1	79, 90
banana	1	159, 159
suspicious	1	84, 72
ms	1	76, 100
army	1	163, 28
employee	2	35, 190
peanuts	1	192, 172
related	1	19, 123

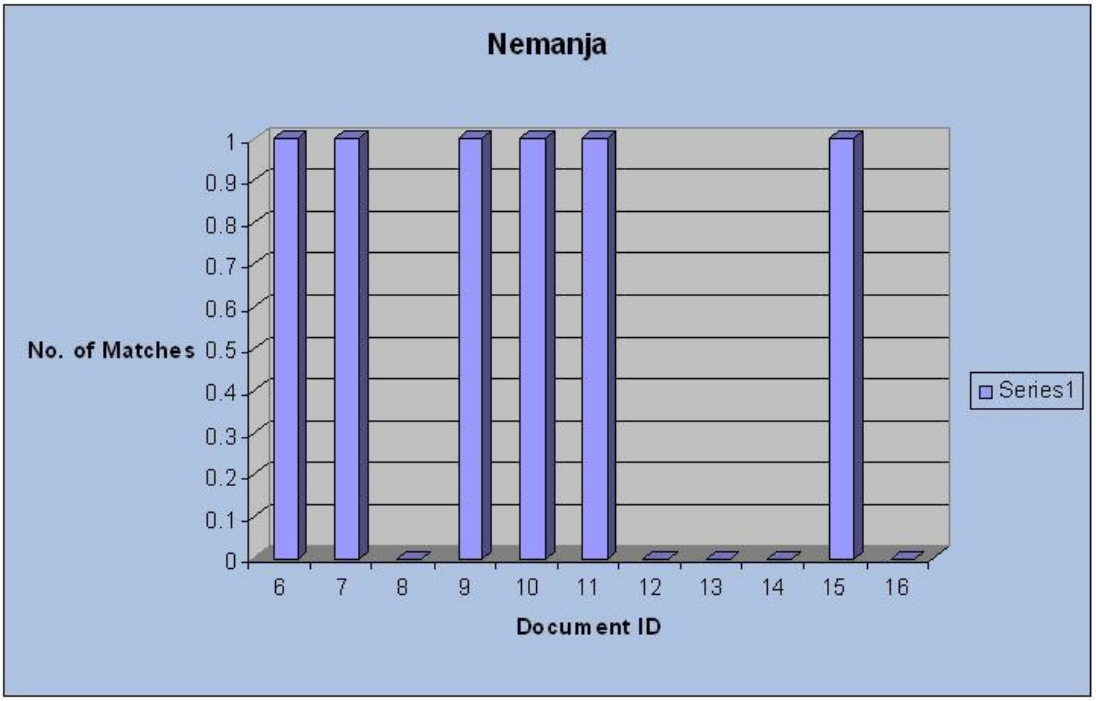
Appendix D

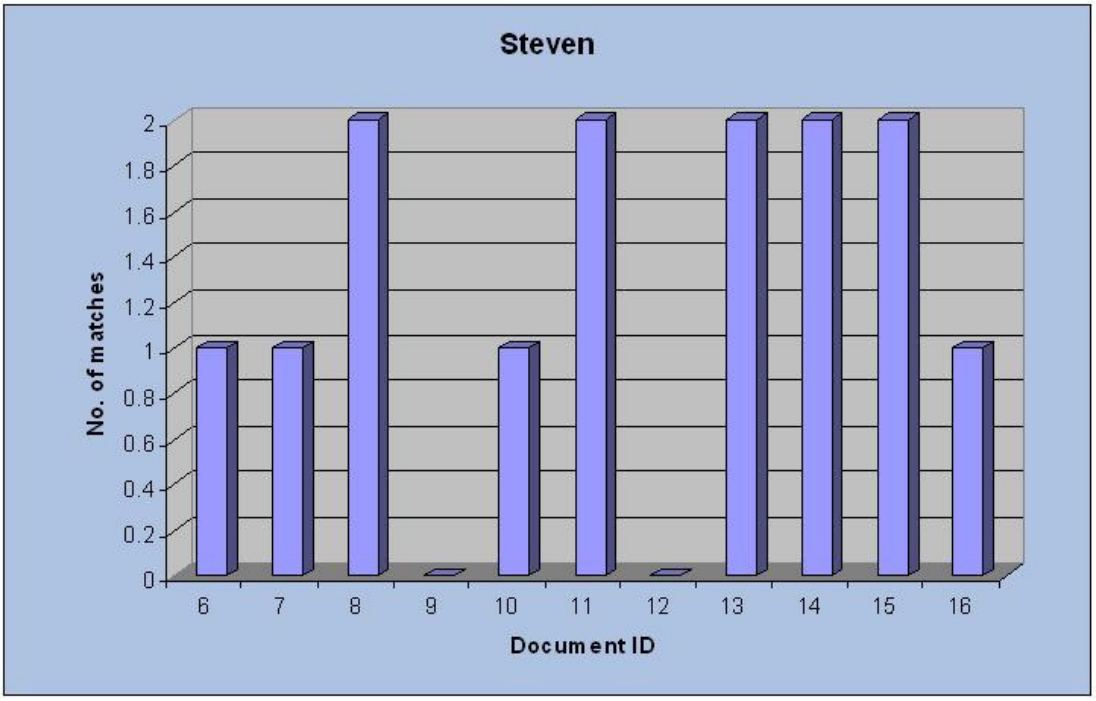
Graphs for MSc student results






Joseph
No. of Matches
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
6
7
8
9
10
11
12
13
14
15
16
Document ID
Series1


Nemanja
No. of Matches
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
6
7
8
9
10
11
12
13
14
15
16
Document ID
Series1


Steven
No. of matches
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
6
7
8
9
10
11
12
13
14
15
16
Document ID

Appendix E

Questionnaire regarding the term "astronaut"

1. astronauts -> shuttle

1	2	3	4	5	6	7	8	9	10

weak association strong association

2. astronauts -> space

1	2	3	4	5	6	7	8	9	10

weak association strong association

3. astronauts -> doomed

1	2	3	4	5	6	7	8	9	10

weak association strong association

4. astronauts -> Korean

1	2	3	4	5	6	7	8	9	10

weak association strong association

5. astronauts -> Texas

1	2	3	4	5	6	7	8	9	10

weak association strong association

6. astronauts -> optimistic

1	2	3	4	5	6	7	8	9	10

weak association strong association

7. astronauts -> discovery

1	2	3	4	5	6	7	8	9	10

weak association strong association

8. astronauts -> safe

1	2	3	4	5	6	7	8	9	10

weak association strong association

9. astronauts -> foam

1	2	3	4	5	6	7	8	9	10

weak association strong association

10. astronauts -> trouble

1	2	3	4	5	6	7	8	9	10

weak association strong association

Questionnaire regarding the word "tea"

11. tea -> diseases

1	2	3	4	5	6	7	8	9	10

weak association strong association

12. tea -> cancer

1	2	3	4	5	6	7	8	9	10

weak association strong association

13. tea -> cups

1	2	3	4	5	6	7	8	9	10

weak association strong association

14. tea -> sodomy

1	2	3	4	5	6	7	8	9	10

weak association strong association

15. tea -> britannia

1	2	3	4	5	6	7	8	9	10

weak association strong association

16. tea -> sarcasm

1	2	3	4	5	6	7	8	9	10

weak association strong association

17. tea -> eccentricity

1	2	3	4	5	6	7	8	9	10

weak association strong association

18. tea -> existence

1	2	3	4	5	6	7	8	9	10

weak association strong association

19. tea -> Sunday

1	2	3	4	5	6	7	8	9	10

weak association strong association

20. tea -> nutrient

1	2	3	4	5	6	7	8	9	10

weak association strong association

21. tea -> lifestyle

1	2	3	4	5	6	7	8	9	10

weak association strong association

22. tea -> futility

1	2	3	4	5	6	7	8	9	10

weak association strong association

Questionnaire regarding the word "gas"

23. gas -> petroleum

1	2	3	4	5	6	7	8	9	10

weak association strong association

24. gas -> fuel

1	2	3	4	5	6	7	8	9	10

weak association strong association

25. gas -> gallon

1	2	3	4	5	6	7	8	9	10

weak association strong association

26. gas -> gasoline

1	2	3	4	5	6	7	8	9	10

weak association strong association

27. gas -> free

1	2	3	4	5	6	7	8	9	10

weak association strong association

28. gas -> suvs

1	2	3	4	5	6	7	8	9	10

weak association strong association

29. gas -> drivers

1	2	3	4	5	6	7	8	9	10

weak association strong association

30. gas -> price

1	2	3	4	5	6	7	8	9	10

weak association strong association

31. gas -> car

1	2	3	4	5	6	7	8	9	10

weak association strong association

32. gas -> pinball

1	2	3	4	5	6	7	8	9	10

weak association strong association

33. gas -> severe

1	2	3	4	5	6	7	8	9	10

weak association strong association

34. gas -> celebrated

1	2	3	4	5	6	7	8	9	10

weak association strong association