Probability Latent Semantic Analysis and Internet Content Filter from Web Crawled Data

by

Kennedy Chengeta

Submitted in partial fulfillment of the requirements for the degree
Masters of Advanced Software Engineering
in the Faculty of Engineering, and Computer Science
"Department of Computer Science"
"CO7501 Individual Project"
"Student Number 139021990 email kc181@le.ac.uk,kennedychengeta@gmail.com"
University of Leicester, Leicester England

July 2014

DECLARATION

All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people's work have been specically acknowledged by clear citation of the source, specifying author, work, date and page(s). Any part of my own written work, or software coding, which is substantially based upon other people's work, is duly accompanied by clear citation of the source, specifying author, work, date and page(s). I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: Kennedy Chengeta

Signed:

Date: 15/01/2015

University of Leicester

Acknowledgements

- I thank my supervisors Professor A. Kurz and Dr Yi Hong for all the guidance.
- I also thank my wife Nyaradzai for the support and help in proof reading this document.
- To my daughter, Sheridan Miriro, i say thank you Lord for such a precious gift, welcome to the world my little one.
- This is also for all the Chengeta family past and present, namely Godfrey, Selina, Mai Tatenda, Mr Chengeta ,my mum Mai Godhie, Tendai, Charles, Simbarashe, Tatenda, Bizmark-Tadiwa, Mai Anesu, Mai Manu and cousins and nephews Emmanuel, Anesu, Rumbidzai, Tinotenda and Taku.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The explosion of web as an information source recently has brought more interesting challenges to document annotation and classification of web documents as well as to the information and collaboration filtering world. Not only does it introduce performance issues due to the huge number of documents, there is even big challenges in that most of the data are unlabeled. The arrival of social media and blogging has increased web data as well. Thus the machine learning community has taken up the interest in using unsupervised learning in classification of such big data. A big group of new learning also termed semi-supervised learning relies on assumptions that unlabeled data can help bring interesting patterns, which the industry can use. One such algorithm making big strides is the Probability Latent Semantic Analysis(PLSA) algorithm which has its roots in statistics.

1.2 Background

Latent Semantic Analysis (LSA) is often used for the classification or retrieval of information. Probabilistic Latent Semantic Analysis (PLSA) has been shown empirically for a number of tasks to be an improvement to LSA[15][25]. PLSA is a statistical latent class (or aspect) model that provides excellent results in several information retrieval related tasks. PLSA has the positive feature of assigning probability distributions over latent classes to both documents and terms[15][25]. PLSA is a topic modeling tool given a dataset and will deduce hidden topics across the documents by looking at the terms and documents and inferring using the Expectation Maximization algorithm, which words belong to, which topic and which topics are covered by what topic based on a probability foundation. The model also allows inferring if given documents infer one or more topics[15].

The PLSA algorithm is an extension of the more algebra based automatic document indexer called Latent Semantic Analysis (LSA). LSA has been researched and implemented with great success for years giving positive

results in getting synonyms, spelling differences as well identifying abbreviations as well[15]. However in 1999, T. Hofmann published the PLSA algorithm as an improvement to the original LSA algorithm but with foundations from the statistical and probability world[15]. The PLSA algorithm's foundation is on the use of statistical inference using a maximum log likelihood estimation function and the Expectation Maximization algorithm and has been proven to be a big improvement to the former algorithm. Widespread use of the algorithm has resulted with wide success reported in document indexing for big industries with large databases of information in the financial industry and beyond[15].

PLSA forms its base on the Expectation Maximization algorithm, which is a good unlabeled data classifier[15]. The process will involve giving the unlabeled data initial randomly parameters, relabelling the data using estimated values and then updating the estimated values in iterative fashion using new labels until convergence is achieved. The PLSA algorithm will assign probability distributions over latent classes in both documents and terms. The model allows both terms and documents to have or belong to more than one topic or latent class. The initial parameters (P(w—z) and P(z—d) are initialized randomly in this research but a uniform initialization could also have been used as well. but it requires some prior knowledge to use the latter method[15]. The key threshold values for the algorithm include change in the log likelihood or the number of topics denoted by k as well as putting a limit on the number of iterations if necessary. The Expectation and Maximization steps of the EM algorithm are then in exchange fashion run iteratively until convergence is achieved[15].

1.3 Motivation and Relevance of the Research

The amount of information found in the media and internet is rapidly increasing thanks to social networks and mobile platforms thus it is imperative to device new ways of filtering information using Information Retrieval techniques (IR). To add to this users need more personalized information from the web and it is also imperative that in addition to information search the web also recommends what the user sees and searches based on background knowledge of the users needs and preferences. This led to the rise of recommender algorithms and systems. This research will seek to understand the major issues around filtering of information with focus on both on content-based and collaborative filtering and how this information can be classified to give meaning to a lot of business answers as well as produce interesting patterns. All this rise in unstructured data on the Internet exacerbated by blogging and chatting programs mean data search on the internet presents more challenges than before for the field of Information Retrieval and those involved in text mining.

Information Retrieval Systems uses software to allow users to get information for user needs. They consists of a software program which try to reduce the overhead of users searching for information they require. The research will do an in-depth review of the Probabilistic Latent Semantic Analysis and compare it to other concepts applied in the information retrieval space. The probabilistic and statistical nature of the algorithm is explored with focus on the aspect model as well as the latent variable model. Other factors to be considered on the algorithm include latent space models. The Expectation Maximization algorithm which forms the basis of PLSA is reviewed

and its implementation and applicability analyzed. The algorithms application in areas like indexing, information retrieval[15], clustering and in areas like computer vision, multi-modal image retrieval and object recognition as well as audio processing is explored. The research seeks to mine data on the web which is mostly sets of news articles be it medical, sports, politics or other areas of news. This will involve using crawling techniques to mine the data and then the text data will form the dataset for this research. This will involve writing a program to download pages from the Internet using a program called a crawler. Challenges in mining web data will be explained and also security issues in getting Internet data will also be discussed. Once the data has been downloaded, then the desire is to seek to identify latent or hidden topics in the downloaded data then identifying which documents forming part of which topic as well as which terms form part of which topic.

As a result one should be able to given a certain name classify the name against a field or category be it sports or professional field. The research will compute the accuracy of such implementation given when meaningless words are included and not included as well. The research will involve two stages, firstly the data mining of the web data off the internet, and the second part being the classification of the data. Probabilistic Latent Semantic Analysis (PLSA) is a successful statistical latent class model which cannot assign likelihood to unseen documents. It is also a statistical latent class model that has recently received considerable attention. PLSA is widely used in Natural Language Processing to analyse co-occurrences of terms in corpus of documents so that hidden factors or concepts can be found. Data is in the form of a document-term matrix.

The mined data will be split into two categories and namely the training and the testing data Using this data the classification aims to derive given a persons name which category that person falls into, for instance given a name called Lewis Hamilton from the classification there should be a derivation that Lewis Hamilton is a Formula 1 car racing driver and also Usain Bolt is an athlete or Tom Cruise is a movie actor. The tool will only count meaningful words and forget meaningless words, which will be removed during the pre-processing stage once the web data, have been mined.

So in this research Tom Cruise or Lionel Messi the footballer will be the concept. The Expectation Maximization algorithm is usually used to implement the Probability Latent Semantic Analysis and will form the basis of the classification implementation. The algorithm, which is of an iterative nature, will compute the maximum likelihood estimate given data with missing or hidden instances. This Machine Learning algorithm will estimate model parameters for which observed data is most likely to take and this will happen in two steps namely the E-Step where the missing data is estimated from the observed data. This is sometimes called the Estimation Step. In the M-step, the likelihood function is maximized based on the assumption that the missing data is known with the estimate of the missing data from the E-Step used in the place of the actual missing data. The web crawling not only faces security issues and network timeout issues on downloading the required data but also excludes pictures and graphic images hence only text data is only analyzed. The text data being analyses is only limited to English hence other languages and non-English Web sites are not considered. Some of the challenges also involve distributed data over millions of websites and most of it being very volatile and of huge volumes. The data to be analyzed is

mostly unstructured as well and heterogeneous including images, videos and different character sets and languages as well. The crawler in the form of spiders should not fall into spider traps where it downloads infinite number of pages and also show politeness on websites, which set limits to visiting frequencies. Link Extraction will involve finding all the links on the page and visiting those links to extract the URLs as well. Python and java are used as the programs of choice in extracting the data and also removing the commonly used words, which have no bearing for the text classification. The classification uses the Expectation Maximization (EM) algorithm. The EM Algorithm is easy to implement and each iteration will improve the accuracy of the estimated values. As the estimation reaches the optimal values the rate of convergence becomes slow. The algorithm is also known to work better when the data that is missing is in slow quantities but not in large quantities. Large dimensionality data sources can also prove a challenge for this algorithm and this slows down the Estimation step of the algorithm.

1.4 Objectives and Milestones

There project aims to firstly understand the Probability Latent Semantic Analysis algorithm and implement it. This is achieved by using the Expectation Maximization Algorithm. Secondly it seeks to implement this algorithm as information filtering software tool by identifying key concepts in web data using unlabeled and unclassified Internet data. It also aims to retrieve large number of documents for classification from the web using automated methods like crawling. The first objective of this research will be looking at various information retrieval methods applicable especially in text classification and comparing these retrieval techniques as well. Various previous work done in the information retrieval space will be discussed. The research shall also do a literature review on the concept of Probability Latent Semantic Analysis and Expectation Maximization algorithm and their applicability in the field of information retrieval. A comparison against other retrieval techniques will also be form part of the literature review. To be able to do the classification of the data there is need to retrieve the data on the web using crawling methods which is then used to do text classification.

A major milestone will be to be able to use a crawling algorithm to retrieve large amounts of data of the Internet and saving the data in a local repository. It is imperative that before this data is classified is pre-processed hence it is crucial that it goes through pre-processing and then classification. The aim of this research is to use unclassified web data on the Internet to filter information and classify the data using a machine-learning algorithm like the Expectation Maximization Algorithm. One of the desired of the objectives will be to allow users to deduce the category or field a concept belongs based on the classification results done. Hence given huge amounts of unclassified web data a concept can be classified into a specific field with for instance a popular figure like Tiger Woods being classified as a golfer based on the web data classified and also his prevalence and in the articles classified as golf articles. Classification done using the Expectation Algorithm will involve training a huge dataset of web data and then testing against another dataset. The other requirement will be to check accuracy of data if using classifier and not by comparing the 2 result sets.

Milestones of the project will involve the following

1.4.1 Milestone 1

1. Pre-processing the data
2. Saving and index the downloaded web documents.
3. Analysis of retrieval algorithms, Probability Latent Semantic Analysis and Expectation Maximization concepts.

1.4.2 Milestone 2

1. Use of the EM algorithm to classify the data. This involves training and testing the data.
2. Analysis of the results and comparing accuracy of the classification of the web data gave various scenarios where spot words are included and excluded.

1.4.3 Objectives

The objectives and questions of this research are summarized below:

1. Can we use PLSA algorithm to classify web documents and identify hidden topics?
2. Can web data be successfully crawled and preprocessed into data which can be analyzed by machine learning?
3. What are challenges in web crawling?
4. Is PLSA algorithm an efficient algorithm?
5. PLSA algorithm works well on processed data than on raw web data?
6. What factors and challenges affect PLSA algorithm in topic identification?
7. Does number of documents, value of topics or iterations matter?

To achieve the objectives, the research crawled web data from different websites and preprocessed it. The processed data and preprocessed data were then classified using a text mining algorithm, the PLSA algorithm, and comparisons were done to see which factors affect this algorithm. Analysis of the results and comparing accuracy of the classification of the web data given various scenarios where spot words are included and excluded was also a key objective of this research.

Table 1.1: Thesis Timetable

Chapter 1	Introduction
Chapter 2	Crawling Literature Review
Chapter 3	PLSA Algorithm Literature Review
Chapter 4	Research Design
Chapter 5	Crawling Implementation
Chapter 6	PLSA Algorithm Implementation
Chapter 7	Analysis and Conclusions

1.5 Conclusion

The research was done with the aim to classify internet web data using the PLSA algorithm, which is a statistical based model based on the Expectation Maximization algorithm. The other aim was to do parameter estimation using maximum likelihood estimation. The data for classifying was obtained through crawling the web and a crawler implemented in java was used in this research. We start this thesis with related work in chapter 2 to give a background in the supporting research fields in the field of crawling.

Document collection involved using a java web crawler to collect documents restricted to mostly .html, .jsp, .asp formats and then saving the documents in a local data repository. Preprocessing of the data where the crawler removed stop words like or, my and other followed this. A document term matrix was then generated which was then used in the classification and training of the documents using the Expectation Maximization algorithm. The algorithm involved repeating the Expectation and Maximization steps iteratively until convergence was achieved.

Following the related work, the algorithm for Probability Latent Semantic Analysis is introduced and explained in chapter 3. In chapter 4 the limitation the research are summarized, and Chapter 5 details the implementation of the crawling algorithm as well as the data preprocessing. The implementation details and evaluation of the PLSA algorithm are discussed in Chapter 6. Chapter 7 discusses changes proposed and implemented in this thesis, and also concludes this thesis by finding the answers to the research questions.

Chapter 2

Web Crawling

2.1 Introduction

The chapter does an analysis on the history of web crawling. It looks at how web crawling is done and its use in data mining. Since this research is focused on mining large samples of web data, the research also looks at past researches that optimised web data searching. The research uses web crawling to gather web data for analysis using the PLSA algorithm. Web crawlers or spidering discover all publicly available pages and follow the links. They move from link to link and download all the web data for either saving the data in their own servers for faster searches latter by creating local copies of the data. Crawlers are also used to do automatic maintenance like html code validation or link validation as well. Changes to existing sites , dead links as well as new sites are validated and identified. Unauthorized harvesters will also include crawlers that harvest email addresses for spam purposes latter on which is a big risk.

Olston and Norjork [33] outline some of the applications of web crawling as downloading and analysing web pages for statistical reasons for instance Attributor web side. Attributor checks all websites for copyright infringements. They also point out that large data sets are downloaded for posterity reasons and archived. The crawlers also enable submitting standing queries like GigaAlert where the crawlers will continuously crawl for information and notify clients of matching as and when they come through in real time. The crawler should show politeness by respecting the web server rules for each web site they visit which govern how many times and at what frequency they are allowed to visit these websites otherwise they will be blacklisted. Downloading many pages simultaneously from one website may overwhelm the server. Various crawlers are used in the modern world with GoogleBot, PolyBot, Spybot, Web Crawler and BingBot being some of the popular ones from Google for the former[33][10, 34]. Open-source crawlers include GNU Wget, GRUB, Apache Nutch and PHP-Crawler[10, 34][33].

2.2 Web Crawling

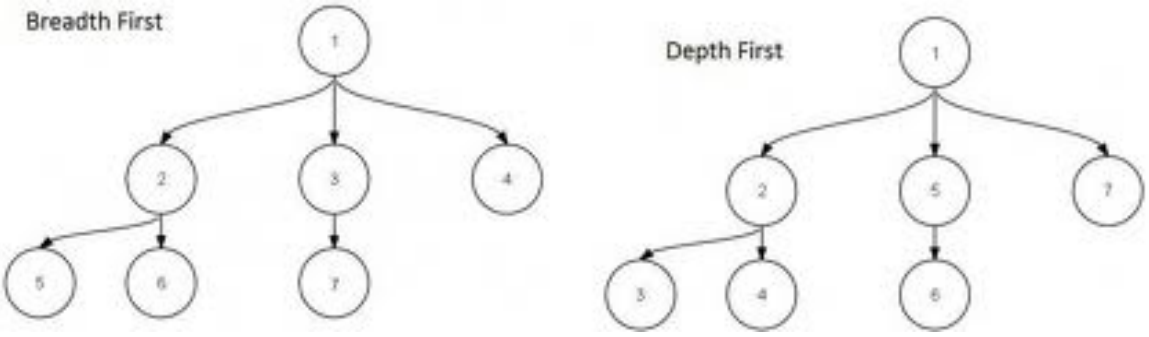
Web Crawling is the process of search engines combing through web pages in order to properly index them and began in the early 1990s and Google is the well known one [34][32]. These web crawlers systematically crawl pages and look at the keywords contained on the page, the kind of content, all the links on the page, and then returns that information to the search engines server for indexing [34]. Then they follow all the hyperlinks on the website to get to other websites. When a search engine user enters a query, the search engine will go to its index and return the most relevant search results based on the keywords in the search term. Web crawling is an automated process and provides quick, up to date data[32]. A web crawler is a software, which begins from a set of seed URL(Universal Resource Locator) downloading all the pages associated to these URLs and the downloaded URL is immediately removed the crawling queue. The downloaded page is then parsed and additional links are extracted, called the crawl frontiers. This is done in an iterative manner until all the contents have been reached down from the seed URL[32, 34][10].

Due to the emergency of rich Internet applications and shift from server side web pages to auto generated client pages as a result of HTML5 and other Web 3 technologies, Deep crawling also emerged[32, 34]. The problem on these kind of crawlers is that most of the client side data was hidden[32, 34]. DeepCrawl is a web crawler, which allows self control and can crawl websites, staging environment, external sites, backlinks and sitemaps, with a host of flexible crawl types. [32, 34].

2.2.1 Crawling Strategies

Crawling is done using either Breadth First or Depth First Strategies. In graph theory, the web is treated as a big directed graph with documents being treated as vertices and hyperlinks as edges[10]. Thus there are a traversal algorithms that are used to traverse web data and Breadth First and Depth First strategies feature prominently[10]. The depth first search (DFS) algorithm, result in getting the initial unvisited link from the start page, visiting the link and getting the first non-visited link. This is repeated until all non-visited links have been crawled[10]. Only then are non-visited pages also done and the process is repeated once again. The Depth First is summarised in the following section.

Figure 2.1: Crawling Strategies Tab[28]



2.2.2 Depth-First algorithm

Data: 1st link not visited from the start page
Result: Web pages downloaded from the web
initialization ;
Step 1: Get the 1st link not visited from the start page
Step 2: Prepare to keep a list of visited nodes (initially empty) VisitedNodeQueue;
while *no visited links* **do**
| Step a: Visit link and get 1st non-visited links Step b: Go to next non-visited link in the previous level
end
Step 5: Print all visited links

Algorithm 1: Depth-First algorithm[10]

While depth-first goes off into one branch until it reaches a leaf node, Breadth First always takes a look at the alternative paths. Therefore the latter strategy is deemed a more complete and optimal strategy although it brings a huge perfomance cost to the implementation due to high memory and CPU demands. Whislt Depth First uses less space space, it keeps track of all the nodes it has visited though they are less than those of the Breadth First algorithm. The follow points summarise the Breadth First algorithm

2.2.3 Breadth First Algorithm

Data: Queue QW with seed Urls chosen
Result: Web pages downloaded from the web
initialization ;
Step 1: Put all the given seeds into the queue QW;
Step 2: Assign queueu NodeQueue to empty. Step 2: Prepare to keep a list of visited nodes in the queue Node-Queue.;
while *QW is not empty* **do**
| Step a: First Remove the first node from the given queue QW;
| Step b: Then the node is appended to the visited nodes list, NodeQueue
| **foreach** ***each*** *edge in node* **do**
| Step c1: If the node at the end of the edge already is already on the visited nodes list or has been added to the queue already, then no more action to that edge;
| Step c2: If not then the node is appended at the end of the edge to the end of the queue QW.
end
Step 5: Print all visited links

Algorithm 2: Breadth First algorithm[10, 23]

2.2.4 Crawling Algorithm used in this research

The generalised crawling algorithm is shown below

2.2.5 Crawling Algorithm

Data: Queue QW with seed Urls chosen
Result: Web pages downloaded from the web
initialization ;
Step 1: Initialize queue QW with initial Urls chosen and known.
while *QW is not empty* **do**
step a : Pop URL a URL1 from front of QW.
If URL1 is not valid(picture, multimedia, social media, etc.) ;
then exit loop.
step b: If URL1 has been visited continue and get follow-up urn;
Download page X from given URL ,URL1
if download fails(Network Error, Robot etc.)
exit loop, else.
step c: add page X contents to MySQL Database after Pre-processing
step d: Fetch new Urls(Parse X and get new list of links of K)
step e: Append K list of new links to the end of QW
end
Step 5: Save all the pages down crawled and downloaded

Algorithm 3: Crawling Algorithm[10, 23]

2.2.6 Incremental Crawling

Cho and Garcia-Molina[8] outline the evolutions and impacts on the web of doing incremental crawling. Through incremental crawling , the crawler visits the web pages periodically to see any if any changes and compares to the previous pages[8]. Some pages were also found to have been deleted as well. In their research findings outlined in the article "The Evolution of the Web and Implications for an Incremental Crawler", they concluded that as the web data grows exponentially like today, it is imperative to build an incremental crawler that selectively chooses pages to crawl and store and also redefines its searches based on quality and considering fresh data[8]. Some of the factors they considered include how often web pages change, web page lifespan and if the web pages can be described using mathematical model.

2.3 Efficient Crawling

Some of the features that enable efficient crawling include focused crawling, URL normalisation and URL and crawling policies[23][10]. Internally, the crawler must deal with huge volumes of data[10]. Unless it has unlimited computing resources and unlimited time, it must carefully decide what URLs to scan and in what order. The crawler must also decide how frequently to revisit pages it has already seen, in order to keep its client informed of changes on the Web. Crawling policies discussed in the following section are one of the most important factors in efficient web crawling[10][23].

2.3.1 Crawling policies and ROBOTS

URLs from the frontier are recursively visited according to a set of policies[23]. This also shapes the behaviour of the web crawler as policies or a combination of policies are used to define how the web pages are visited. Selection policies determine how web pages are downloaded[23]. Re-visit policies determine what states crawlers need to check when web pages change[23][10]. Politeness policies govern how web pages can avoid belong overloaded with requests[23]. Parallelization policies state govern how web crawlers can be distributed and coordinated for faster crawling. Some of the selection policies touch on what restrictions are in place by web pages on some of the links that need to be followed. Selection policies also include governance on path ascending crawling, focused crawling as well as deep web crawling which includes server and client side web page crawling[23].

Due to web being so dynamic, revisit policies govern the freshness and age of web pages and how often crawler can check the page for changes. Krenke[23][14] recommends high average freshness and low web page age averages. Cho and Garcia-Molina[8] studied two simple re-visiting policies They concluded that there exist either a uniform policy was found to involve revisiting all pages on same frequency of change and proportional policy involved revisiting pages that changed more frequently[8].

Politeness policies were introduced to avoid page overload. Krenke further suggests that crawlers can cause huge performance problems on web sites due to network resources usage, web server overload, server or network router downtimes as well slow network response times[23]. To mitigate this, she suggests the use of robots exclusion protocol. As part of web pages design the web sites either specify a Robots Exclusion Protocol (robots.txt), which shows, excluded web links which any crawler should bot visit[23]. She also suggests using a ROBOTs META tag, which tags specific documents which should not be indexed nor have its links followed. Most web servers use a robots.txt show what a crawler can do and or not do as well. The Website administrator puts a robots.txt file at the root of the hosts web directory. Some specific robots or crawlers can be allowed as well[23][14] . The following shows a sample robots.txt on one of the websites the research looked at namely www.redhotcleaning.co.za. The ”User-agent: *” says that this applies to all crawlers and robots. ”The Disallow:” followed by the directory shows a list of all the website the crawler will not visit as a result[14] . The robot can exclude a specific robot for instance GoogleBot not to crawl anything at all and they are placed in the top level to be of use e.g. http://www.ken.com/robots.txt.

```
#Exclude a specific robot:
User-agent: GoogleBot
Disallow: /
User-agent: *
#Exclude specific directories:
Disallow: /tmp/
Disallow: /~backups/
Disallow: /home/kennedy/pictures/
```

```
Disallow: /~mypages/soccer.html
#Allow total access to specific robot:
User-agent: *
Disallow:
```

Cho and Garcia-Molina[8] use 10 seconds as an interval for accessing web pages in the crawler. If it took t seconds to download a document from a given server, the crawler waits for 10t seconds before downloading the next page[8]. A more detailed cost-benefit analysis is needed and ethical considerations should be taken into account when deciding where to crawl and how fast to crawl.

The parallelization policy for the web was created to govern parallel crawling processes. The goal was to increase the download rate, reduce parallelization overhead effects and reduce the chance of redownloading already downloaded webpages[23]. A policy is given to each URL discovered in the crawling process.A policy is defined for assigning each new URL discovered in the crawling process. The motivation would be to download as many web pages from a given website through ascending. That way a crawler would ascend to every path in each URL that it intends to crawl. In a simple case, when given a seed URL of http://cricinfo.org/c/m/mycricket.jsp, the crawler will attemp to crawl /c/m and /c/[23].

2.3.2 Focused crawling

In industrial cases and other areas of interest focused crawling is recommended. Focused crawling allows the given crawler to download only pages that satisfy a special interest or topic it chooses[23]. The crawler uses probability that a given page is relevant to its chosen focus or topic before it starts downloading the page. One of the tools used to do the prediction is the anchor text of links. In some cases, the download of the page would be done and then after analysing the contents the web page is then deemed irrelevant or not. The relevant pages are then stored and indexed and only their urls would be added to the frontier for further crawling. All the web pages deemed to be less than a certain threshold of relevance are not considered further[23].

2.3.3 URL normalization and different web data formats

Crawlers use URL normalisation to reduce chances of crawling same web pages again. This involves modifying URLs in a consistent and standard way across the spectrum by converting the url to lowercase, removing '.', '..' as well adding other trailing slashes to the URL[5]. Crawlers usually perform some type of URL normalization[5, 21] in order to avoid crawling the same resource more than once. This is also called URL canonicalizing. Crawlers also face huge challenges in getting web data in different formats be it RDF or Resource Description Framework, XML or Extensible Mark up Language, as well as HTML. In cases where continous crawling is enabled, then new data formats always come and it is imperative that such web crawlers use canonicalizing in order to identify duplicates and near duplicates. The following java code will output "http://www.redhotcleaning.co.za" after normalisation. Canonicalisation also includes decoding encoded characters and converting hostnames into lowercase form. Java

API methods like ”Reference.normalize()” from the Restlet REST(Representational State Transfer) API Framework will even also strip a URL like ”http://kennedy.com/./my%26name” into ”http://kennedy.com/my&name”[5, 21].

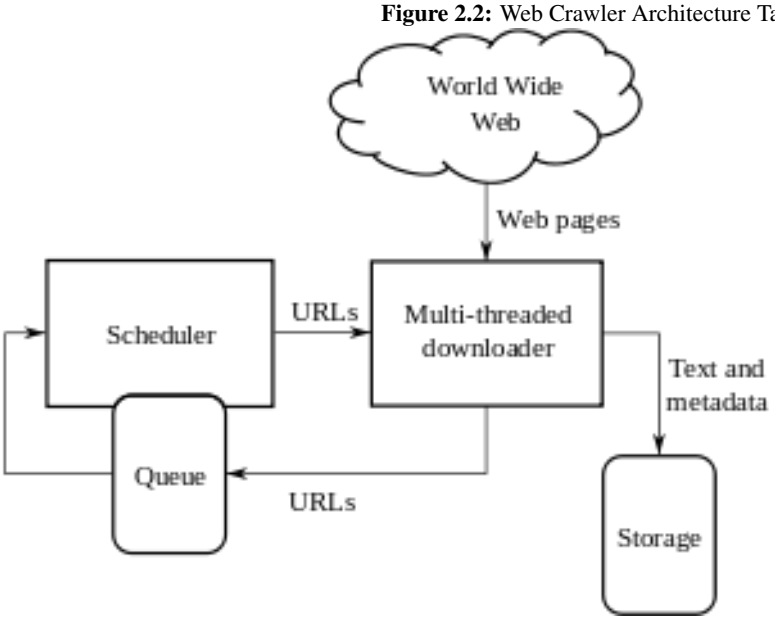
```
String redhotcleaninghome = "http://www.redhotcleaning.co.za/about&executives";

try {
    URL url = new URL(redhotcleaninghome);
    System.out.println(url.getProtocol() + "://" + url.getHost());
} catch (MalformedURLException e) {
    e.printStackTrace();
}
```

2.3.4 Page Dynamics and Changes

Changes to website through domain expiry, dead links and network errors on some of the pages is a challenge to web crawling as most of the pages return error messages during crawling. Some of the errors include a 404 ”Not Found” Links which indicates a dead link on the website and that The server has not found anything matching the Request-URI. Links generally direct to a 404 error because a page has been moved or deleted without being properly redirected. Most successful crawling http calls will get Status ’200 OK The request has succeeded’ message. Some web pages return error message to do with authorisations for instance Status 403 with the message, ’Unauthorized The request requires user authentication.’ A 403 Forbidden shows that the server understood the request, but is refusing to fulfil it. A 407 Proxy Authentication Required shows Unauthorized error but indicates that the client must first authenticate itself with the proxy. Timeouts return a Status Message 408 Request Timeout.

2.4 Web Crawler Architecture

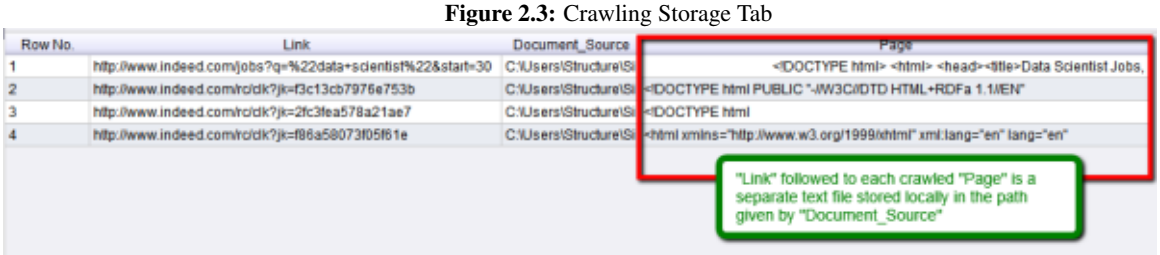
Figure 2.2: Web Crawler Architecture Tab[28]



2.5 Storage of Crawling Data for Data mining purposes

Crawled data to be used for text mining is normally either stored in flat files or in relational databases. Popular databases include MySQL, SQL Server from Microsoft or DB2 and Informix from IBM as well as Oracle Database[33]. The cloud also offers larger corpus with raw web page data and text extracts like the Common Crawl data stored on the Amazon Web Services cloud platform. This research used MySQL for data storage. To also ensure all the web page data is stored different options used include storing the web page html data as binary large objects(blobs).

Figure 2.3: Crawling Storage Tab



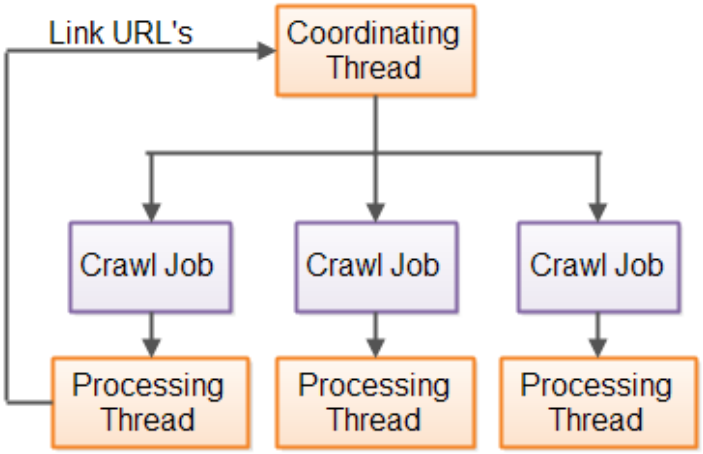
2.6 Web Crawling using java

Popular java crawlers include Sprindle, Spider, Jsoup, Crawler4J and WEKA tool from New Zealand Waikato University. Nutch based on Lucene as well as WebSPHINX are other major java web crawlers on the open source world. Web-Harvest is a java implementation that collects Web pages and extracts useful data from them leveraging on text/xml manipulation like such as XSLT, XQuery as well as Regular Expressions. Java web crawlers will parse the root web page, and get all resultant links from this page. The algorithm then accesses each URL and parse HTML page. From the URLs that was retrieved in the previous step, the algorithm then parses those URLs. The algorithm then tracks which page has been processed before so that they are processed only once. Thus the crawler uses a database to store all the previously stored websites.

2.6.1 Java Multithreading

Web crawling processing time has seen improved times where java multithreading is applied. In Java multithreading is applied using threads by implementing the Runnable Interface as well as the run java method to spawn multi-threads. Various java crawling multithreading options include a Single threaded and synchronous crawler, a multithreaded concurrent crawler, a single threaded and network- input output(nio) based crawler as well as a multithreaded nio based crawler[18]. It is simpler to implement a multithreaded input output based design, in which each thread is only downloading and parsing a single page at a time. That will both utilize the bandwidth and CPU of the crawling computer better than the other multithreading options. The worker threads are located in a thread pool. The crawl job object implements either the Runnable or Callable interface, depending on the concrete design of the crawler. The following is a diagram illustrating a multithreaded Java web crawler:

Figure 2.4: Web Crawler Multithreading sourced from [18]



2.6.2 Crawling Ajax

With the emergency of Asynchronous Java and XML technology or Ajax urls no longer have unique URL representations for a unique state. Thus as crawlers and spybots access the URL, will need a fully preprogrammed javascript engine to access a full content of the ajax based web pages. Mesbah et. al. in the article Inferring User Interface State Changes [31]suggest a crawler based on a state machine where each action executed on the browser from the root down is stored. The states are discovered by searching a DOM or Document Object Model tree for possible events that can be actioned or triggered. Some of the actionable events include the onClick, onMouseOver or onMouseOut and onButtonClick events[31].

2.7 Crawling as an Information Retrieval tool for Data mining

Information retrieval (IR) is a field where documents both structured in text form are used to satisfy information needs from large document collections[29]. Crawling is done for various reasons that include searching for mirroring websites, copyright infringements, monitoring sites for content and structural change, testing websites for syntax and structure validity as well as for text mining or data mining[33][10]. A text mining research will have three stages namely gathering the data in unstructure form, converting it into meaningful pattern and analysing it using datamining[10].The research uses crawling to retrieve and gather thee large document collection which is then used to do the information retrieval. The documents are then parsed to filter irrelevant words. The research is more focused on text mining, which is more focused on document classification, clustering, ontology construction and sentimental analysis[29].

2.8 Pre-processing of Crawling Data for Text mining purposes

Pre-processing involves concepts like token normalisation, filtering stop words, and excluding picture files as well as storing the crawled data in a database storage for text mining purposes. Token normalization involves canonicalizing tokens so that they match regardless of superficial token character sequences differences[10]. Tokenization involves chopping away apostrophes in words for instance in words like aren't or isn't. Capitalization or case folding has also been used as a pre-processing tool to make sure same words with different case being upper and lowercase are considered the same as well[10]. Manning et al.[10] also suggest considering equating British spelling and American words who have the same semantic meaning for instance the word colour in British English and the American spelling colour to be considered the same. Different accents are also considered having the same meaning in the word. Pre-processing also involves stemming and lemmatization to reduce words, which are derivationally related like democractic, democracy and democratization[29] Through stemming all the words are chopped off when necessary and an implementation has been done through Porters algorithm and the Paice/Husk stemmer[29]. Experiments and debates on the advantages and negatives of stemming is done and summarised by the works of Salton (1989), Harman (1991)as well as Hull (1996)[29].

2.9 Measuring and improving crawling performance

Crawling performance measures include Time to first byte, Base page download time, full-page download time and size of file download. Crawling is affected by the following performance metrics as well as the infrastructure resources used[29]. The number of crawler threads, politeness delay as well as the crawling depth affects performance.

2.9.1 Crawling metrics

Some of the metrics in crawling that influence performance include Backlink count, PageRank, Forward Link Count as well as location metrics. The Backlink count defines the number of links to a page P that appear on the entire Web[10]. The PageRank backlink metric defines recursively the importance of a page as a weighted sum of all the backlinks that link to it[10]. The forward link count of a page P, defines total count of all links that emanate from page P. Backlink Count[10]. The value of I(P) is the number of links to P that appear over the entire Web. The Location Metric looks at the location of the page as in the physical and virtual locations including where the web server is located. URLs ending with .com or .co.za, which are South African domain URLs might be prioritised depending on the crawling objective[10]. The other location considers URLs with fewer slashes more useful than those with more slashes[10].

2.9.2 Crawling speed

The number of crawler threads, politeness delay as well as the crawling depth[29][14] affects performance. It is ideal to reduce politeness delay hence reduce load on small websites.

2.10 Conclusion

For the purposes of this research crawling is being done to compile all the web data for Information Retrieval Text Analysis and these websites are downloaded and stored in a MySQL Database for further analysis using a chosen Information Retrieval algorithm called Probabilistic Latent Semantic Analysis (PLSA). This data is also pre-processed to remove stop words, which are not needed for the analysis as well[33]. For the purposes of this research only one connection is opened to any given host at a time. The waiting time is also set to a few seconds to avoid being blacklisted and politeness is respected as well.

Chapter 3

Information Retrieval Algorithms: Expectation Maximisation Algorithm and Probability Latent Semantic Analysis

3.1 Introduction

Previous chapter focused on literature on crawling. The current chapter looks at information retrieval and application of Probability Latent Semantic Analysis algorithm as the algorithm for text classification. Current literature on Information Retrieval is discussed as well PLSA algorithm application and use. Kowalski [24] worked more on information retrieval and Hofmann [15] did research most work on PLSA through his articles namely 'Unsupervised Learning by Probabilistic Latent Semantic Analysis' and 'Probabilistic latent semantic indexing and analysis'[24]. The chapter discusses the Expectation Maximisation algorithm as the major backbone of the PLSA algorithm as well as the use of the Maximum Likelihood function to maximise the Expectation Maximisation algorithm.

3.2 Overview of Information Retrieval Algorithms

Information Retrieval dates back forty centuries back with books being written with table of contents[24]. At the turn of 20th century information volumes sky-rocketed and need to have faster access to the stored information grew. In the late 1940s computerised information retrieval began. Information retrieval (IR) includes finding data in mostly document form and in unstructured forms like test or other forms that will fulfil information needs. The data is usually stored on the Internet or in other large document collections like databases or ontology repositories. On the web, the information is shielded from human or communication protocols like location or software type or version[24].

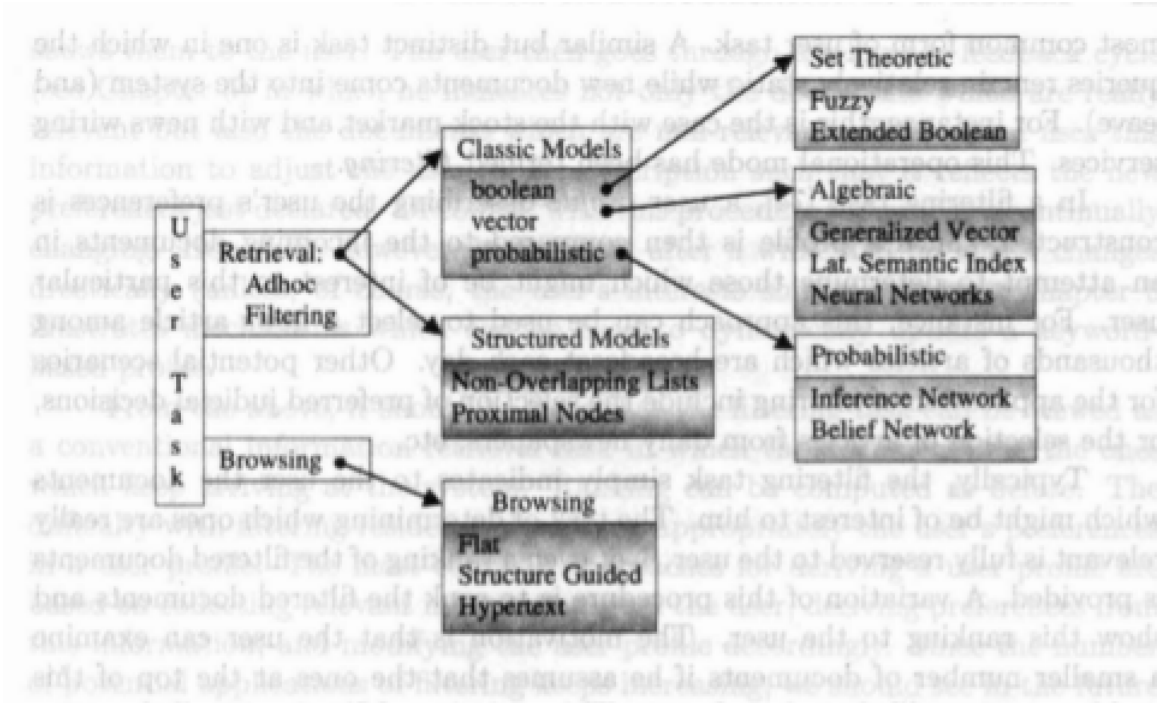
3.2.1 Text Mining

Information Retrieval has of late been expanded to include document classification, information filtering, information modelling and data visualization as well as document categorization. The field also allows for supporting users when browsing or filtering information mostly on the Internet. Text mining or text data mining is the special field in information retrieval that allows for these operations as well as carrying out analysis of the text to deduce patterns of the data statistically or otherwise. On searching some of the major searching algorithms include Extended Boolean model and the ranked retrieval[24]. This vocabulary lookup operation uses a classical data structure called the dictionary with classes of solutions including hashing or using search trees. Information Retrieval retrieves documents using various strategies like set-theory models, algebraic models and probabilistic models. Set-theory models represent documents using words or phrases like Standard or Extended Boolean models as well as the Fuzzy retrieval equivalent model[24].

The algebraic models uses vectors or matrices to represent documents represent documents and queries[24]. These include mostly Vector space model, an extended Boolean model, Topic-based Vector Space Model and latent semantic analysis. The third major model group includes Probabilistic models which treat document retrieval as an example of probabilistic inference[24]. Probabilistic theorems like the Bayes' theorem form the backbone of these models and these include models like Latent Dirichlet allocation, Binary Independence Model, Probabilistic relevance model as well as PLSA or Probabilistic Latent Semantic Analysis model[24]. These algorithms were compared by Lee et al[25][24] in their research titled AN EMPIRICAL COMPARISON OF FOUR TEXT MINING METHODS. The research will use Probabilistic latent semantic analysis(PLSA) for its analysis.

The purpose of this research is to mine text data from the web and analyse it using for text mining using information retrieval algorithms like Latent Semantic Analysis or Probability Latent Semantic Analysis. Latent Semantic Analysis uses Singular Value Decomposition and allows for capturing synonyms of words. Some of the limitations include being difficult to determine the number of topics as well as deduce neither the meaning nor labelling a topic using a words [25] [24]. The following model summaries all the information retrieval models.

Figure 3.1: Information Retrieval Models[24]



3.3 Latent Semantic Analysis (LSA)

Latent Semantic Analysis (LSA) is used in Natural Language Processing to analyse text-using matrices. It allows to find relationships between documents and terms[15][25]. It is used for document classification, clustering and text searching. The model whilst admitting that words are observable argues that topics are latent and not visible[15][25]. LSA in addition to recording document keywords, allows to see which other documents contain synonymous words and it is based on the premised that documents with many commons words are semantically close and terms occurring in similar contexts are associated. The vector space methods treat terms and documents as rows and columns and this occurrence matrix is translated into a vector space model[15]. The model uses a mathematical technique termed Singular Value Decomposition to single out patterns in relationships among terms and concepts in the unstructured text repositories. Singular value decomposition (SVD) is used to reduce the number of rows while preserving the similarity structure among columns[15][25]. Its probabilistic equivalent model, Probabilistic latent semantic analysis(PLSA) has been widely used in text mining applications like text retrieval, document clustering and categorization as well[15].

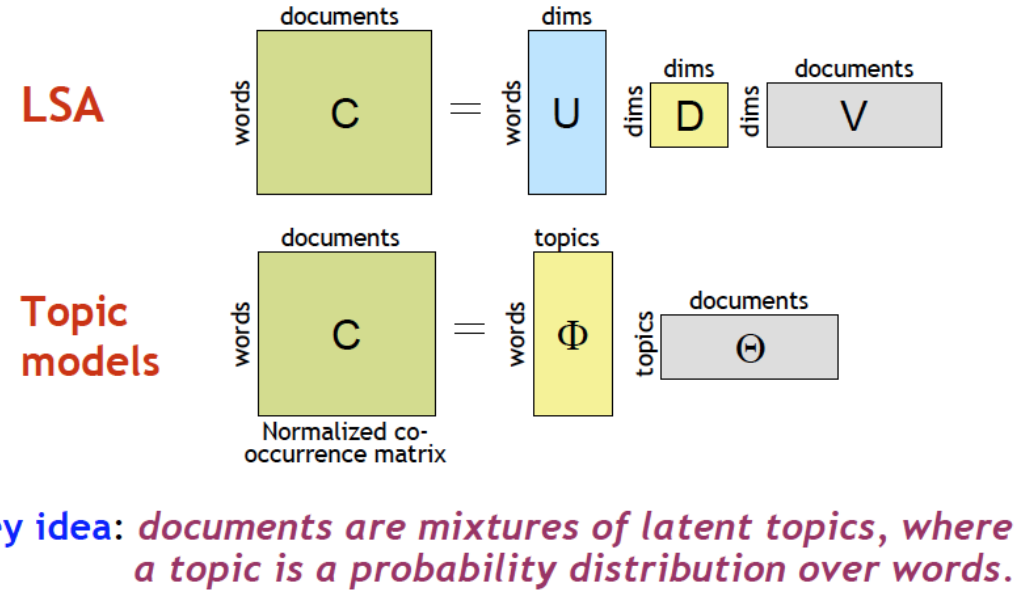
3.4 Latent Dirichlet Allocation (LDA)

Blei et al[4] proposed and applied LDA to explain several tasks with topic recognition, entity statement and spam categorization[15][25]. LDA too assumes that each document is a mixture of multiple topics, and each document can have different topics weights. LDA is a full generative model and readily generalizes to unseen documents[4] [25]. The LDA generative process is based on the premise that for a given document, sample a topic multinomial with size K from a Dirichlet distribution Dir() then for each word position, sample topic index z and sample a word from the given topic w z[4] [25][15].

3.5 Probabilistic Latent Semantic Analysis

The development of algorithms that enable computers to automatically process text and Natural language is becoming a necessity with huge cases of web data and millions of websites available[15].Probabilistic Latent Semantics Analysis (PLSA) stems from a statistical view of LSA and as opposed to the standard LSA, PLSA uses a generative data model. Probabilistic latent semantic analysis(PLSA) is an unsupervised clustering method that does automatic recognition of topics in written documents. It tries to address limitations of Latent Semantic Analysis(LSA) with its probabilistic nature[15]. PLSA is implemented using the Expectation Maximization algorithm implementation.

Figure 3.2: LSA compared To PLSA[15, 30]



Probabilistic Latent Semantic Analysis (PLSA) allows identification and separating varying contexts of words and reveals topic similarities by grouping words with common context. The model performs document categorization by automatically detecting concepts within documents via a statistical analysis of word contexts. It is

a technique based on topic models that aims to model co-occurrences of information using a probabilistic reasoning hence discovers the underlying semantic data structure. The model is based on a probability distribution between latent semantics and documents or words whereby a latent semantic space mapping documents and words is generated[16]. The paper presents perplexity results for different types of text and linguistic data collections and discusses an application in automated document indexing. The experiments indicate substantial and consistent improvements of the probabilistic method over standard Latent Semantic Analysis. PLSA associates a latent context variable with each word occurrence, which explicitly accounts for polysemy[15].

PLSA allows indexing documents by topic and not just by keywords and unsupervised document categorization, using categories or topics that are suggested by the document collection[15][25]. Systems utilizing PLSA will group documents based on having similar concepts. The topics cannot be directly observed ad they are hidden (latent) in the document structure text. The concept-based indexing has better performance as opposed to keyword-based indexing, since it provides a document semantic representation in concept space[25]. PLSA utilizes the model to estimate the probability that a certain word will be used a document query term including semantically equivalent words not in the document space but with same meaning with words that appeared in the search. For instance words, 'car, and traffic will match semantically the word automobile therefore the document will have an increased probability relevance to query for a word like 'automobile'[15][25]. Each document has terms and the probability P(w—d) will give the frequency of a term w in a given document d. Each corresponding document is composed of topics and the Probability P(t—d) will give the frequency of topic t in a given document d. The model of the co-occurrence data and the corresponding word and document probabilities are derived the aspect model with well-defined probability distributions and factors with the probabilistic meaning. This topic model tries to uncover latent topics of a text collection whilst terms not occurring in a document will be assigned a zero probability(Zero frequency problem)[15][25].

For polysemy, words in a topic from PLSA can appear in other topics simultaneously[15]. One major disadvantage of PLSA is that [15] for PLSA a long document might have many words related in the same document. If a length of a document is long, many words in the document probably have relation within the document but with other documents, which reduces high-order co-occurrences. Thus, long lengthy documents are not appropriate for PLSA[15]. The training data is usually a large set of documents, which is referred to as corpus in the form of a document-term matrix indicating the number of times, each word appears in a document. The desire is for the PLSA algorithm to use the co-occurrence matrix to extract topics and describe documents as mixtures of topics. The latent variable model expresses data using mainly the 3 variables documents, words and topics[15][40]. In PLSA, the values in P(d—z), P(z) and P(w—z) are interpreted as probabilities, but in LSA, the values just show loading values, which have sometimes-negative values. Conclusion is that no prior knowledge about concepts is required as context and term co-occurrences are exploited.

PLSA(Probabilistic Latent Semantic Indexing) and LSA(Latent Semantic Indexing) have major differences. PLSA uses word probabilities given topics p(w—z) [15] whilst LSA will use a given matrix U. PLSA uses docu-

ment probabilities given topics p(d—z) whilst LSA uses matrix V The former is also based on topic probabilities p(z) whereas the latter uses a given matrix S hence for the former they are all probabilities whereas for LSA all values in all matrices are normalized and non-negative[15].

3.5.1 Probabilistic Latent Semantic Analysis: Research in the field

Hofmann[15] pioneered use of this algorithm in the field of text retrievals and in areas like document indexing, document retrieval and clustering. The primary goal of probabilistic latent semantic analysis (PLSA) is to learn unobservable probabilities based on a Maximum Likelihood framework, which allows one to infer the hidden aspects. The use of the algorithm has been even extended to fields like computer vision and audio processing. Levy(2001) applied PLSA algorithm in the retrieval and annotation of musical models. Yi[40] applied the same algorithm to classify Chinese Text and Wu[38] used it for tag clustering research.

Hyung and Lee[17] concluded that music requested for a Korean radio and the corresponding documents or letters written by the listeners requesting the music are highly correlated. Li and Hu[26] presented the PLSA algorithm to overcome problems with PLSI or probabilistic Latent Semantic Indexing like storage efficiency and performance of the queries. They proved that time complexity of PLSA query based on a dictionary generated was much more smaller and efficient. Their research they highlighted the importance of the probabilistic latent semantic dictionary which is a matrix showing relationships between documents and terms[17]. Their findings were published in the article Recommending Music Based on Probabilistic Latent Semantic Analysis on Korean Radio Episodes, concluded that by using PLSA algorithm, the Korean radio show could recommend music based on text analysis. Jin et al[39, 19] used PLSA in discovering hidden web behavioural patterns by users. In contrast to other approaches at the time which used user session clustering as well as frequent navigational patterns, Jin et al[39] used PLSA algorithm to discover hidden semantic relationships between the web user and the Web objects they were searching or navigating. These relationships were measured in probability terms hence probability inference was used to do user segmentation and web page classification and collaborative recommendations as well.

Tsai et al[13] used PLSA algorithm in analysing corporate blogs and published their findings in their article called ”Probabilistic Latent Semantic Analysis for Search and Mining of Corporate Blogs”. PLSA was used to detect keywords of certain range of topics and then used to track popular corporate conversations and topics in a blogosphere. The probabilistic approach allowed improved information retrieval in blog search and keyword detection. McInerney et al[19] also applied PLSA in their research titled ”Improving Location Prediction Services for New Users with Probabilistic Latent Semantic Analysis”. In their research, they applied the PLSA algorithm to determine mobility patterns in people’s movements on a normal day. The PLSA algorithm worked perfectly on limited training data to predict information of mobility habits of a group of users. This was a huge improvement to algorithms like Non-linear Series analysis and Eigenvalue decomposition as well as Markov Models, which relied on huge sets of training data [19].

Hu et al[37] also applied incremental recommendation algorithm based on Probabilistic Latent Semantic Analysis (PLSA) to consider negative and positive feedbacks when users visited wikis, blogs and other knowledge management platforms as well as in general Web 2.0 technologies to ask questions and get answers. User experience was improved through the user of automatic recommendations being applied. The Finnish team, Kakkonen et al[22] applied Probabilistic Latent Semantic Analysis (PLSA) as an information retrieval technique to improve problems found in the Latent Semantic Analysis algorithm. The former improved efficiency, though accuracy with the latter the team developing an Automatic Essay Assessing system for grading assignment essays. Lin et al[27] used Probabilistic Latent Semantic Analysis for developing a personalized web search. The research took advantage of PLSA model's ability to analyse co-occurrence of data to help the mining of unseen factors from web logs to personalized web searches. This was based on the premise that all web users show common behaviour and tends to acquire similar information when they have submitted a query on the web and these unseen factors would then be used to improve future search results[27].

PLSA uses an expectationmaximization (EM) algorithm that is an iterative method that is used to find maximum likelihood estimates[16]. PLSA has been used also to do document indexing, topic based document segmenting, image analysis and annotations and news classification. Bosch et al[2] used PLSA and k-nearest neighbour to do scene classification in an environment where there was grass, roads, buildings with an objective to discover certain objects in an unsupervised way. Sun et al[36] also applied PLSA algorithms in their experiment on "Unsupervised video-based lane detection using location-enhanced topic models" to detect lane detection. Choudhary[11] applied PLSA algorithms in detecting unusual activity aided by video epitomes as well.

3.5.2 Probabilistic Latent Semantic Analysis algorithm

Probabilistic Latent Semantic Analysis(PLSA) is a technique that categorizes based on topic models so as to model co-occurrences in a probabilistic framework so as to identify the underlying semantic structure of the data. PLSA is a latent variable model[16] that uses the probabilistic model called an aspect. The latent or hidden variables are represented as topics or concepts and are associated with the observed variables, which are documents and words in the text and document retrieval world[15][1].

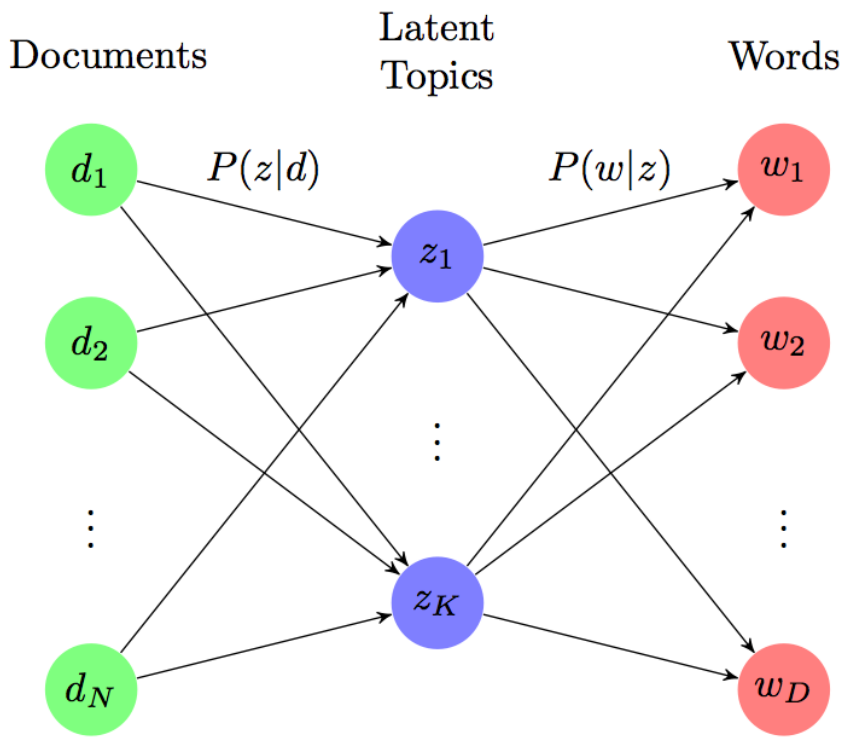
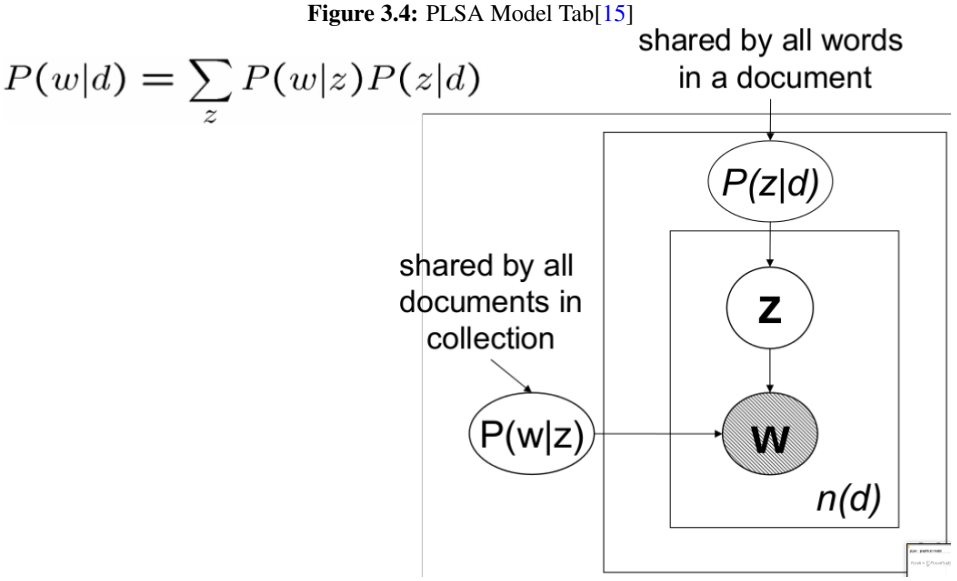


Figure 3.3: PLSA Latent Model[15]

Given a document S, a term set W and a topic set Z, the conditional probability of sentence-term P(d,w) can be described as shown below where P(w—z) is the conditional probability of words in latent semantic topics, P(z—d) is the conditional probability of topics in the documents[16].For synonyms, words in a topic from PLSA are more closely related than words in a topic from LSA. The inputs to the algorithm are a document term matrix X(w,d) as well as K the number of topics desired. The algorithm will initialise 2 arrays M1 and M2 with numbers between [0,1] and eventually normalise them to be able to have sum total of 1 along each given row. Iteration is then done until convergence. The output to this is two arrays M1 and M2 holding estimated parameters namely P(w—k) as well as P(k—z) respectively. Two parameters namely are P(w—z) and P(z—d). The equations for computing the parameters are derived using the Maximum Likelihood Estimation to derive P(w—z) for all w and z, which is a topic by term matrix. Also as output will be P(z—d) for all k and d which outputs a topic by document matrix for the topics chosen[15].

Figure 3.4: PLSA Model Tab[15]



$$P(d_i, w_j) = \sum_{i=1}^{k} P(d_i)P(z_k|d_i)P(w_j|z_k) \tag{3.1}$$

$$P(w,d) = P(d)\sum_z P(z|d)P(w|z) \tag{3.2}$$

$$P(w,d) = \sum_z P(z)P(d|z)P(w|z) \tag{3.3}$$

The two models are equal since P(z given d) =P(d given z) P(z)[15]

3.5.3 Probabilistic topic models

Aspect models are based on the idea that documents can be modelled as a mixture of topics. Each document is a probability distribution over topics. Distribution over topics represents the essence, the body, or the gist of a given document[15]. This is symmetric formulation of PLSA[15]. We select a topic z and then with the probability p(d—z) a document d and then with the probability p(w—z) words for that document. This is repeated for all documents[15].

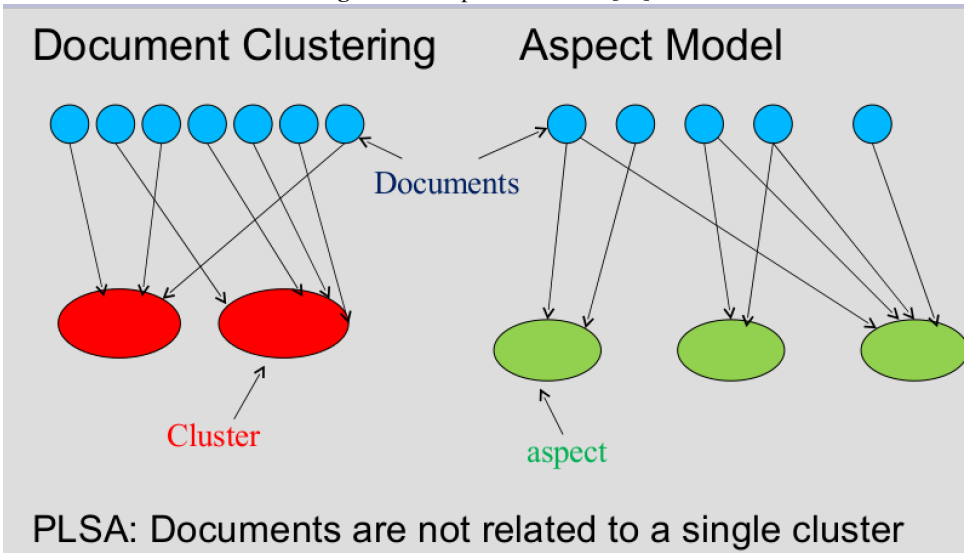
$$P(d, w_n) = P(d)\sum P(w_n|z)P(z|d) \tag{3.4}$$

3.5.4 Aspect Model

The core of PLSA is a statistical model, which has been called aspect model. The aspect model is a latent variable model for co-occurrence data which associates an unobserved class variable z k ? z_1 , . . . , z_k with each observation, an observation being the occurrence of a word in a particular document. The model involves selecting

a document d with probability P(d) and picks a latent class z with probability P(z given d) to generate a word w with probability P(w given z)[15]. A word is treated as a vocabulary item indexed by 1,...,V and these are represented as unit-basis vectors. The vth word is represented by a V-vector w such that only the vth element is 1, while the others are 0[15]. The document is treated as a sequence of N words denoted by w = 1, w_2 , ... , w_n, where wi is the ith word a given sequence. The collection of all these documents is then treated as a corpus. The set of different remaining words is called dictionary or vocabulary[15].

Figure 3.5: Aspect Model Tab[15]

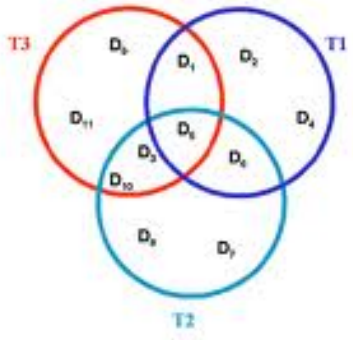


3.5.5 Document Term Matrix

The PLSA algorithm is dependent on the source data which is in the form of a document term matrix and the result is a topic term matrix where a document can belong to several topics and also a term or word can also belong to several topics as shown below.

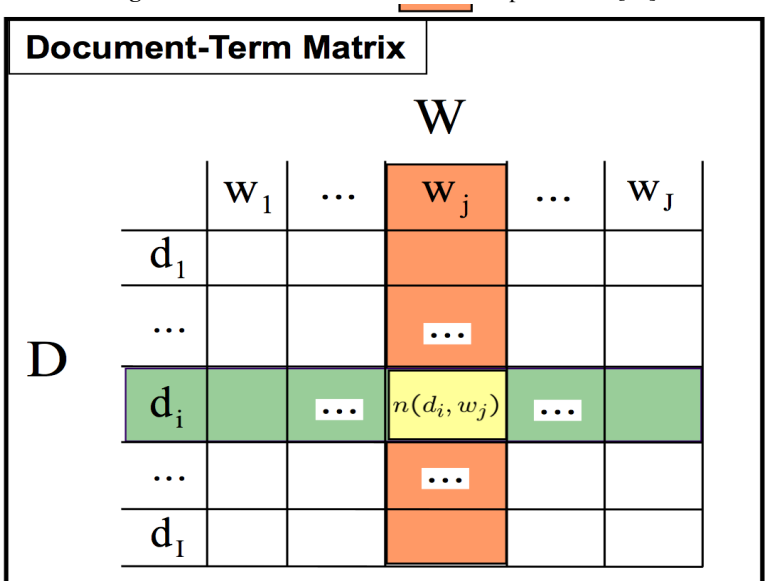
$$P(d,w) = \sum_{z} P(w|z)P(d|z)P(z) \tag{3.5}$$

Figure 3.6: Topic Term Matrix source:[15, 30]



docs	t1	t2	t3
D1	1	0	1
D2	1	0	0
D3	0	1	1
D4	1	0	0
D5	1	1	1
D6	1	1	0
D7	0	1	0
D8	0	1	0
D9	0	0	1
D10	0	1	1
D11	1	0	1

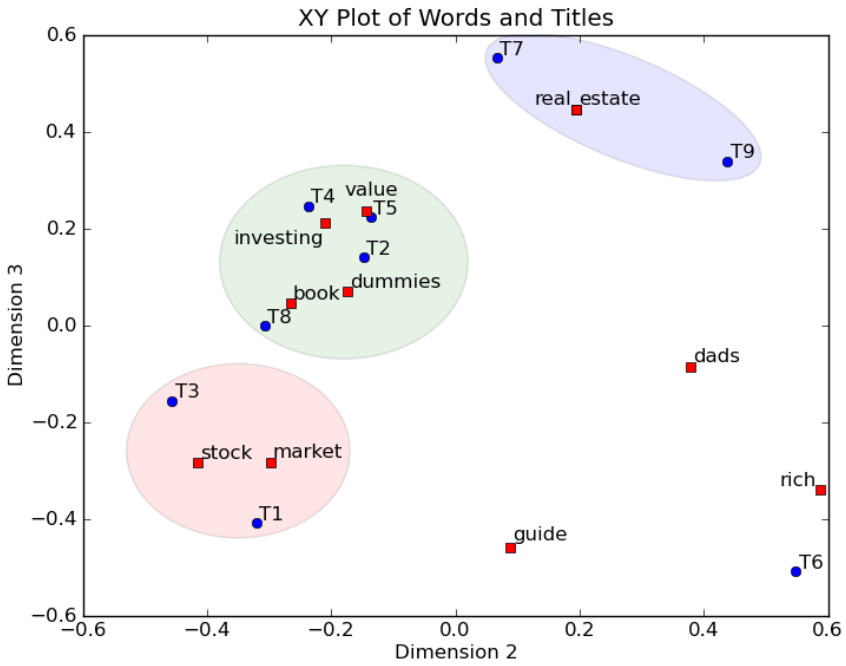
Figure 3.7: Document Term Matrix adapted from: [30]



3.5.6 Bag-of-Words Approach

PLSA is an unsupervised method based on ”Bag of Words” idea[25]. ”Bag of words” is on the premise that a document is an unordered list of words and grammatical information is lost in this sense. Documents are represented as numeric vectors in a space of words[15][25]. The word order is lost but the co-occurrences of words will give insight about the topics of the document collection. Documents are denoted as ”bags of words”, and word order is of no importance except the frequency of the words in each document[15][25]. Concepts will represent a pattern of words that appear together in documents. For instance, ’smell’, ’desert and starter’ are words that will appear in a lunch concept based documents[15].

Figure 3.8: Bag of Words[15]



3.6 Expectation Maximisation Algorithm

The PLSA model depends on the unobserved latent variables and uses Expectation Maximisation(EM) algorithm for its calculations. The EM iteration will alternate from performing the expectation denoted by E step where a function of the expectation of the log-likelihood is evaluated using the present estimate of the parameters [15]. The maximization step denoted by M step computes parameters by maximizing the expected log-likehood found in the previous step E[15]. These estimates are then used to determine the distribution of the latent variables in the next iteration or the E step till convergence is achieved.

Rather than picking the single most likely completion of the missing data on each iteration, the expectation maximization algorithm computes probabilities for each possible completion of the missing data, using the current parameters[15]. These probabilities are used to create a weighted training set consisting of all possible completions of the data. By using weighted training instances as opposed to one best completion, the Expectation Maximisation algorithm will then account for the model's confidence interval as each data training completed accounts for the confidence of the model. In summary, the Expectation Maximization algorithm alternates between the steps of guessing a probability distribution over completions of missing data given the current model, the Expectation step and re-estimating the model parameters using these completions, the Maximisation step[15]. The Expectation step means there is no need to explicitly form the probability distribution over completions, but rather there is need

to compute expected enough statistics over such completions. In an identical fashion, the Maximisation step is derived from the reasoning that model re-estimation is maximising the log-likelihood of the given data[15]. The EM algorithm is a very general iterative algorithm for parameter estimation by maximum likelihood when some of the random variables involved are not observed or either missing or incomplete[15]. It formalizes an intuitive idea for obtaining parameter estimates when some of the data is missing.

The basic E-M Strategy zX = (Y, Z) is composed of complete data X which is what the research would like to have[15], observed data Y, which are the individual observations and missing data Z which are the class assignments. The algorithm uses estimated parameters to infer Z and update estimated parameters using Y and Z. The estimation and updates iterate until convergence is attained. An initial value for π_0 is guessed first, then repeatedly alternate computing the expected value of L given π^t and $\{y_i\}$ is done, and then maximising the expectation with respect to π. The EM algorithm finds an ML solution for models with hidden variables. Each iteration increases the likelihood of the data and it is guaranteed to converge. EM consists of two steps namely the E-step, which will calculate expectation or posterior probabilities for latent variables given the observations by using the current estimates of the parameters[15]. In the E step, a function is created with the expectation of the log-likelihood using the current parameter estimates[15]. The M-step will update parameters such that the data log-likelihood increases using the posterior probabilities found in the Expectation step and also calculates parameters which maximize the expectation of the log-likelihood. These parameter estimates are used to determine the distribution of the latent variables in the next E-step iteration[15].

The EM algorithm defines an iterative process that allows to maximize the likelihood function of a parametric model when some variables are unknown or hidden. Theoretically using a minimization algorithm to find the maximum of the likelihood function for all parameters is also considered equivalent[15]. The reason EM is needed is that Machine-Learning conditions resolved better in static environments. Expectation Maximisation is useful wherever the model has fitting problems due to missing, incomplete and or hidden data hence iteratively estimates Maximum Likelihood Estimate(MLE) to get this missing data[15].

$$log p(x|a) = log \sum_{z} p(x,z|a). \tag{3.6}$$

3.6.1 PLSA Algorithm

Data: Document and words terms
Result: Latent topics in the document structure
initialization ;
Step 1: Remove all stop words from the document;
Step 2: Start training PLSA and initialize K,L,M randomly ;
Step 3: Build Term-Document Matrix ;

Step 4: Initialize probabilities P(z), P(d—z), P(w—z) randomly.
Step 5: Repeat the EM algorithm until convergence
while *no convergence* **do**
    step a : E-step: Compute posterior probabilities for the latent variables. Replace missing values by estimated values using estimated parameter values as true values ;
    step b: M-step: Maximize the expected complete log-likelihood likelihood and estimate parameters using estimated values as observed values ;
end
Step 5: Print the Topic Term matrix;
Step 6: Print the Topic Document matrix;

Algorithm 4: PLSA implementation algorithm [16]

PLSA assumes that each document d (with word vector w) is generated from all topics, with document- specific topic weights. The generative process of PLSA is the following. Given a document collection with n documents, we represent it as a document-term matrix with entry c(d, w), which represents the count of word type w in a given document d. During iteration firstly a topic z = 1 . . . K p(z) is picked follwed by a document a document and a word type independent of each other, though they both depend on the topic given by d p(d—z = k) and w p(w—z = k). The process then generates an additional word w to document d. This is repeated until a document-term matrix is generated. Based on this model , the probability of picking a given cell (d, w) is given by the equation 3.7.

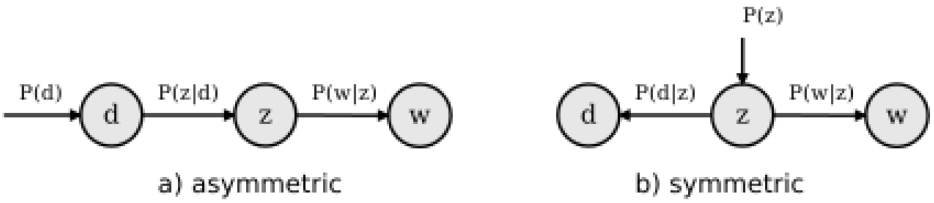
$$P(d,w) = P(d)P(w|d) = P(d)\sum P(w|z)P(z|d) \tag{3.7}$$

$$where -- > P(w|d) = \sum P(w|z)P(z|d) \tag{3.8}$$

Conditional Independence assumption is based on the premise that documents and words are independent given z which means the equivalence holds. The first formulation is classified as assymetric formulation where for each document d, a latent class is the conditionally chosen to the document based on P(z—d) as well as a word is produced from the latent class based on the probability P(z—d). The symmetric formulation, sees both w and d being deduced from the latent class z in identical fashion using their condition probabilities namely P(d—z) and P(w—z)[16, 25] as shown in the Figure 3.9.

$$P(d,w) = \sum_{z} P(z)P(d|z)P(w|z) \tag{3.9}$$

Figure 3.9: PLSA Model Tab



The modelling assumption says that conditional distributions P(w—d) is given by an approximation of a combination of factors P(w—z), where the model parameters are T = p(z), p(d—z), p(w—z). The equation 3.10 is then used to find the MLE to maximize the likelihood of the observed document term matrix[16, 25].

$$maxT = \sum_{d=1}^{n} logP(d,w) \tag{3.10}$$

$$maxT = \sum_{d=1}^{n}\sum_{w=1}^{V} c(d,w)log(\sum_{z=1}^{K} P(z)P(d|z)P(w|z)) \tag{3.11}$$

3.6.2 Expectation Maximization algorithm steps

The E-step and M-step are repeated until convergence. P(w—z) are the topics. Each topic is defined by a word multinomial and has distinct semantic meanings[16, 25]. From p(d—z) and p(z), p(z—d) p(d—z)p(z) is computed. The probability, p(z—d) is the topic weights for a given document d. The expectation maximization algorithm initial states are summarised below.

Initialisation:

```
X is a matrix of size |D| x |W| representing the observed document-term matrix
P(z) is a vector of length |Z| and it has a uniform initial value.
P(w|z) is a  normalised(by random starting value) matrix of size |W| x |Z|.
P(d|z) is a normalised(by random starting value) matrix of size |D| x |Z| .
P(d,w,z) is a normalised(by random starting value) matrix of size |D| x |W| x |Z|.
```

E step:

Compute the posteriors for the latent variables z using equation 3.12[25].

$$P(z_k|d_i,w_j) = \frac{P(w_j|z_k)P(z_k|d_i)}{\sum_{l=1}^{2} P(w_j|z_l)P(z_l|d_i)} \tag{3.12}$$

M step:

Maximising with respect to P(w given j—z given k) and P(z given k—d given i) whose total sum of probabilities is both 1 for k from 1..K and n from 1..N is given by equation 3.13[25].

$$\sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i,w_j)\sum_{k=1}^{K} P(z_k|d_i,wj)log[P(w_j|z_k)P(z_k|d_i)] \tag{3.13}$$

Thus the M-Step is summarised as below based on the maximum likelihood hypothesis

$$\begin{aligned} P(w_j|z_k) &= \frac{\sum_i n(d_i,w_j)P(z_k|d_i,w_j)}{\sum_{m=1}^{M}\sum_{n=1}^{N} n(d_n,w_m)P(z_k|d_n,w_m)} \\ P(z_k|d_i) &= \frac{\sum_{j=1}^{M} n(d_i,w_j)P(z_k|d_i,w_j)}{n(d_i)} \\ P(z) &= \sum_{d=D}\sum_{w=W} n(d_i,w_j)P(z_k|d_i,w_j) \\ where --> n(d_i) &= \sum_{j=1}^{M} n(d_i,w_j) \end{aligned} \tag{3.14}$$

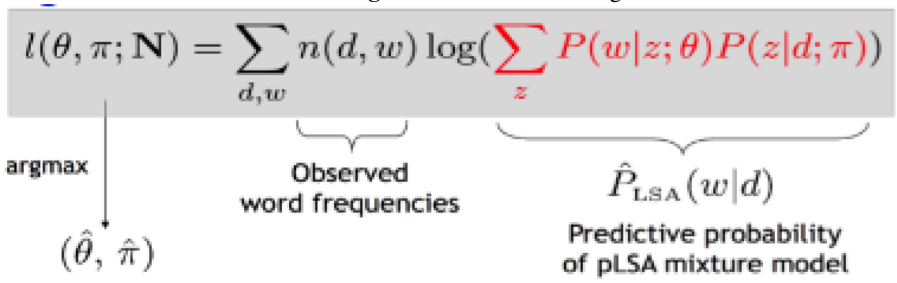
3.6.3 PLSA via Likelihood Maximization: Parameter inference

EM begins with an optimization of a likelihood function simplified by assuming that a set of ”missing” or ”hidden” data is known[25]. Computing the expectation of the data likelihood in the E-step can be done by finding the expectation of the latent data. Parameters are inferred using Maximum Likelihood Estimator (MLE)[25]. The probability of observing the complete document collection is then given by the product of probabilities of observing every single word in every document with frequency of occurrences[25]. Maximum likelihood, also called the maximum likelihood method, is the procedure of finding the value of one or more parameters for a given statistic, which makes the known likelihood distribution a maximum. It is a method of estimating population characteristics from a sample by choosing the values of the parameters that will maximize the probability of getting the particular sample actually obtained from the population[25]. The success factor is to find model parameters that maximize the log-likelihood or alternatively the average predictive probability for the observed word frequencies of occurrences, which is a non-convex optimization problem. In decision theory, a major decision-making rule says under conditions of uncertainty states, the decision maker should suppress all possible events and give priority to the one most likely to occur[25]. The decision maker should then select the course of action that produces the best possible result which is a maximum, which can be a gain or a loss[25].

To summarize, the function is given by taking into account observed values starting from x1....xn as fixed observed values with the function observing them as fixed parameters of the function and will be the function's variable and allowed to vary[25]. Based on this, the distribution function will be defined as the log-likelihood which is more convenient scaled version. Logarithm is a monotone transformation, so there is an argument that says its a fact that the MLE or Maximum Likelihood Estimate is equivalent when maximising the likelihood or maximising the log-likelihood function, [25]. The likelihood is given by the sequence of equation below 3.15

$$L = \prod_{i=1}^{n}\prod_{j=1}^{m} p(w_i,d_j)n(w_i,d_j) L = \sum_{i=1}^{n}\sum_{j=1}^{m} n(w_i,d_j) log(\sum_{j=1}^{k} p(wi|zl)p(zl)p(dj|zl))) \tag{3.15}$$

Figure 3.10: Maximum Log Likelihood Tab



3.7 Conclusion

The chapter did a literature review of information retrieval concepts. Various models like PLSA, LSA and DTA were discussed as examples of information retrieval models. Discussion on previous literature on PLSA algorithm centred on various applications of the algorithm as well as application in the industry. The PLSA algorithm was analysed including the various steps followed. The Expectation Maximisation algorithm was identified as the backbone of the PLSA algorithm and an in depth discussion of its Expectation and Maximisation steps were also discussed in the research. As part of the EM algorithm, the Expectation and Maximisation steps were discussed in depth including the Maximum Likelihood to maximise the EM algorithm. The next chapters look into the implementation of the crawling to gather data on the web and storing it, as well as using the PLSA algorithm to do the classification of the data. The methodology used in this research is also explained.

Chapter 4

Research Methods

4.1 Introduction

This chapter will look at the research methodology used as well as the population sample size and also discusses the ethical considerations in this research. The quantitative research envisaged will include a description of the population to be studied. The chapter shall include population sample, relevance and a list of variables and data collection tools to be used. The means used to analyse the data is discussed and the chapter concludes with sections on the limitations. This section is to justify the means in which the study was obtained and will help in giving it purpose and strength as it will then be truthful and analytical sound. Research and ethical considerations are discussed. The credibility of findings and conclusions extensively depend on the quality of the research design, data collection, data management, and data analysis. This chapter will be dedicated to the description of the methods and procedures done in order to obtain the data, how they will be analysed, interpreted, and how the conclusion will be met. All these will help in the processing of the data and the formulation of conclusions.

4.2 Quantitative research method

The study focused on utilising secondary data from the web by downloading internet data. The quantitative research method permits specification of dependent and independent variables and allows for longitudinal measures of subsequent performance of the research subject. Besides, quantitative research plainly and distinctively specifies both the independent and the dependent variables under investigation. It also follows resolutely the original set of research goals, arriving at more objective conclusions, testing hypothesis, determining the issues of causality and eliminates or minimises subjectivity of judgment.

4.3 Population Relevance, Sample Design and Sampling Method

The research since its a data mining research, focused on large downloads of documents as part of the research. The research crawled for over 3 000 web pages and used it for the Expectation Maximisation Algorithm. The data also went through pre-processing hence for every file there was a post-processed file as well.

4.4 Ethical Considerations during Crawling

The research considered the crawling selection policies of all the websites it downloaded and to prevent performance problems on all these websites the research also had a 3 second delay on crawling one page after the other as well. Some of the ethical issues the research considered include:

- Harm and Risk- The researcher gave an undertaking that the research shall not prejudice any of the subjects. Key to this is the fact that the research avoided overloading websites by limiting number of websites parent URL to only 200 and also put a delay of three seconds to avoid lots of multiple threads.
- Honesty and Trust-Honesty and trust of data analysis and collection shall be observed. ROBOTS rules are observed.
- Privacy, confidentiality and honesty- Company websites are not downloaded for privacy reasons.

4.5 Crawling Limitations

Some of the limitations of crawling on the web done by the research include restricting to English Websites only since there are a huge number of different languages on the website. For perfomance reasons, the downloading of the websites was restricted to only 100 pages per website and there was also strict observation of the Robot Exclusion policy. Robot exclusion is where websites and download pages are specified in a configuration file showing if they are allowed to be visited or not. The implementation includes reading a robots.txt on each website to see which pages are allowed or not on the specific web server. The research also limited crawling to only text websites only and excluded downloading videos, music and pictures. The impact is the loss of huge potential data that could be used by the PLSA algorithm.

To prevent flooding webservers with multiple threads then causing shutdown of such web servers and violating web politeness, the research put a time delay between consecutive downloads by 3 seconds to the same server.The research also only opened one HTTP connection to prevent webserver flooding. The crawling also normalised the websites hence ignoring other page fragments and this is termed link canonicalisation[10]. All the text downloaded was in HTML format and no other format since the crawler is not designed with such sophisticated language models like RDF or XML type languages. Character Encoding on some websites also puts a limit to the web sites being downloaded as such content was not decoded hence the research ignored such content.

4.6 PLSA Limitations

The PLSA algorithm is limited to identifying only English words and also the research does not take into account the same words written in different forms for instance 'time' and 'times' are treated as two different words yet they mean semantically the same thing. Some words mean more than one different thing semantically depending on how they are used in the sentence as well hence this brings a semantic constraint. The tokenisation 'bag of words' model of the algorithm also brings the problem of losing the meaning of some of the words hence splitting a phrase or double barrel word will lose the meaning of the word.

```
kennedys-MacBook-Pro-2:13.0 kennedy$  sudo /usr/local/mysql/support-files/mysql.server starting
```

4.7 Define the system architecture/infrastructure

The system included running a combo of java and python programs to crawl the web for data and pre-processing before it gets classified using the EM algorithm. The data was split into training and testing data for the given algorithm. The crawling architecture is shown in the figure below[**?**]. The classifying of the data was done using the EM algorithm and the algorithm is an iterative algorithm for parameter estimation by maximum likelihood when some of the random variables involved are not observed. The non-observed values are either missing or incomplete[10]. The EM algorithm is a formal method used to obtain by iteration close parameter estimates of missing data and is summarized in the following algorithm **??**:

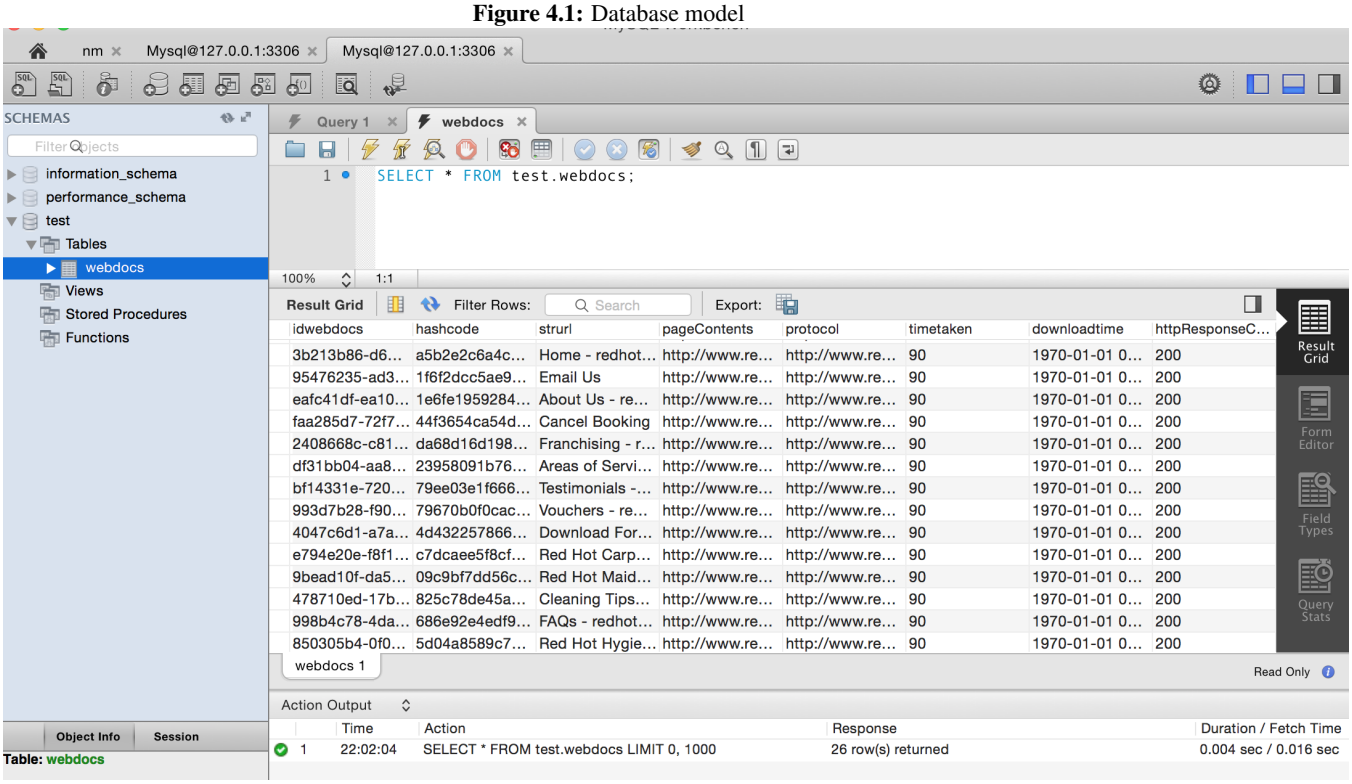
Step 1: Select startup parameters.
Step 2: Initialise parameters.
while *not yet converged* **do**
    Step 1: Using estimated parameter values as true values,replace missing values
    Step 2: Using estimated values as the observed values iterate.
end
Step 5: Print the output

Algorithm 5: EM Algorithm Definition[25]

4.7.1 Define the database model

The crawled web pages were saved before and after pre-processing of the given data, which was then used for the classification of the data. The web pages were saved as BLOB Binary Large Objects in an MySQL Database and given unique indexes, which were then used to refer to these objects as well. The database includes a table called Web pages with the following fields namely, reference number, blob column for the web page, date saved, URL downloaded as well as user responsible for the download.

Figure 4.1: Database model



4.7.2 Non-functional requirements

Non-functional requirements included web security. Security of the websites being crawled is a major issue since it is imperative that all URLs that are secured are excluded from being crawled as well. The response time of the crawling program is of critical nature since downloads might fail before the data has finished to be saved. Timeouts might occur and delays were be done between webpage downloads to avoid being blocked by websites which are in danger of being overloaded with data requests and might fall over. Storage of the database is also a key issue but placing a cap on the number of web pages to 200 per URL reduce the number.

4.7.3 Define a dictionary for all relevant concepts/entities

The research excludes all the stop words that are frequently used and these will be put in a dictionary to be referenced during the data pre-processing. Only pdf and html as well as word documents were considered and all image documents excluded during the crawling and saving of the web page data. The research also excluded all words with letters less than three on the argument that such words do not contribute to topic identification meaningfully. Other non alphabetical characters like '?' were excluded from consideration as well.

4.7.4 Infrastructure Setup

The system was implemented on an MacOS system with 16GB RAM of main memory. The machine used runs on Mac and MacOS Operating System. A Virtual Machine for Ubuntu was also used to install and run MySQL Database. The software used in the research include, Subversion for software repository with the Leicester Server, Eclipse for java development and implementation, Tortoise SVN for all documents repository, MySQL Workbench to monitor MySQL as well as VisualVM for monitoring java processes used in the research. A 1GB Internet based with Wi-Fi Internet is used for internet research using the local South African internet service provider, Vodacom as well as Neotel is used in this research.

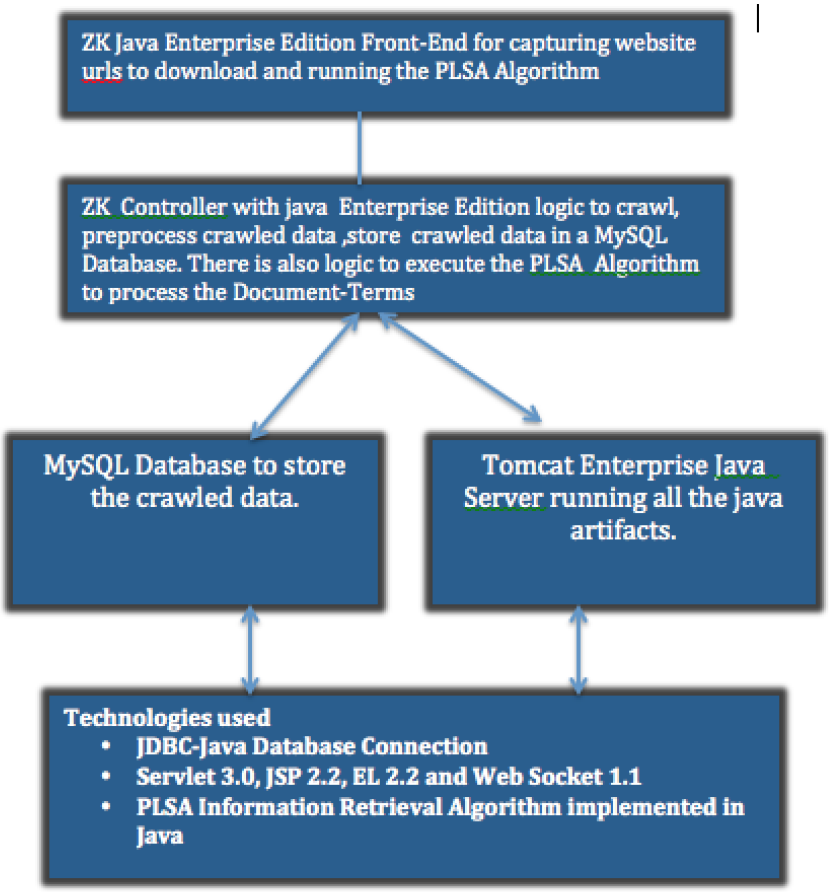
```
kennedys-MBP-2:~ kennedy$ sudo /usr/local/mysql/support-files/mysql.server start
Password:
Starting MySQL
. SUCCESS!
kennedys-MBP-2:~ kennedy$
```

4.8 Define the interfaces to other systems

The major interface is the fact that the java program will crawl around the chosen websites and retrieve URLs and download the given web pages for further processing and categorization. A java page is needed to show the output as well as input for the crawling. Reports are also run to show all the crawled data and shown in a web GUI frontend page with scrollable features. The technology used is called ZK Framework together with a Tomcat java server and JBOSS application server. The java application links to the MySQL database to store the crawled data as well as the output of the PLSA algorithm.

The project designed and ran the java program on a small web engine called Tomcat. A small web admin program using java ZK Framework was used for specifying the input of the web projects to be classified as well as the urls to be crawled as well.

Figure 4.2: PLSA Design Model



4.8.1 ZK Framework

The research used an Model View Controller (MVC) application to present and allow user input to crawling and the PLSA document classification system forming part of this research. ZK is an Ajax-based(asynchronous java and xml) and an event based java framework for rich client model web based rich user clients[7]. The applications are presented in the form of XUL-compliant files and the web pages require little or non-JavaScript[7]. The pages also work well with other modern java frontend technologies like java server pages as well as java server faces, which are event driven as well. The GUI designed has a sleek front pages with controllable User Interface components. The basic ZK page is called a zul file and if user interacts with any component[7], then the controller model will process[7]. The controller model coordinates between the view(zul pages) and the model, which in this case includes connection models to the database. Events from the view are then used detect changes and upload them to

the model as well as retrieving data from the model. The following diagrams show images of the ZK framework.

Figure 4.3: ZK MVC Tab[7]

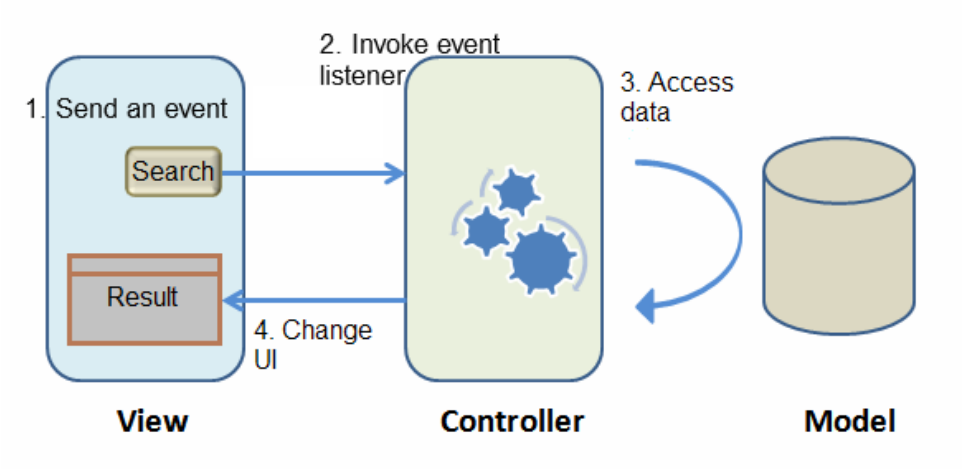


Figure 4.4: Creating ZK MVC Tab[7]

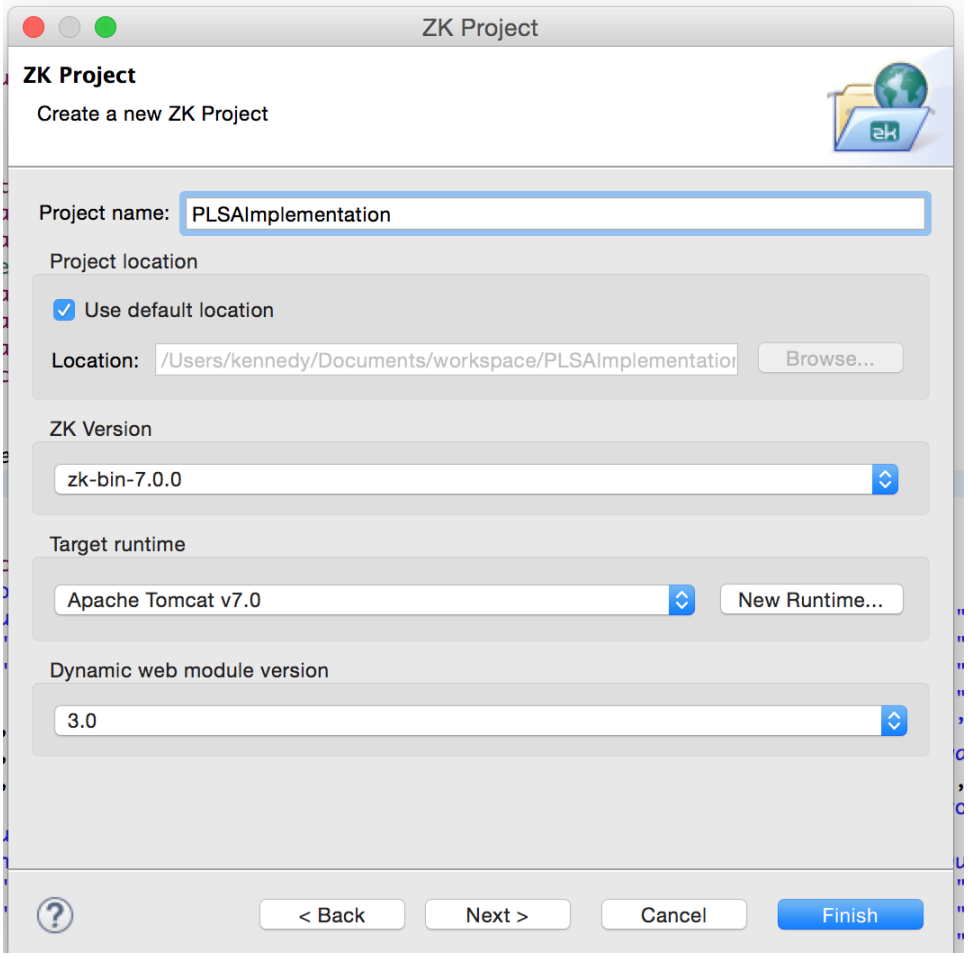
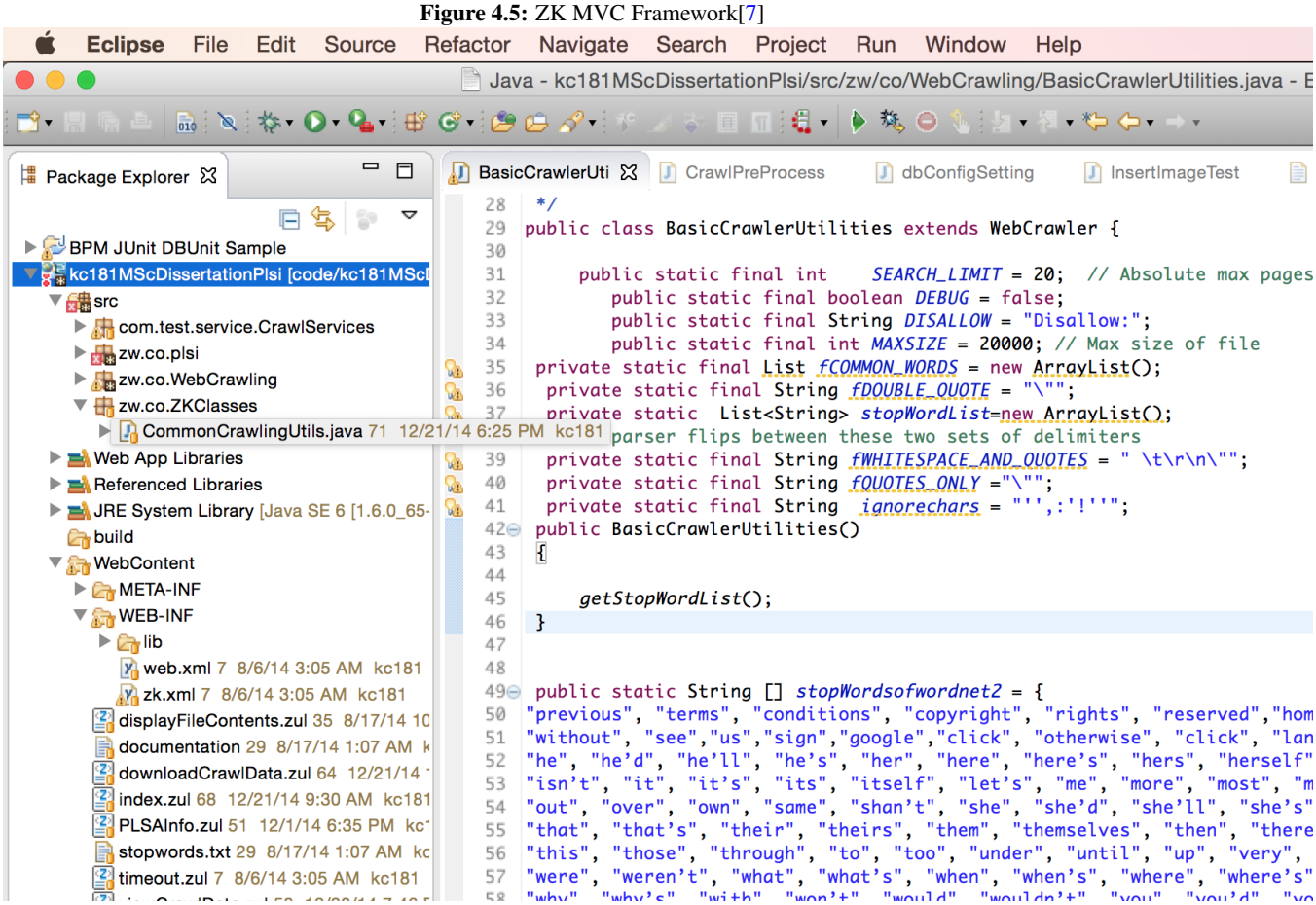


Figure 4.5: ZK MVC Framework[7]



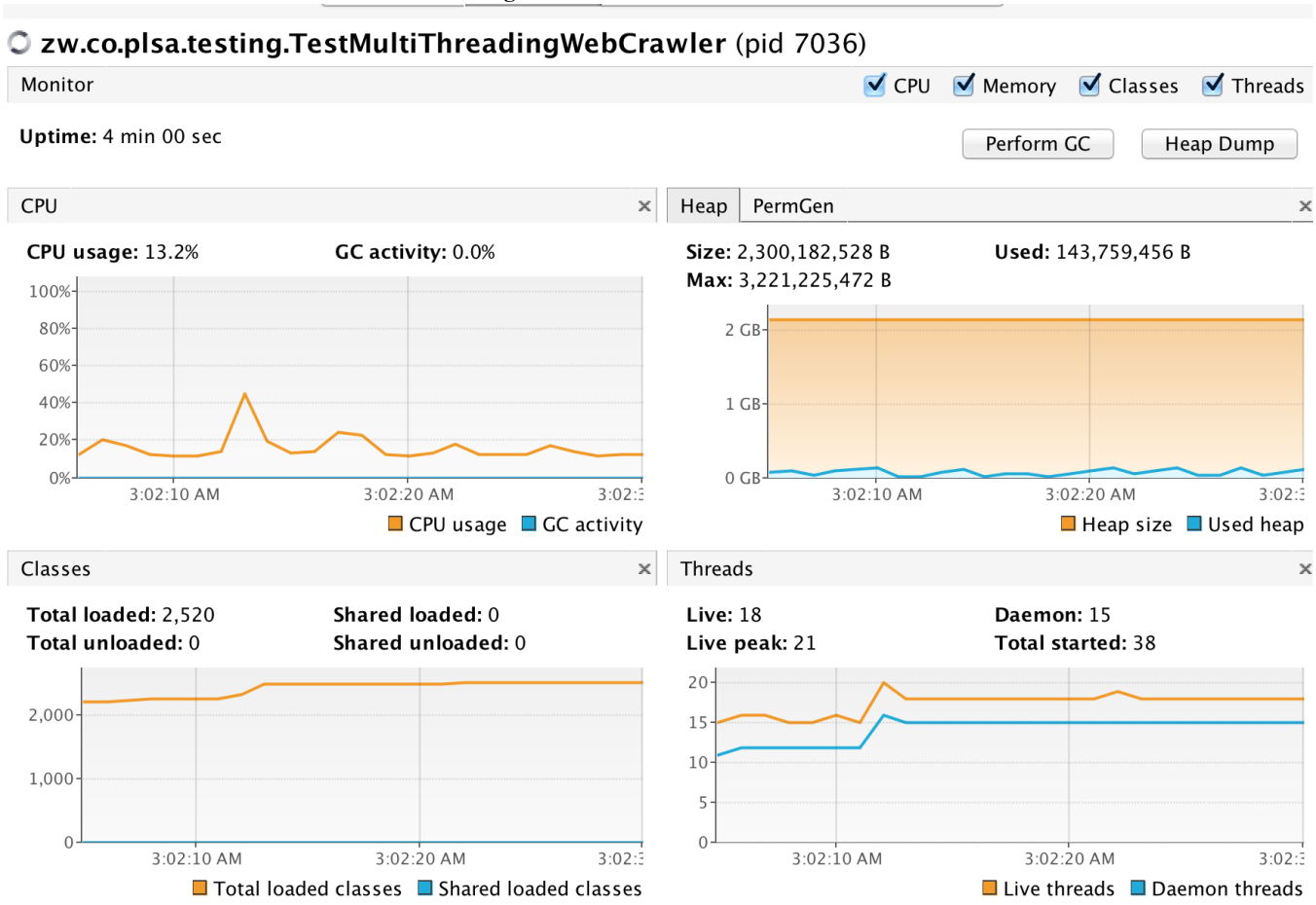
4.8.2 Java Application Server-JBOSS and Tomcat

To do the implementation the JBOSS java enterprise server was used to run the crawling as well as the PLSA algorithm implementation. Initial tests were done using Tomcat 7.0 though the research to improve performance felt the need to use a more reliable server with more performance and robustness qualities. Eclipse was used as the tool for development of the java programs as well being used to design the system models.

4.8.3 Performance Monitoring using JConsole and VisualVM

Monitoring of the PLSA and the crawler is an important part of this research and efficiency is regarded as important. Various java tools are used to do the monitoring and these include JConsole, which is a java generic swing application for application monitoring. For detailed monitoring, VisualVM an open source java-monitoring tool was used. For all round monitoring to include socket and database monitoring, the JProfiler tool was used. CPU and memory statistics even though they were found from these tools, were also derived from the MacOS command line.

Figure 4.6: Visual VM



4.8.4 Logging and Exception Handling using AspectJ

Logging was used in this research for two issues. The first being to log all start and end times of all executed methods so as to measure performance in both during the crawling phase of the research as well as the implementation of the PLSA algorithm. The second issue involved using logging with Log4j instead of the System.out java printing method which has huge performance bottlenecks Through use of point cuts, join points and advices based on the Aspect Oriented Programming AspectJ model, the research utilised its crosscutting benefits to be able to show when all methods started and stopped. The example shows a snippet of one such aspect used in this research.

```
pointcut extractWebCrawledLinks() : execution(*
    zw.co.WebCrawling.PLSAWebCrawler.extractWebCrawledLinks(..));
   before() : extractWebCrawledLinks() {
     PropertyConfigurator.configure("crawlingLog4j.properties");
        start5=System.nanoTime();
```

```
    log.warn("extractWebCrawledLinks for website has started: ");
}
after() : extractWebCrawledLinks(){
    PropertyConfigurator.configure("crawlingLog4j.properties");
    end5=end5+System.nanoTime()-start5;
    log.warn("extractWebCrawledLinks for website has ended: "+end5/1000000+" milliseconds");
```

4.8.5 Risk Table

The following table shows the risks encountered from starting the research to the implementation of the PLSA algorithm as shown below.

Table 4.1: Risks Table

Crawling Implementation Table	
Risk	**Remedy**
Different languages for data on the web	Restrict to using English websites only
Infinite loop when downloading web data	Limit to 100 pages per website. Use Robot Exclusion where websites and pages are specified in a configuration file as not to be visited or indexed or crawled. A website administrator would have put a robots.txt file at the root web directory and all crawlers should abide by it.
Heterogeneous data being images , and text	Limit research to only text pages only
EM Algorithm is affected by large number of missing data	
Flooding web servers if multiple threads are used and violating web politeness visiting limits	to open only one connection to a given Web server at a time and to take a delay between two consecutive accesses:
Link Canonicalization leads to various types of pages being downloaded and some links are page internal fragments	Equivalent variations of ending directory normalized will have their ending slash removed and all page fragments are ignored.
Text stored in many different formats	Use tagging for all downloaded text and Character Encoding
Storing the downloaded web documents.	Store as BLOB objects in a database,Chance of losing track
Using supplicated and same versions of the web documents and near duplicate detection and spam pages	Use check sums
Noise in the form of advertisements,images, copyright information as well as general navigation links	Use DOM representation and traverse across the tree removing unneeded information.

4.9 Conclusion

The chapter looked at the research methodology, population sample as well as the ethical considerations done in this research as well. Sampling methods, research limitations and ethical considerations are also discussed. The next chapter details the implementation followed in this research starting with the crawling of the web data and

the classification using the Expectation Maximisation Algorithm as part of the Probability Latent Maximisation Algorithm.

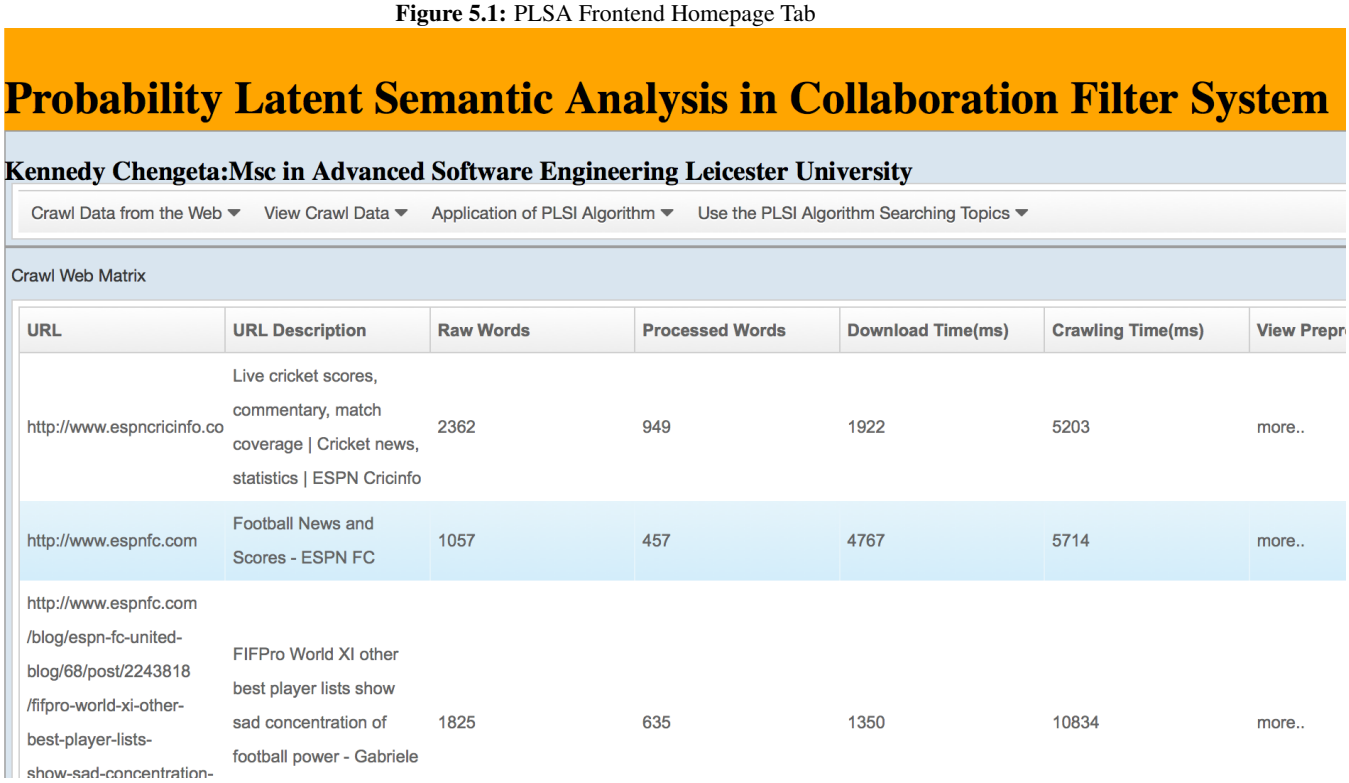
Chapter 5

Data Extraction Web Crawling

5.1 Introduction

The chapter details the implementation of data extraction method used in this research. It details use of crawling as the web method used to crawl websites to retrieve web data. Whilst the implementation is done using java and jboss application as the java application server, the storage of the processed data is done using a MySQL database and the documents are stored as blob or binary large objects. Pre-processing is also discussed with the research having implemented removal of stop words or common words, removed all the HTML tags from the source web document as well the removal of other non-alphabetical characters. The performance of the crawling is done and measured using a tool called Jconsole as well as VisualVM, which are plugins of the java framework. A java frontend designed using a ZK open source java framework, eclipse as well as MySQL database is also discussed. The data is stored in an open source database tool in this case being MySQL database. A table called 'crawledWebData' is used to store the crawled data. The chapter details the results analysis as well as the database results and also the performance bottlenecks and the ethics observed in this research.

Figure 5.1: PLSA Frontend Homepage Tab



5.2 Crawling

5.2.1 Design

The web crawler is divided into a controller module including a pre-processing function, parser and fetcher module as well as the ZKFrontEndGUI module. The ParserFetcher module includes functions to crawl the web and how urls are processed to prevent recrawling an already processed urn. The ZKFrontEndGUI module includes functions to present a web page for a user to enter web urls to crawl and other data parameters. It also shows all the websites that have been crawled as well as being able to search for a document that has been crawled by a given URL. The frontend GUI module is web based and also has a login user name and password for security purposes.

The Controller module has all the logic to process the crawled data and this includes functions like pre-processing the data to remove stop words, non-alphabetical characters as well as removing non-English documents. The controller also has implementation logic in java to save the post-processed data and other parameters in the database. The parserFetcher module is for retrieving and fetching all the links of the crawled urls as well stripping all the html tags. The module will consider exclusion of all urls that are listed in the 'robots.txt' of websites and also make sure the processing is done till the given maximum crawling pages number is reached. The research crawling process will write all the output to a database saved as files in a blob object. Logging is done using java

open source packages called Log4J as well as AspectJ tool.

5.2.2 Tools needed

The research installed JDK 1.6 and Eclipse Luna 4.4. The choice of version 1.6 of java as opposed to 1.7 is primarily the fact that its the one used widely in the South African market and some of the java archive files used were compiled using this version of java. MySQL Database was downloaded store the crawled URLs and the performance statistics as well as documents downloaded as binary large objects. Tomcat 7.0 and JBOSS, popular webservers running java engines were downloaded and installed. For crawling the respective jars downloaded included itext.jar for printing files to acrobat reader format. The research also used log4J for logging all the tracing data and logs as well as aspectj.jar for monitoring start times and end times for each java method launched. For implementing a Model View Controller Framework to allow a web browser interface to enter websites to download the research downloaded ZK Framework and implemented a java frontend tool using zk zul files which are form of java server pages. To connect to the database the research used jdbc(java database connect) driver for MySQL database, mysqljdbc.jar and wrote utility class to maintain java connectivity. MyWorkBench and phpMyAdmin, both with GUI or Graphical User Interface were used as the administration tools for connecting to the MySQL database and running Standard Query Language SQLreports. Unix commands were however used to start the database as shown in Appendix.

5.2.3 Start crawling using Java

The crawling methodology used in this research is based on a java framework. A new project is created in Eclipse, an open source development tool. Various java archive files or jar files are also used to reference the project namely jsoup java library as well as the MySQL jar namely mysql-connector-java-xxxbin.jar. These jar files are then put on the project build path. The project goal is to retrieve data from the web and classify it. The aim being to be able to use unclassified data to determine which category a given concept belongs to as well as identification of users data requirements upfront. The crawling java processes are divided into the following methods for processing as shown in the table.

```
Data: Web sites to be crawled
Result: Document and words terms
initialization urls to be crawled;
foreach url in URLList do
    validate URL;
    check if URL is robotSafe;
    generate a new processing thread to allow multithread ;
    while list of unvisited URLs is not empty do
        take URL from list ;
        fetch content ;
        record whatever it is you want to about the content ;
        if content is HTML then
            parse out URLs from links ;
            foreach actor in actors do
                if it matches your rules ;
                and its not already in either the visited or unvisited list ;
                add it to the unvisited list ;
            end
        else
            ...;
        end
    end
end
```

Algorithm 6: The Web Crawling Algorithm[10][23]

Table 5.1: Crawling Implementation Methods

Method 1	**crawlMultiThreadingWebCrawler**	This method implements the java multi-threading for batch of seed Urls
Method 2	**extractWebCrawledLinks**	This method will extract all the web links on any visited page and add it to the univisited links
Method 3	**guessLunguageWithConfidenceInterval**	This method will call the language guessing web service to decide the language it belongs to.
Method 4	**startfetchPLSACrawledPageContent**	This method will involve fetching and downloading contents on the crawled page storing in the database.
Method 5	**Pre-process and Write To File**	This will involve preprocessing the url file and writing to a file ewhich will then be uploaded on the database
Method 6	**Save Web Page Record**	The method allows to save the web page on a database using blob object.
Method 7	**Analysis and Conclusions**	Analysis and conclusions are done in this chapter.

5.2.4 Crawl Data Persistence

The decision to store or persist some of the objects as java beans was taken. During the crawling of each page, data would be persisted in a java bean object shown below and while all the data has been gathered including the pre-processed and post processed files then a single write to the database would be done, this is to prevent multiple database connection which have huge performance impact. The research though of using transaction commits, and committing in batches but discarded this idea since when downloading multiple threads, there were duplicating threads being uploaded in the database due to the mixing in times of the commits to the database. Some of the data persisted in the java object include start and end times as well as seed url. The files were saved in the local files and then saved as blob objects on the database.

```
private String idwebdocs;
private String hashCode;
private String strUrl;
private String pageContents;
private String protocol;
private int httpResponseCode;
private long downloadTime;
private long timeTaken;
private Blob originalWebDoc;
private Blob refinedWebDoc;
private Timestamp startDownloadTime;
private Timestamp endDownloadTime;
private Timestamp startCrawlime;
private Timestamp endCrawlTime;
private long originalFileSize;
private long processedFileSize;
private int wordsProcessedFile;
```

5.2.5 Retrieving Web Links

The first step is to verify the URL first by checking if it begins with 'http'. All picture files will be excluded. ASP.NET the following file extensions will be considered aspx, axd, asx, .asmx, .ashx, .aspx and ashx. For Cold Fusion cfm file extension will be considered. For Flash ,the file extension swf will be considered. Generic html extensions will include, html, htm, xhtml, jhtml and java web servers will also contribute jsp, jsf, jspx, wss, zul, .action(for struts2 web pages), .do(for struts web pages). XML based pages, rss feeds, .atom Atom Feeds(RSS) and web pages as well as twitter page links will be excluded from the study. PHP the most prevalent website language will have the following file extensions namely php, php4, php3 and phtml. Other extensions considered include .cgi and Django web server's .dtl web pages. IBM based Lotus Notes will have the nsf file extension. One of the key features to retrieving web urls is maintaining state of the following

- The seed url or starting URL.
- List of all unvisited URLs.
- A list of all visited URLs
- A list of all illegal characters and rules not allowed in a web URL. All web Urls with illegal characters are rejected.
- Keep persistence of the crawled urls in a MySQL database for further text processing and analysis.

Some websites that failed validation as shown in the log4j output files indicate that they were mostly Facebook, Twitter and multimedia web URLs. The research crawler implemented a procedure barring all video urls, twitter urls as well other social media urls found during the fetching phase due the fact that this data was needed for text mining from authentic sources hence the need to bar individual postings or comments on twitter where there is also high risk of spelling errors and use of unofficial English language as well as varying dialects and use of native language on such postings by users as well. One finding was on a South African website were comments by local contributors were using the local Zulu and Xhosa languages prompted the research to put that restriction. The research also did set follow redirects to false to prevent diverting from the parent website. The This somehow mitigated problem of many popular websites used in this research having many advertised links with no relation to the topic in question.

Figure 5.2: URL Validation Errors

```
22:22:31,491 ERROR zw.co.WebCrawling.PLSAWebCrawler - link has been accessed before or is not validhttp://
www.strategyawards.com/uploads/gallery/PRJ21898/962/resizedimages/PRJ21898_962_dr_temel_kotil_jpg.jpg
22:22:31,491 ERROR zw.co.WebCrawling.PLSAWebCrawler - link has been accessed before or is not validhttp://
www.strategyawards.com/uploads/gallery/PRJ21898/962/resizedimages/PRJ21898_962_ad-stratawards14-17.jpg
22:22:31,491 ERROR zw.co.WebCrawling.PLSAWebCrawler - link has been accessed before or is not validhttp://
www.strategyawards.com/uploads/gallery/PRJ21898/962/resizedimages/PRJ21898_962_ad-stratawards14-04.jpg
22:22:31,491 ERROR zw.co.WebCrawling.PLSAWebCrawler - link has been accessed before or is not validhttp://
www.strategyawards.com/uploads/gallery/PRJ21898/962/resizedimages/PRJ21898_962_ad-stratawards14-06.jpg
22:22:31,491 ERROR zw.co.WebCrawling.PLSAWebCrawler - link has been accessed before or is not validhttp://
twitter.com/
22:22:31,492 ERROR zw.co.WebCrawling.PLSAWebCrawler - link invalidhttp://www.strategyawards.com/
strategyawards2014/2014-gallery
22:22:38,437 ERROR zw.co.WebCrawling.PLSAWebCrawler - link has been accessed before or is not validhttp://
videos.airbus.com/video/e1950d4e23as.html
22:22:38,437 ERROR zw.co.WebCrawling.PLSAWebCrawler - link has been accessed before or is not validhttp://
videos.airbus.com/video/e1950d4e23as.html
22:22:38,437 ERROR zw.co.WebCrawling.PLSAWebCrawler - link has been accessed before or is not validhttp://
videos.airbus.com/video/963fcab2db7s.html
22:22:38,437 ERROR zw.co.WebCrawling.PLSAWebCrawler - link has been accessed before or is not validhttp://
videos.airbus.com/video/963fcab2db7s.html
```

5.2.6 Downloading Web Url

After validating the web url , the crawl downloads the web page using HTTP protocol. To mitigate slow responses the research implemented connection timeouts of 5000 milliseconds and read timeouts of 5000 milliseconds as well to each URL the crawling program tired to download. Only URLs, which returned HttpURLConnection

of (OK)200 where downloaded by the research-downloading program. To mitigate also overloading the web server performance potentially caused by high frequency of the download, the research put a delay of 3 seconds between each download. Once connection was successfully negotiated, the crawler opened a stream connection and downloaded the pages storing them into a temporary buffer java object(String Buffer). The String Buffer Reader and writer objects were proffered to the conventional String Class due to their thread-safe, mutability on a sequence of characters and ability to handle multiple threads characteristics. Due to high number of concatenations the String buffer is used as opposed to the regular String object since it faster and also the length and content of the String Buffer can be changed as the method is called. The Buffered Reader java class is also used to read the downloaded web files and extracting the content. A maximum crawling size for each seed or leading URL was set at program level and this was set to hundred to prevent overloading the server.

Removing html tags

Once the file is downloaded it is pre-processed through first extracting the html tags and in this case the Jsoup java package is used to extract only tags and leave out html tags in the web page. The time to download is also measured in nanoseconds for future analysis of download time and also any URL that failed to download threw a FileNotFoundException, URISyntaxException as well IOException and all web urls were were ignored. Most of the URLs that failed with URISyntaxException were either due to the network failing due to speed, security prohibitions or being a dead URL link on immobile websites.

Preventing downloading of same web page

The research implemented a database storing all the web pages already downloaded and before a new web page was saved on the database the research checked using SQL whether the URL was not already in the database as a result of an earlier download and in such a case the download would be skipped by using the continue keyword in java. The research made sure it was critical that no single failure of one web page stopped downloading of the next web page through handling of java exceptions listed in the previous paragraph. Even after downloading the content and the content was found to have zero length then that page was discarded as well.

The research also used getHashValue to encode each downloaded page and on downloading another page the hashcode of the page was compared against the hash codes of all their webpages on the system and if there was a match then the page was deemed to have been downloaded before. This was to prevent pages having the same contents as well.

5.2.7 Basic Detection of web crawled pages' language using JSON API and REST API

Each page downloaded had its contents validated to see if it was English or another language. This as done by implementation of a REST(Representation State Transfer) API web service call to a publicly available web language detector web site[12]. The web service implementation done in java accepted a Java String Object Notation(JSON) format string and returned an indicator of the language to approximate levels of probability and confidence interval

if it was English denoted by 'en' or another language. A disclaimer on their website indicate that they detect languages varying to 164 languages hence this was considered as a good level of tool to use to check rather than implementing the research's own language detector in java in the given timeframe. The REST API call done using either a GET or POST method for larger requests[12] is done to the website http://ws.detectlanguage.com/0.2/detect The results are returned in JSON format API producing results in JSON format as an array of language candidates with the one with the largest confidence interval being at the top Each JSON response object will consist of language code, reliability clause true or false as well as the confidence score. A partial implementation using jlanguagedetect public java library was also done but the former was prefered due to the fact that it covered more languages and was well proven in the market. The bottleneck however is that it introduced performance bottlenecks during downloading as well since this was a web service call as well. The research did store the language against the downloaded web page but did also consider and words that came out as non- English since the risk was minimised by choosing only English seed urls.

```
public static String guessLunguageWithConfidenceInterval(String detect)
   {
     detect=detect.replace(' ', '+');
   String url12="http://ws.detectlanguage.com/0.2/detect?q[]="+detect+"&key=demo";
   String result=new GuessLunguageOfCrawledWebsite().sendPostRequest(url12);
   return
        result;
   }
```

5.3 Web Crawling Pre-processing

Pre-processing of web pages involves removal of stop words. In total 670 stop words are not considered in this research. The research also disregards all the words less than 3 letters in length as well as removes all non-alphabetical characters from the word. This how ever has an impact of changing the meaning of the word Pre-processing of the web crawled data include removing stop words, html links and getting rid of all non-html files. The crawled data is rid of all common words, which number to around more than 670 words and these include, 'or', 'the', 'is'. Also excluded included are common words on websites that include 'terms', 'conditions', 'home'. The pre-processing also excludes all files of type, 'jpg', 'jpeg', 'png' and other non-text files since they are not required for the text processing stage. To ensure the same word with different case is not repeated the pre-processing also converts every word to lower case as well. The pre-processing stage also removes also characters like brackets, commas, penning and closing quotes as well as apostrophe among the following characters '',','','':','!',';'.

```
   public static String removeCommonWords(String wpage)
      {
          StringBuilder result = new StringBuilder(wpage.length());
        for (String sm : wpage.split("\\b")) {
```

```
            if (!BasicCrawlerUtilities.isCommonWord(sm))
              result.append(sm);
        }

        String cleanstring=result.toString();
        Jsoup.clean(cleanstring, Whitelist.none()) ;
        return cleanstring.toString();
      }
```

5.3.1 Cleaning up words

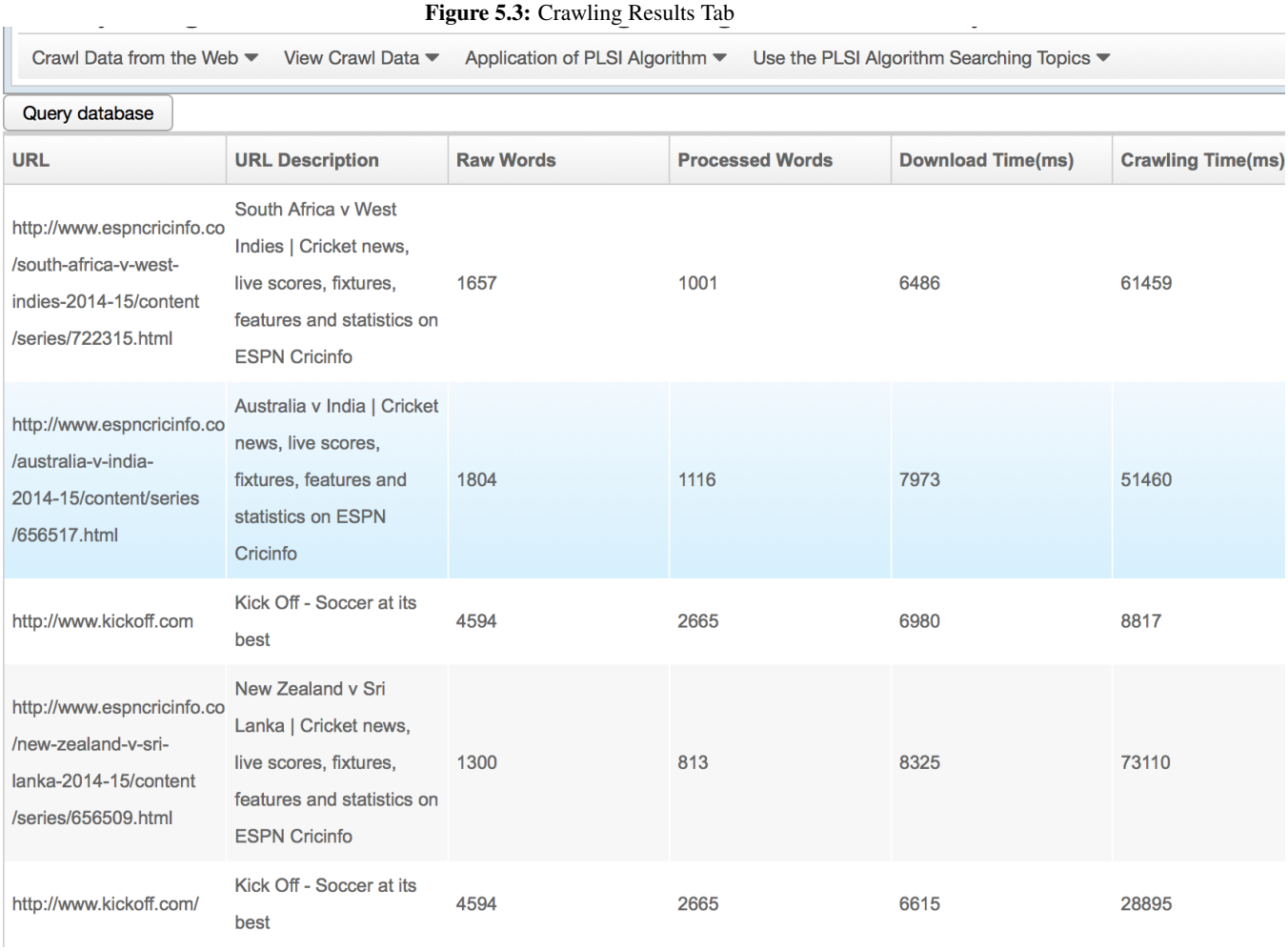
Cleaning up of words involved removing stop words some of which are shown below and this saved processing time as well as reduce unnecessary words from being considered in the Document Term Matrix. The trade-off however is loosing the sense of phrases that give a meaning and for instance removing the word first word in this phrase, 'San Antonio loses the meaning of the word, which is a city in America. The research also ignored all words less than 3 letters but this might lose meaningful words like 'war' and also removed string operators like 'ór which means words like isn't will be isnt. The java .toLowerCase() String operator is used to convert all the crawled data into lower case letters hence reduce the comparison problems. Also a java program is used to check if the words are English or not , thought the non-English words are not removed from the set. The stemming algorithm does not leave a real word after removing the stem.

```
public static String[] stopWordsofwordnet2 = { "password", "username",
        "designed", "day", "advertise", "privacy", "policy", "facebook",
        "twitter", "LinkedIn", "previous", "terms", "conditions",
        "copyright", "rights", "reserved", "without", "see", " ",
        "us", "sign", "Google", "click", "otherwise", "click", "languages",
        "forgot", "sign", "'", "English", "search", "browser", "cookies",
        "register", "home", "login", "logout", "id", "email", "news",
        "try", "site", "go", "email", "unless", "due", "also", "must",
        "might", "like", "couldn't", "did", "didn't", "do", "does",
        "doesn't", "doing", "don't", "down", "during", "each", "few",
        "for", "from", "further", "had", "hadn't", "has", "hasn't", "have",
        "haven't", "having", "he", "he'd", "he'll", "he's", "her", "here",
        "here's", "hers", "herself", "him", "himself", "his", "how",
        "how's", "i ", " i", "i'd", "i'll", "i'm", "i've", "if", "in",
        "into", "is", "isn't", "it", "it's", "its", "itself", "let's",
        "me", "more", "most", "mustn't", "my", "myself", "no", "nor",
        "not", "of", "off", "on", "once", "only", "ought", "our", "ours",
        "ourselves", "out", "over", "own", "same", "shan't", "she",
```

5.3.2 Preprocessing results

Evidence of crawling is in the following screenshot taken during crawling processing of Batch 1 seed url www.espnfc.com . The processed words show that they have hugely less values than the original words, which indicate that stop words and other invalid non-alphabetical words have also been removed.

Figure 5.3: Crawling_Results Tab



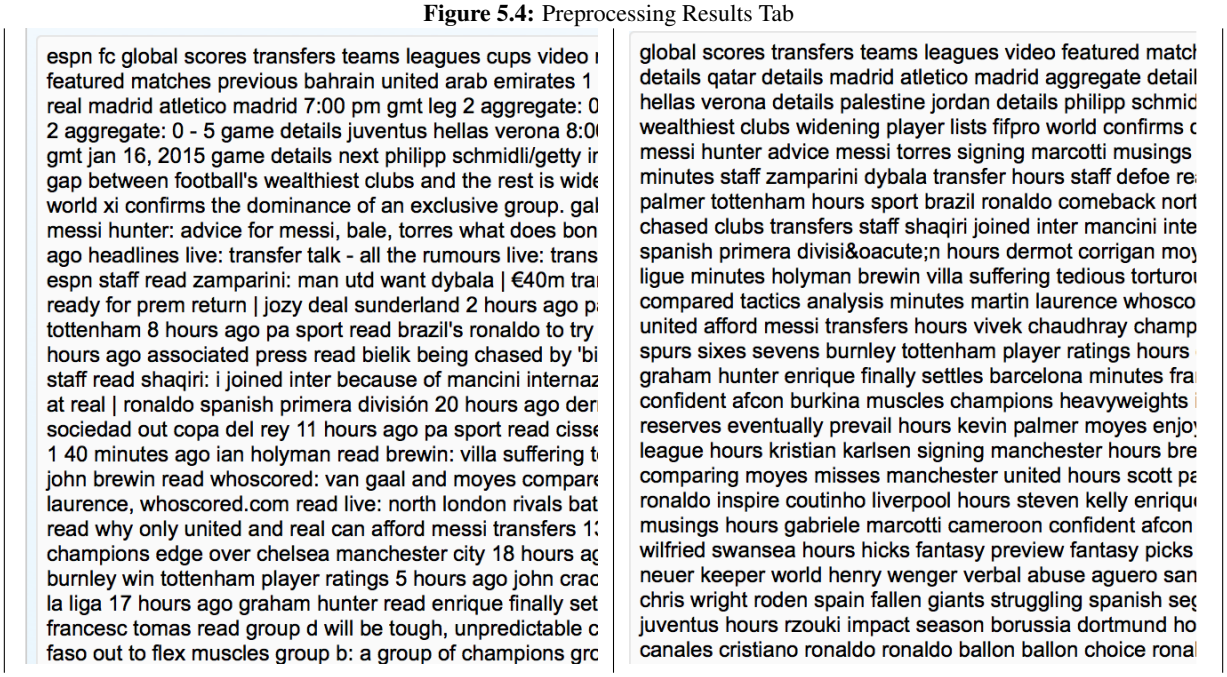
URL	URL Description	Raw Words	Processed Words	Download Time(ms)	Crawling Time(ms)
http://www.espncricinfo.co /south-africa-v-west-indies-2014-15/content /series/722315.html	South Africa v West Indies | Cricket news, live scores, fixtures, features and statistics on ESPN Cricinfo	1657	1001	6486	61459
http://www.espncricinfo.co /australia-v-india-2014-15/content/series /656517.html	Australia v India | Cricket news, live scores, fixtures, features and statistics on ESPN Cricinfo	1804	1116	7973	51460
http://www.kickoff.com	Kick Off - Soccer at its best	4594	2665	6980	8817
http://www.espncricinfo.co /new-zealand-v-sri-lanka-2014-15/content /series/656509.html	New Zealand v Sri Lanka | Cricket news, live scores, fixtures, features and statistics on ESPN Cricinfo	1300	813	8325	73110
http://www.kickoff.com/	Kick Off - Soccer at its best	4594	2665	6615	28895

5.3.3 Preprocessing File Snippets

The snippets below show the blob objects printout for both pre-processed and post processed files. The pre-processing successfully removed digits, characters with length less than four letters as well as trimming all letters of non alphabetical characters which the program supplied. One of the challenges the program faced it did not cater for all the possible characters coming from other languages like Chinese or Japanese or Arabic, which do introduce other characters, the program has not understood since it catered for mostly English words. The dilemma faced is that if the crawling program had chosen to recognise alphabetical characters rather than exclude non-alphabetical

characters then there was big risk of excluding people and place names since most of them are not English names hence the program chose not to explore that option. To cater for abbreviations the program created another list of all the abbreviations and added them to the list of the pre and post processed file after all the letters had been processed. Thus a word LSA for Latent Semantic Analysis, which would have not met the length criteria are now able to be in the word list. The program also resolved to covert all words to lower-case in order to avoid having the same duplicate word considered differently. The crawling implementation also ignored words with length greater than 20 characters since it deemed them not worthy contributing to the PLSA text mining program analysis.

Figure 5.4: Preprocessing Results Tab



5.3.4 Save Web Page Record

After the web page download was done, the research then saved it as a temporary file in a folder on the processing machine. The processing also produced another processed file after removing stop words. A record was then created on the database detailing from the number of words, both files, time taken to crawl and download time as well as the start end time of both activities. Both files were then converted into database binary large object, blob format for storage in binary format on the database. A frontend using the java ZK framework running initially on the tomcat version 7 server then on the Jboss server was used to showcase this data in a friendly format using Java Enterprise Edition standards. The research utilised java beans to store the data in flight first before counting words. The research also saved all the pages from the same url or urls in the case of multithreading batches to a batch number. This then enabled selective selection of documents by batch number when running the PLSA algorithm and also generating the document term matrix.

5.4 Storing of Crawled Data

To mitigate risk of saving same web page as a blob object in the database, the research used near duplicate detection and check sums to prevent such occurrence. The algorithm prevented also storing of copyright information. The navigational links are traversed using a DOM or Document Object Model tree. The following fields were chosen as the data to download

Table 5.2: Crawling Batch Run Table

Gather Crawling Data		
Category	**frequency**	**seed url**
tennis	101	http://espn.go.com/tennis/
general news	51	http://www.bbc.co.uk/news/world/africa/
general news	1	http://www.bbc.com/news/world/africa/
finance	159	http://www.bloomberg.com
politics	51	http://www.cbsnews.com/politics/
cricket	197	http://www.cricbuzz.com
cricket	101	http://www.espncricinfo.com
soccer	100	http://www.espnfc.com
rugby	91	http://www.espnscrum.com
car racing	101	http://www.formula1.com
golf	199	http://www.golf.com/news
golf	8	http://www.golfchannel.com/news/
medicine	53	http://www.hopkinsmedicine.org/news/media/releases
medicine	100	http://www.medicalnewstoday.com/
politics	202	http://www.reuters.com/politics
space	98	http://www.space.com/news/

5.4.1 MySQL Crawling Tables

Two tables are used in crawling namely the CrawlingData and CrawlingBatchRun tables. The former tables gathers all the crawling data from the website. Key columns include the strUrl which shows the url of the web page downloaded, seedUrl which is the starting URL for all the fetched websites during a batch download as well as the refinedWebDoc and originalWebDoc, which are blob objects containing raw data of the web page and post processed text contents respectively. The choice of using a Blob rather than clob or character large object was arbitrary as both would have worked. Performance statistics are given by the startDownloadTime and endDownloadTime for deducing the download time taken by a website and 'startCrawlTime' and 'endCrawlTime' for deducing the total crawling time of the website. The crawling time in this case includes the total time taken to fetch the URL. Various database SQL queries are run to either deduce the total number of documents downloaded per original seed url. The 'numberOfWords' and 'wordsProcessedFile' indicate the original word count in the originalWebDoc file and refinedWebDoc(post processed file after removing stop words) respectively. The response code of downloading from the web is given as codes 404 meaning page not found, 200 for successful HTTP response and 401 for a page not authorised error and all these are used to populate the httpResponseCode field in the table. The language field indicates the language of the given downloaded article.

Table 5.3: Crawling Implementation Table

Crawling Implementation Table		
Column	**DataType**	**Description**
idwebdocs	int(11)	NOT NULL AUTOINCREMENTPRIMARY KEY
hashCode	varchar(255)	DEFAULT NULL
strUrl	varchar(255)	DEFAULT NULL Index idxcrawledWebSitesstrUrl strUrl
seedUrl	varchar(255)	DEFAULT NULL Index idxcrawledWebSitesseedUrl(seedUrl)
pageContents	varchar(255)	DEFAULT NULL
protocol	varchar(255)	DEFAULT NULL
httpResponseCode	varchar(255)	DEFAULT NULL
startDownloadTime	timestamp	NULL DEFAULT NULL
endDownloadTime	timestamp	NULL DEFAULT NULL Index idxcrawledWebSites startDownloadTime(startDownloadTime)
startCrawlTime	timestamp	NULL DEFAULT NULL
endCrawlTime	timestamp	NULL DEFAULT NULL
numberOfWords	int(11)	DEFAULT NULL
downloadTime	mediumtext	..
timeTaken	int(11)	DEFAULT NULL..
originalFileSize	mediumtext	.
processedFileSize	mediumtext	.
originalWebDoc	blob	.
refinedWebDoc	blob	.
wordsProcessedFile	int(11)	DEFAULT NULL
batchNumber	varchar(200)	DEFAULT NULL
language	varchar(200)	DEFAULT NULL

Figure 5.5: Crawled Data Table

h	strUrl	seedUrl
3	http://www.golf.com/photos/worst-and-best-golf-club-logos	http://www.golf.com/news
1	http://www.golf.com/photos/greg-norman-designs-bluffs-vietnam	http://www.golf.com/news
1	http://www.golf.com/photos/mexicos-wow-courses	http://www.golf.com/news
a	http://www.golf.com/photos/destination-golf-florida	http://www.golf.com/news
b	http://www.golf.com/photos/golfers-throwing-their-clubs	http://www.golf.com/news
c	http://www.golf.com/photos/ted-bishops-excellent-misadventures-pga-americ…	http://www.golf.com/news
e	http://www.golf.com/photos/best-quotes-week-cimb-classic	http://www.golf.com/news
4	http://www.golf.com/photos/awkward-photo-shoots-around-globe	http://www.golf.com/news
8	http://www.golf.com/photos/2014-wgc-hsbc-champions	http://www.golf.com/news
8	http://www.golf.com/photos/rory-mcilroy-career	http://www.golf.com/news
5	http://www.golf.com/photos/best-quotes-week-2014-hsbc-champions	http://www.golf.com/news
e	http://www.golf.com/photos/2014-ohl-classic-mayakoba	http://www.golf.com/news
2	http://www.golf.com/photos/2014-dp-world-tour-championship	http://www.golf.com/news
5	http://www.golf.com/photos/2014-golf-awards	http://www.golf.com/news
f.	https://www.youtube.com/user/GolfMagazine	http://www.golf.com/news
8	http://www.golf.com/2015/01/02/augusta-national-hikes-4-day-masters-badge…	http://www.golf.com/news
1	http://www.networkadvertising.org/	http://www.golf.com/news
0	http://www.AboutAds.info	http://www.golf.com/news
2	http://subscription-assets.timeinc.com/prod/assets/themes/magazines/default…	http://www.golf.com/news
b	http://www.golf.com/equipment/nike-lunar-control-3-golf-shoes-few-our-favori…	http://www.golf.com/news
6	http://www.golf.com/equipment/tiger-woods-has-new-nike-clubs-play-hero-w…	http://www.golf.com/news

Batch Runs

Since each download session involved running a batchrun and giving it a number, each test was also assigned to a batch. The test were run from three different mechanisms for crawling, namely from the java GUI ZK Frontend designed for this system, from running a java client program and also running a JUNIT test based on the JUNIT framework.

Table 5.4: Crawling Batch Run Table

Gather Crawling Data		
Column	**DataType**	**Description**
crawlingBatchRunId	int(11)	AI Primary Key
subjectMatter	varchar(255)	Will show the subject of the batch run, based on seed url
description	varchar(200)	
batchNumber	varchar(255)	Batch number of the downloaded urls fetched from given seed url
numberOfDocs	int(11)	number of documents downloaded from fetched urls
startTimeCrawling	timestamp	start time of downloaded batch
endTimeCrawling	timestamp	end time of downloaded batch
timeTakenCrawling	double	time taken in milliseconds download the crawl the whole batch
timeTaken	double	Time taken to download all the web pages for the whole batch excluding pre-processing time

The batch crawling table will detail all the crawling batches run by the research. The description shows the type of batch download whether its a multithreaded java mode or single thread with the former downloading multiple pages at the same time hence having more than one seed url. The subject matter will detail the subject of the seed url(s) used in this batch download. The time taken is measured by the startTimeCrawling and endTimeCrawling of which the difference of the two shows the time taken to run the batch(timeTaken). A summary of the downloaded data is shown below

5.5 Crawling Exception Handling and Logging

5.5.1 Logging during crawling

The research used log4j to replace System.out standard java logger during both crawling and as well as the analysis of the PLSA algorithm as indicated in the next chapter.

Table 5.5: Logging during crawling

Column	**Description**
Configurability	The research was able to turn the logs off and on as performance required. When the research encountered performance bottlenecks logs were switched off.
Maintainability	The research was able to name logs appropriately with crawling logs being prefixed by crawling and PLSA implementation algorithm also prefixed by PLSA.
Granularity	Each class was configured with each own logs and different statements were logged to different logs for instance a log was created for all rejected urls
Utility and File Appenders:	The research created two log files java appenders namely LogCrawlingFileAppender and LogPLSAFileAppender to name log files for different scenarios

5.5.2 Monitoring Execution

The research used AspectJ for monitoring and logging starting and ending time of each. AspectJ was also used to track how many websites were downloaded, how much time was spent to download as well as frequency of execution. The research also prioritised using nanoTime() instead of currentTimeMillis() for measuring time differences

since the latter proved to have inaccuracies with small differences when time differences were very small. All the methods used in the Crawling and pre-processing of the research were grouped under one aspect program and the following join points were created as shown below. For more on the crawling aspectJ implementation please refer to Appendix One. The research focused on monitoring the fetching or the crawling, guessing of the web page language(calling a web service) using a java REST API technology. The pre-processing and the inserting of the web document into the MySQL database was also monitored and logged using log4J. Also the collation of the cumulative frequency of the time taken to download as total crawling time was done.

Table 5.6: Methods for AspectJ and Log4J

Gather Crawling Data using AspectJ and Log4J		
Join	**Join**	**Execution**
before(),after()	startfetchPLSACrawledPageContent()	execution(* zw.co.WebCrawling.PLSAWebCrawler. fetchPLSACrawledPageContent(..)).
before(),after()	guessLunguageWithConfidenceInterval()	execution(* zw.co.WebCrawling. GuessLunguageOfCrawledWebsite.guessLunguageWithConfidenceInterval(..)).
before(),after()	extractWebCrawledLinks()	execution(* zw.co.WebCrawling.PLSAWebCrawler .extractWebCrawledLinks(..)).
before(),after()	preprocessAndWriteToFile()	execution(* zw.co.WebCrawling. PLSAWebCrawler.preprocessAndWriteToFile(..)).
before(),after()	insertWebDocumentIntoDB()	execution(* zw.co.WebCrawling. PLSAWebCrawler.insertWebDocumentIntoDB(..)).

The following shows the results of logging done using AspectJ and log4J framework indicating the methods' execution time. The start and end time of each execution is shown with extracting the web links being the most expensive operation in the crawling process followed by the processing of saving web documents into the database.

Table 5.7: Gather Crawling Data using AspectJ and Log4J

12:28:07,086 WARN zw.co.exceptionlogging.AspectJAspect - extractWebCrawledLinks for website has started:
12:28:07,097 WARN zw.co.exceptionlogging.AspectJAspect - extractWebCrawledLinks for website has ended: 11 milliseconds
12:28:07,386 WARN zw.co.exceptionlogging.AspectJAspect - extractWebCrawledLinks for website has started:
12:28:09,732 WARN zw.co.exceptionlogging.AspectJAspect - insertWebDocumentIntoDB for website has started:
12:28:09,825 WARN zw.co.exceptionlogging.AspectJAspect - guessLunguageWithConfidenceInterval for website has started:
12:28:09,832 WARN zw.co.exceptionlogging.AspectJAspect - guessLunguageWithConfidenceInterval for website has ended: 6 milliseconds
12:28:09,850 WARN zw.co.exceptionlogging.AspectJAspect - insertWebDocumentIntoDB for website has ended: 117 milliseconds and 200pages saved
12:28:09,858 WARN zw.co.exceptionlogging.AspectJAspect - extractWebCrawledLinks for website has ended: 2471 milliseconds
12:28:13,276 WARN zw.co.exceptionlogging.AspectJAspect - fetchPLSACrawledPageContent for website has started:
12:28:14,561 WARN zw.co.exceptionlogging.AspectJAspect - startfetchPLSACrawledPageContent for website has ended: 1284 milliseconds
12:28:14,572 WARN zw.co.exceptionlogging.AspectJAspect - extractWebCrawledLinks for website has started:
12:28:15,130 WARN zw.co.exceptionlogging.AspectJAspect - extractWebCrawledLinks for website has ended: 558 milliseconds
12:28:20,085 WARN zw.co.exceptionlogging.AspectJAspect - extractWebCrawledLinks for website has started:
12:28:33,950 WARN zw.co.exceptionlogging.AspectJAspect - insertWebDocumentIntoDB for website has started:
12:28:34,175 WARN zw.co.exceptionlogging.AspectJAspect - guessLunguageWithConfidenceInterval for website has started:
12:28:34,198 WARN zw.co.exceptionlogging.AspectJAspect - guessLunguageWithConfidenceInterval for website has ended: 22 milliseconds
12:28:34,208 WARN zw.co.exceptionlogging.AspectJAspect - insertWebDocumentIntoDB for website has ended: 258 milliseconds and 201pages saved
12:28:34,213 WARN zw.co.exceptionlogging.AspectJAspect - extractWebCrawledLinks for website has ended: 14127 milliseconds
12:28:37,654 WARN zw.co.exceptionlogging.AspectJAspect - Crawling for website has ended: 3719786 milliseconds

5.5.3 Exception handling

Some of the exceptions the research handled range from IOException and FileNotFoundException for all file handling processing. The research also encountered OutOfMemoryError errors as the program ran out of java heap size and this was mitigated by giving the program minimum and maximum java heap sizes of 2GB and 3GB respectively. Socket input stream errors as well as java database connection exceptions were also witnessed. Unknown hosts also returned socket exceptions as well as socket timeout exceptions in cases when the download speeds were very slow especially during midday. This was mainly mitigated by running downloads for the crawling program in the middle of the night. From the java frontend developed with ZK , which are jsp enhanced framework pages, the research had to do web frontend validations as well. Valid url inputs were validated using javascript if they were of the right format on the client browser hence saving the processing time to do validations of the seed urls. If the input was wrong the right message would also show to the use to enter the right web url. From the java client program a check of the url is done before processing and the user is given chances to enter the url before the program terminates.

5.6 Crawling using java frontend

Whilst most of the tests were initiated from the eclipse java programs triggered manually, the research also created a full Model View Controller framework to trigger downloads as well as view crawling data. The research also created pages to do multicrawling downloads as well.

5.6.1 Crawling using Multithreading

The crawling involved running crawling batches by specifying different number of websites as seed urls. The research used multithreading to speed up the crawling. The number of crawling batches numbered up to 3 and in total a total of 3600 documents were downloaded by the research of which 1000 documents were downloaded using multicrawling.

Figure 5.6: Single threading Crawler source:[18]

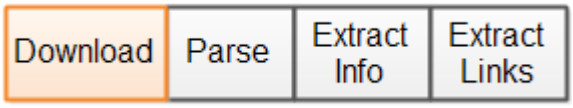
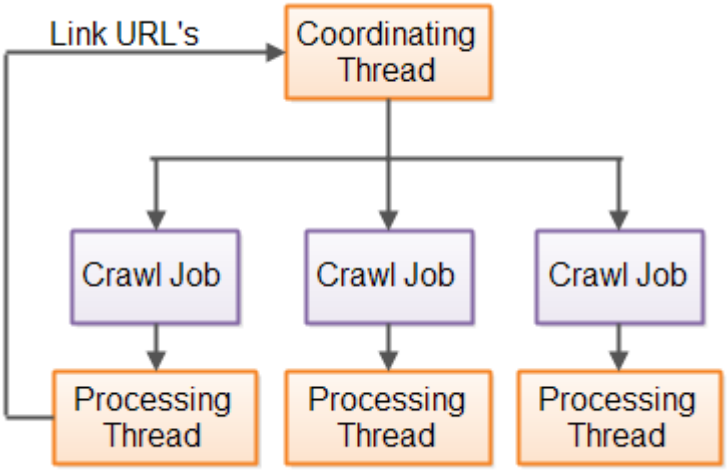


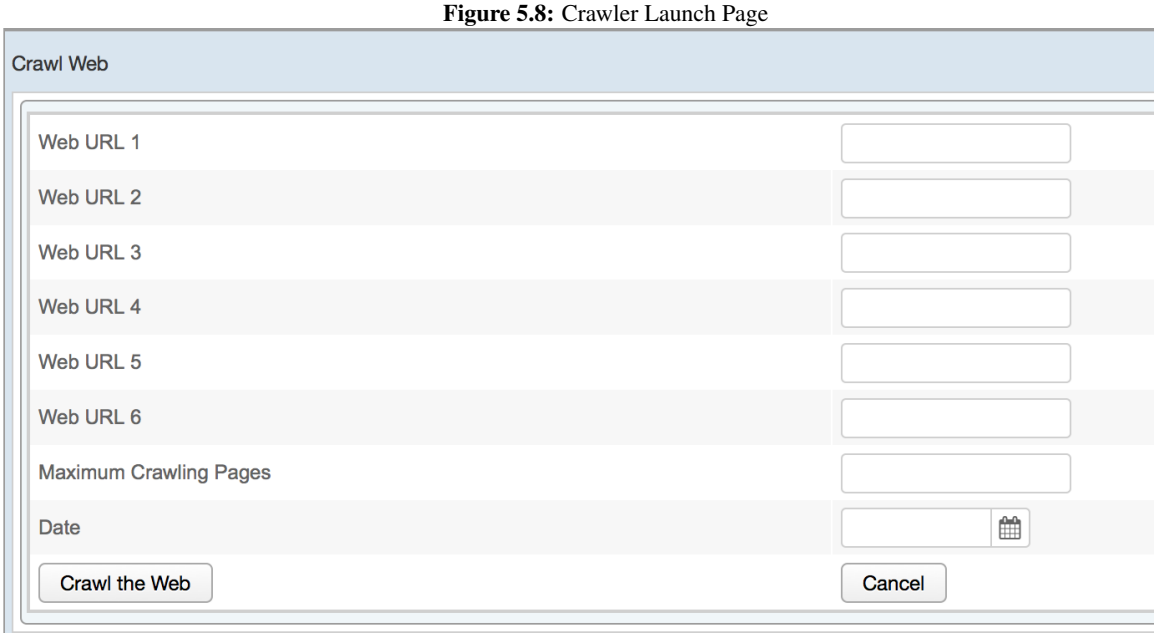
Figure 5.7: Multithreading Crawler source:[18]



The following websites were downloaded as part of a multithreading batch run

- http://www.cbsnews.com/politics/ -100 pages
- http://www.espnfc.com-100 pages
- http://www.espncricinfo.com -100 pages
- http://www.formulaone.com-100 pages
- www.reuters.com/politics 100...

Figure 5.8: Crawler Launch Page



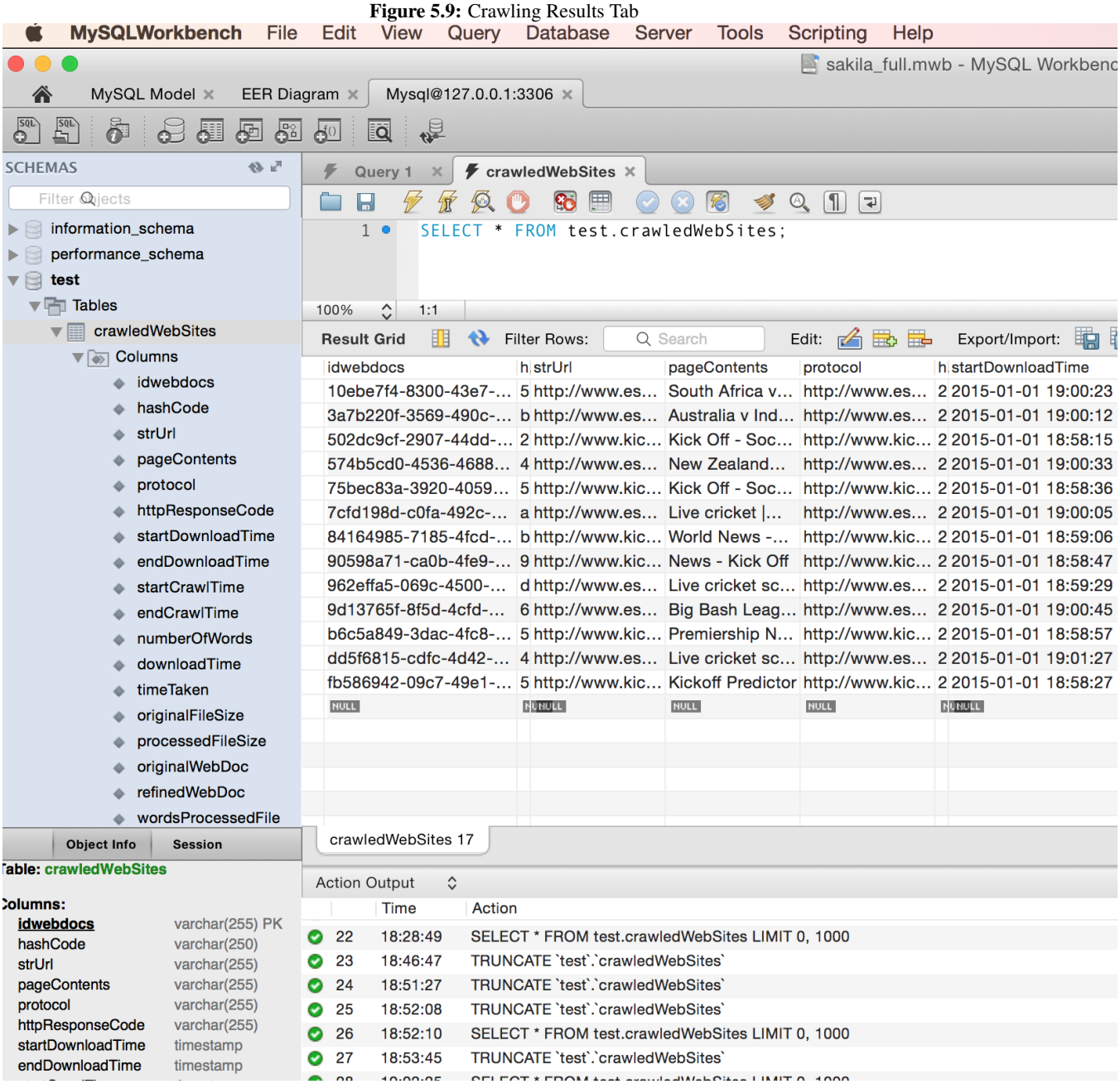
5.7 Crawling Results

The batches of crawling data that came out of this research included running the research as follows:

Table 5.8: Crawling Batch Tests

Batch 1	This batch involved downloading 100 pages on the espnfc.com website. No Multithreading was used in this scenario
Batch 2	This batch involved downloading 100 pages on the espncricinfo.com website. No Multithreading was used in this scenario
Batch 3	This batch involved downloading 100 pages on the formulaone.com website. No Multithreading was used in this scenario
Batch 4	This batch involved downloading 100 pages on the espn.com/nba website. No Multithreading was used in this scenario
Batch 5	This batch involved downloading 200 pages on the www.cricbuzz.comwebsite. No Multithreading was used in this scenario
Batch 6	This batch involved downloading 200 pages each on the talksport.com and espncricinfo.com website. Multithreading was used in this scenario
Batch 7	This batch involved downloading 1000 pages each on 5 websites with focus on soccer, cricket, politics, formula one racing and medicine. Multithreading was used in this scenario
Batch 8	This batch involved downloading 300 pages each from 3 websites ,still using the same topic categories above. Multithreading was used in this scenario
Batch 7	Save Web Page Record method
Batch 7	Analysis and Conclusions

Figure 5.9: Crawling Results Tab



5.7.1 Preprocessing Results

Figure 5.10: preprocessedFile Results Tab

markets share prices currencies commodities gilts & bonds uk earnings nasdaq index 15 min delay previous close value *all charts show local time select time span for charts:one monththree monthstwelve monthsintra-day index value change % 52 wk-h 52 wk-l 4736.05 -41.39 -0.87 4806.91 3996.96 click name to view detailed share information top 5 winners value change % nephrogenex, inc. 13.35 +8.70 +187.10 cytokinetics inc. 8.01 +1.11 +16.09 velocityshares daily 2x vix short term etn 2.76 +0.35 +14.52 atossa genetics inc. 1.43 +0.14 +10.94 neonode inc 3.38 +0.30 +9.74 top 5 losers value change % velocityshares daily inverse vix short term etn 31.14 -2.82 -8.30 vimpelcom ltd ads 4.18 -0.25 -5.54 el pollo loco holdings, inc. 19.97 -0.85 -4.08 plug power inc. 3.00 -0.12 -3.85 gopro inc. 63.22 -2.18 -3.33 more index pages click name to view detailed information and chart global time (gmt) index value change % bbc global 30 thu 00:15 5063.32 0.00 0.00 europe / africa time (gmt) index value change % london ftse 100 wed 12:36 6566.09 +19.09 +0.29 ftse 250 wed 12:46 16085.44 +95.13 +0.59 ftse 350 wed 12:46 3595.28 +12.17 +0.34 ftse all share wed 12:46 3532.74 +11.52 +0.33 ftse techmark wed 12:36 3522.00 +16.33 +0.47 pan european ftseurofirst 300 thu 08:00 1368.52 +5.38 +0.39 dj eurostoxx 50 tue 16:38 3135.95 -49.22 -1.55 amsterdam aex wed 13:00 424.47 +2.88 +0.68 frankfurt dax tue 13:06 9805.55 -121.58 -1.22 mdax tue 13:06 16934.85 -95.18 -0.56 sdax tue 13:07 7186.21 -41.63 -0.58 tecdax tue 13:06 1371.36 +1.68 +0.12 paris cac 40 wed 13:00 4272.75 +27.21 +0.64 brussels bel 20 wed 13:00 3285.26 +8.85 +0.27 madrid ibex wed 13:03 10279.50 +0.30 +0.00 zurich smi tue 16:34 8983.37 -51.18 -0.57 spi thu 08:00 8857.03 -45.93 -0.52 moscow micex tue 16:01 1396.61 -36.33 -2.54 rts tue 16:01 790.71 -4.38 -0.55 johannesburg all share fri 15:01 49159.77 -294.82 -0.60 south asia / middle east time (gmt) index value change % bombay bse sensex thu 09:51 27507.54 +8.12 +0.03 karachi kse-100 wed 10:20 32111.11 +122.15 +0.38 lahore 25 wed 10:39 6121.84 +22.79 +0.37 colombo cse all share wed 09:20 7298.95 +7.62 +0.10 tehran tepix wed 11:21 68969.80 -284.70 -0.41 asia pacific time (gmt) index value change % sydney asx all ords wed 03:16 5388.60 -3.70 -0.07 tokyo nikkei 225 tue 06:45 17450.77 -279.07 -1.57 hong kong hang seng wed 04:06 23605.04 +103.94 +0.44 shanghai sse composite wed 07:50 3234.68 +68.86 +2.18 sse se 50 wed 07:50 2581.57 +57.94 +2.30 americas time (gmt) index value change % new york dow jones wed 21:01 17823.07 -160.00 -0.89 nasdaq wed 21:00 4736.05 -41.39 -0.87 chicago mercantile ex. s&p 500 wed 21:01 2058.90 -21.45 -1.03 sao paulo bovespa tue 20:56 50007.41 -586.41 -1.16 mexico ipc wed 22:30 43145.66 +126.91 +0.30 all market data carried by bbc news is provided by digitallook.com. the data is for your general information and enjoys indicative status only. neither the bbc nor digital look accept any responsibility for its accuracy or for any use to which it may be put. all share prices and market indexes delayed at least 15 minutes. 52 week high and low values are calculated from close price data. click here for terms and conditions top stories syria war 'killed 76,000' in 2014 china leader seeks shanghai answers thomas piketty rejects france award first airasia victim laid to rest brazil's rousseff takes new term oath features & analysis tolstoy's top tips the secret ingredients of a happy life - and they're 100 years old nazi legacy the new year's day tradition with a dark history in pictures how you captured the start to 2015 lithuanians take note mixed feelings as baltic state joins euro club most

Figure 5.11: Post Processed File Results Tab

01 January 2015, 22:06

experience device mobile local bbc sport weather earth future shop radio term market data africa asia australia europe latin america mid east canada business health sci environment tech entertainment video market data economy entrepreneurship business sport companies technology business knowledge economy jan updated gmt company market overview stock markets share prices currencies commodities gilts bonds earnings nasdaq min delay close charts local time select time span charts month three months twelve monthsintra change view detailed share information winners change nephrogenex cytokinetics velocityshares daily vix short term etn atossa genetics neonode losers change velocityshares daily inverse vix short term etn vimpelcom ads pollo loco holdings plug power gopro pages view detailed information chart global time gmt change bbc global thu europe africa time gmt change london ftse wed ftse wed ftse share wed ftse techmark wed pan european ftseurofirst thu eurostoxx tue amsterdam aex wed frankfurt dax tue mdax tue sdax tue tecdax tue paris cac wed brussels bel wed madrid ibex wed zurich smi tue spi thu moscow micex tue rts tue johannesburg share fri south asia middle east time gmt change bombay bse sensex thu karachi kse wed lahore wed colombo cse share wed tehran tepix wed asia pacific time gmt change sydney asx ords wed tokyo nikkei tue hong kong hang seng wed shanghai sse composite wed sse wed americas time gmt change york dow jones wed nasdaq wed chicago mercantile wed sao paulo bovespa tue mexico ipc wed market data carried bbc digitallook data information enjoys indicative status bbc digital accept responsibility accuracy share prices market indexes delayed minutes week values calculated close price data stories syria war killed china leader seeks shanghai answers thomas piketty rejects france award airasia victim laid rest brazil rousseff takes term oath features analysis tolstoy tips secret ingredients happy life nazi legacy tradition dark history pictures captured start lithuanians note mixed feelings baltic joins euro club popular shared secret happy life courtesy tolstoy world celebrations bring tradition dark history thomas piketty rejects france award huge crowds read thomas piketty rejects france award world celebrations bring people stabbed belfry secret happy life courtesy tolstoy china leader seeks shanghai answers deadly syria war tradition dark history airasia victim laid rest magna carta north korean leader proposes talks video audio fireworks globe watch drunks causing disruption watch killed stampede watch minute world watch inequality gap heading watch booked airasia flight watch guide watch brutal country heartland watch obama apologises wedding watch bbc guilty pleasures healthier happier embraced bit vice programmes talking movies watch tom brook movies world services mobile connected feeds alerts mail bbc editors blog bbc college journalism sources media action editorial guidelines bbc mobile bbc choices contact bbc parental guidance bbc bbc responsible external sites read page viewed date web sheets enabled view page current visual experience upgrading software enabling sheets

5.8 Crawling Performance

Crawling statistics show that the java heap was averaging around 50 megabyte and peaked to 75MB and for a processing server running on MacOS operation system and 16GB or RAM, this was considered satisfactory performance. CPU usage hovered around per cent peaking to 20 per cent at times. Number of java classes hovered around 3000 classes and threads open at any point in time average around 15. This was taken for a download of a batch of 200 documents but this observation was consistent across all batch downloads in this research.

5.8.1 CPU and Memory

The research focused on the websites in the following table and a summary of key statistics are laid in the table

Table 5.9: Performance of Crawling

Seed URL	Number of Documents	Description
aviationweek.com/	190	This is an aviation website
espn.go.com/boxing/	158	This is a boxing portal
espn.go.com/tennis/	101	Tennis website
www.bbc.co.uk/news/world/africa/	51	African news, politics and general
www.bbc.com/news/world/africa/	1	African news, politics and general
www.bloomberg.com	159	Finance and Investments
www.cbsnews.com/politics/	51	Political news
www.cricbuzz.com	197	Cricket news
www.espncricinfo.com	101	Cricket news
www.espnfc.com	268	Soccer news
www.espnscrum.com	208	Rugby news
www.flightglobal.com/news/aviation/	193	Aviation news
www.formula1.com	101	Car racing news
www.goal.com	197	Soccer news
www.golf.com/news	199	Golf news
www.golfchannel.com/news/	8	Golf news
www.hopkinsmedicine.org/news /media/releases	93	Medicinal news
www.medicalnewstoday.com/	98	Medicinal news
www.reuters.com/politics	202	Political news
www.space.com/news/	98	Space news

5.9 Findings

Some of the findings found in the table mean that due to fetching links in the table the research found out that a URL for soccer on document number 233, '1789', 'http://espn.go.com/boxing/', 'Louis van Gaal tight-lipped on Robin van Persie fitness ahead of Manchester Uniteds clash with QPR — Football News — ESPN.co.uk' which happened also to have the highest time taken to download and biggest file with 9977 bytes and 5898 bytes pre and post processed as well as post processed 796 words. This means the research ended up with url downloads of one category being downloaded in a different category though this could be because of the close relationship between www.espnfc.com and www.espn.com/boxing from the same base url. The lessons leant would never have to use the same url for two different categories since during the fetching phase the crawler will strip all and take out the

base url as its seed url. The research also found characters like ’ ’ which the research would not have included in its exclusion list hence the problem of the web being a multilingual centre proved a challenge for the research due to presence of such words. Urls like ”bet365 Online Sports Betting” sneaked in due to the inability of the research to filter all the web urls. With goal.com having many variants differing with countries the research should have opted to use www.goal.com/en instead which is the English version.

5.9.1 Crawling and Download Times

The research noted the huge number of rejected URLS with www.talksports batch download showing 4000 plus for rejected downloads. The average crawling time is 200 and average download it is 200. Highest crawling rates were seen on the www.espn.com/boxing seed url with values of 40ms whilst the fastest rate was 8ms. Espnfc.com has the highest average download time and cricbuzz.com has the lowest value. The website which the fetching algorithm fetched and downloaded pages at the fastest rate is bloomberg.com a financial data website and seashipnews.com and flightglobal.com an aviation information website had the worst website perfomances.

Figure 5.12: preprocessedFile Results Tab

description	b	n	startTimeCrawling	endTimeCrawling	t.	timeTaken	seedUrl	rejectedUrls
http://www.space.com/news/		1	2015-01-17 0…	NULL	N	980042	http://www.sp…	NULL
http://www.hopkinsmedicine.org/ne…		1	2015-01-17 0…	NULL	N	1003004	http://www.ho…	NULL
http://www.medicalnewstoday.com/		1	2015-01-17 0…	NULL	N	1104074	http://www.m…	NULL
http://espn.go.com/boxing/		2	2015-01-17 1…	NULL	N	2167965	http://espn.go…	NULL
http://www.goal.com		2	2015-01-17 1…	NULL	N	1384189	http://www.go…	NULL
http://www.flightglobal.com/news/a…		2	2015-01-17 2…	2015-01-17 2…	N	4332865	http://www.fli…	10044
http://www.seashipnews.com/		2	2015-01-17 2…	2015-01-17 2…	N	2595051	http://www.se…	4072
http://talksport.com/football		2	2015-01-17 2…	2015-01-18 0…	N	3109203	http://talkspor…	16186
NULL			NULL	NULL		NULL	NULL	NULL

Table 5.10: Perfomance of Crawling Downloads

AVG (number Of Words)	AVG (downloadTime)	AVG (timeTaken)	download Rate BytesperMs	AVG(original Filesize)	AVG(processed FileSize)	Docs	seedUrl
957.2789	4595.573684	1286761.69	1.396783404	6419.021053	4098.873684	190	www.aviationweek.com/
1211.6266	3799.202532	1155987.873	1.925185832	7314.170886	4446.594937	158	espn.go.com/boxing/
1787.2261	5948.63913	3651835.391	1.790562986	10651.41304	5886.195652	230	espn.go.com/nba/
1085.6337	2133.386139	377402.6535	3.112603958	6640.386139	3690.485149	101	espn.go.com/tennis/
664.3604	3298.279188	1409123.827	1.230186176	4057.497462	2487.451777	197	talksport.com/football
1001.098	2126.941176	183483.8235	2.974546896	6326.686275	3622.431373	51	www.bbc.co.uk /news/-world/africa/
1154	3092	4413	2.401358344	7425	4519	1	/www.bbc.com/news /world/africa/
1398.9245	1567.081761	452824.2579	6.160680029	9654.289308	6701.012579	159	www.bloomberg.com
1221	2019.078431	144791.8235	3.682848902	7435.960784	3958.196078	51	www.cbsnews.com/politics/
923.1675	2639.817259	7678482.751	1.988700956	5249.807107	2629.720812	197	www.cricbuzz.com
1777.6733	2878.60396	1907034.683	3.596658859	10353.35644	5532.336634	101	/www.espncricinfo.com
1400.1381	3627.022388	994620.0672	2.277272998	8259.720149	4739.481343	268	http://www.espnfc.com
572.4231	2003.461538	864229.875	1.769694279	3545.514423	2335.5	208	www.espnscrum.com
743.7461	5595.84456	2479181.078	0.887753496	4967.73057	3300.569948	193	www.flightglobal. com/news/aviation/
542.1485	1143.544554	238264.3663	2.845780879	3254.277228	1699.49505	101	http://www.formula1.com
1312.731	2498.664975	694960.0406	3.281906886	8200.385787	5548.654822	197	http://www.goal.com
1189.7236	2762.336683	1994288.633	2.475336772	6837.713568	3534.914573	199	www.golf.com/news
1379.5	2107.875	30769.125	3.689260511	7776.5	4793.5	8	www.golfchannel.com/news/
659.9247	3641.817204	541968	1.241463407	4521.182796	3274.215054	93	www.hopkinsmedicine.org /news/media/releases
3187.6224	6127.765306	544962.5102	3.818684109	23400	18544.38776	98	www.medicalnewstoday.com/
745.9257	1491.148515	663996.7525	3.595155571	5360.910891	3515.89604	202	www.reuters.com/politics
408.3969	4860.664948	1250784.18	0.582912058	2833.340206	1807.149485	194	www.seashipnews.com/
877.2143	4148.969388	505773.7653	1.331279713	5523.438776	3415.510204	98	www.space.com/news/

5.10 Analysis of Crawling implementation observed

Some of the challenges experienced during crawling include the fact that only http links were supported by the implementation. HTTPS, FTP and other web technologies are not catered for due to the security, SSL and login challenges as well as privacy concern. The research ran 10 batches of crawling downloads and managed to download a total of 3600 web pages.Stop words removal also followed. A total 5000 website rejections due to either the website timeout, duplicate link being downloaded and invalid web urls were encountered. The research utilised Log4J logger and AspectJ to monitor and log all the perfomance, frequency and execution of each method used in the crawling namely, the Crawling method, extractWebLinks method, fetchWebPAgeURL method, preprocessCrawledData method and the insertWebDocumentIntoDB method. These method were discussed in the previous chapter. Each crawled page was saved in the database with time statistics for each method and the time in milliseconds it took to execute including starting and end time of each method. Through the use of AspectJ and Log4J java logger, each batch crawling download gave an output of 4 files, namely a Failed Crawled pages output file, CrawlingTracing output file showing all the stages the crawling process, a CrawlingExceptionOutput file showing all the exceptions the crawling batch experienced during processing. The batch also outputted the total time of the whole crawling batch. As part of the crawling the data was saved in a MySQL database and this also includes the size of each downloaded file.

5.10.1 Preprocessing

Of all the 20 batches run, they all went into a processing of the data that involved removal of stop words which were put in the file as shown on 5.13, The pre-processing involved all the words with less than 3 characters excluding abbreviated words only. The pre-processing module also involved removal of all whitespaces and quotes and other special characters as well as trimming file over the given maximum file download size of 5 megabytes. Though this represented a risk of information loss it made the model easier to compute when the data was being mined in the PLSA algorithm. The performance analysis was done together as part of the crawling analysis and this involved the last two methods, namely insertWebDocumentIntoDB method as well as the preprocessCrawledData methods.

Figure 5.13: Sockets Tab

idwebd...	seedUrl	pageContents	timeTaken	originalFileSize	processedFileSize	wordsProcesse..
2598	http://www.flightglobal.com/news...	Eclipse starts 550 production in uncertain market -...	3228971	3572	2261	263
2597	http://www.flightglobal.com/news...	ANA 787s affected by defect in Trent 1000 gearbox...	3223170	4501	2594	320
2596	http://www.flightglobal.com/news...	Nations opposed to EU ETS agree on retaliatory me...	3186393	3202	2119	247
2595	http://www.flightglobal.com/news...	Business aircraft operators struggle with EU ETS pr...	3174190	7281	4289	505
2594	http://www.flightglobal.com/news...	Superjet passed pre-flight check before take-off: air...	3168052	3651	2369	286
2593	http://www.flightglobal.com/news...	SpaceX reusability trials coming soon - 3/29/2013 -...	3117826	3497	2293	279
2592	http://www.flightglobal.com/news...	China confirms test of “hypersonic missile delivery...	3111622	4371	2827	343
2591	http://www.flightglobal.com/news...	Unforgiving void: Spaceflight tragedies remembered...	3104183	11074	6742	845
2590	http://www.flightglobal.com/news...	Aerospace and Aviation News | Aviation Industry &...	3074703	10485	6392	769
2589	http://www.flightglobal.com/news...	Aerospace and Aviation News | Aviation Industry &...	3000086	10101	6294	753
2588	http://www.flightglobal.com/news...	NBAA 2014 | flightglobal.com	2991762	2008	1292	165
2587	http://www.flightglobal.com/news...	Big in Japan: Tokyo’s Top 10 aircraft projects - 10/2...	2985063	6596	4019	469
2586	http://www.flightglobal.com/news...	MEBA 2014 | flightglobal.com	2976438	1769	958	129
2585	http://www.flightglobal.com/news...	Aerospace and Aviation News | Aviation Industry &...	2943782	5800	3484	426
2584	http://www.flightglobal.com/news...	World Airforces Directory	2935476	10504	6212	753
2583	http://www.flightglobal.com/news...	FARNBOROUGH: Redesign transforms KC-390 into...	2929908	15540	8988	1058
2582	http://www.flightglobal.com/news...	ANALYSIS: Russian Helicopters looks to civil marke...	2921388	12869	7299	850
2581	http://www.flightglobal.com/news...	Airline Business Interview - Rusdi Kirana - Lion Air	2868760	1685	1007	121
2580	http://www.flightglobal.com/news...	Aerospace and Aviation News | Aviation Industry &...	2862577	5990	3595	431
2579	http://www.flightglobal.com/news...	?ALTA: Flightglobal insight report shows Latin mark...	2855809	3868	2512	307

5.10.2 Multithreading and Duplicate URL Elimination

The research implemented multithreading in the crawling program and this improved download times. This however caused overlaps where multiple threads or processes running in parallel downloaded same pages whilst not being aware that the website was being downloaded by the other thread. This was however mitigated by making sure that before every url as part of its validation was checked against the data already stored in the MySQL table with all the crawled data against the urlName column to see if it wasnt downloaded already and if so then the URL validation also failed. Since also the crawling was done with different batches to reduce performance and memory overheads on the MacOS machine used to do this research, the urls were also checked against previous batches as well to see if the data had not been downloaded already.

5.10.3 Privacy Concerns

Some of the websites failed to download if they were https and ftp urls since the security of theses websites was also blocked. This was mostly for web pages, which needed mostly captchas or needed username and passwords to login. All of these security-failed websites were not considered in the research. The risk also was on download company information from company websites and being used in the crawling. Even though there is no legislation which bars crawling on company data in South Africa this posed a risk, which was mitigated by the research

choosing only non-company websites as crawling websites. From an implementation point of view the research was also not equipped to be able to cater for this necessary requirement in the timeframe of the research taken. The research also mitigated the risk by checking against the robots.txt for all websites that were not allowed to be crawled and observed this.

5.10.4 Speed Control and Off peak crawling

The crawling did put huge perfomance loads especially on crawling smaller websites with less allocated bandwidth. For instance after a downloading pages on 2 South African websites www.redhotcleaning.co.za and www.kickoff.com, the crawling program started getting timeouts and a visit of the former website on the internet also showed very slow response times to show that it was being overloaded. To resolve this problem the crawling had a 2 seconds wait time between each crawling so as to throttle the web crawling download speed and reduce overloading the websites being crawled. Crawling where the high responses were done were done during 12pm midday which was considered a high volume and low internet speed time in South Africa hence future crawling batches were run at 4pm and afterwards to reduce network traffic loads and overworking the web servers hosting these websites. Visual VM and Jconsole, java-monitoring tools were also activated as well as logs were monitored during the crawling period to see if the network timeouts frequency increased.

5.10.5 Robustness

The research's target was to download huge amounts of data and there was need to ignore failed downloads on websites either giving slow download responses as well as timeouts on downloads. Websites like www.herald.co.zw and www.newzimbabwe.com, which the research wanted to use to crawl and mine text data on Zimbabwe news, gave very slow responses and hence the download was cancelled in order to avoid causing problems on those respective. This was largely due to the source of the web data being in areas with limited network and internet bandwidth.

5.10.6 Language Detection Web Service

The research's detection services involves checking if the web document's language is English or not. The web service not only adds to performance problems since it is a remote website calls to and fro on the server will severely damaged.

5.10.7 Denial of service

The design of the web pages in a site can result in unwanted denial of service. This is important because a web site may have a small number of pages but appear to a crawler as having a very large number of pages because of the way the links are embedded in the pages (an unintentional spider trap).

5.10.8 Downloading Time of web pages

Due to network challenges and size of some pages, some of the pages returned timeouts whilst downloading. Also the size of the downloaded pages had to be limited not to exceed 1 megabytes of data to avoid network download speeds as well as increased storage. The smallest page of the 3600 crawled documents was 20kb and the largest page was 800kb with the average being 50 kilobytes which is understandable since these are mostly text web pages.

5.10.9 Analysis of PLSA Crawled Preprocessing

The pre-processing of the data included filtering of unwanted data, removal of all stop words as well as removal of all words with less than 3 characters. After the pre-processing the data as stored on the MySQL database in Binary Large Object format for further processing by the PLSA algorithm.

Filtering html content

The crawled data was filtered of all the html tags by the java crawler and all html tags like ¡html¿, ¡div¿ were filtered. As part of this filtering including extracting only the textual data on that page hence all pictures and other multimedia data on the web page was also filtered as well.

MYSQL Data storage

The 'crawledWebData' was used to store all the data downloaded and some of the tables include the batch tables for processing of the data using the PLSA algorithm as detailed in the latter chapters of this research.

5.11 Filtering Irrelevant and Junk web pages

The research put a filter on the crawler to filter most of the popular social network sites like Facebook and twitter as well as any url found with an @ character or 'mail' characters as well. The research also filtered all the websites urls that had some of the following characters, 'png—tiff?—mid—mp2—mp3—mp4—wav' or 'rm—smil—wmv—swf—wma—zip—rar—gz' which means excluding any pictures, music downloads as well as zipped files as well. The research also excluded downloads from 'ftp' or file transfer protocol and other non (HTTP) hypertext transfer protocol. The research also ignored all pages that are found on all common websites like privacy pages, 'About Us' as well as 'Contact Us' pages by putting filter on the crawling program used as well.

5.11.1 Multimedia Sites and Social Network Sites

The web is evolving from just a huge textual exchange centre to now include massive amounts of multimedia and huge exchange of pictures and videos as supplements to the textual data. The research's focus being on mining and filtering textual data excluded all multimedia and picture data from consideration so as to simplify the model. The crawler implementation was also not equipped to read this multimedia data in the timeframe given to do this research. Thus a look at the list of log files with rejected websites there was a huge presence of multimedia links as well as picture links, which the algorithm rejected. All urls which contained mp3 or mp4 or swf as file extensions which are some of the major multimedia files on the internet were filtered and excluded from consideration in this research.

5.11.2 Social Network Sites and Blogs filtering

The crawling of the data was intended to be used for text data mining of mostly credible sources and official articles hence individual opinions and unofficial articles from social sites like Facebook and twitter as well as Instagram well filtered by filtering the URL in the crawling algorithm. The research mostly focused on getting news articles from credible and high reputation websites like www.bbc.com for politics and other news in general, www.espncricinfo.com for cricket news as well as www.formulaone.com for car racing news. Not only did the research avoid using social sites' unofficial language which general people use, but it also prevented mining text data from unofficial and unedited news or untrue web articles as well.

5.11.3 Advertising and explicit content web url links

Whilst the research tried to filter by putting some unsavoury words in the stop words and also filtering sites with such words, it did not able to filter illicit content nor did it filter advertising links found on most of the websites used in the crawling as the seed urls. The risk was mitigated by the choice of the seed or parent urls being reputable websites like www.bbc.co.uk and major websites.

5.12 Memory and CPU Bottlenecks

Due to bandwidth problems some of the websites failed to download due to timeout. Even though the research increased the processing timeout of each download, still 3 per cent of processed downloaded websites 3 600 failed to download in time hence losing potential raw data which could have been used during the processing. The MacOS machine used with 16GB RAM also ran out of memory on certain occasions as the java heap ran out. This was resolved by increasing the maximum and minimum heap sizes of the Eclipse and JBOSS application server used in this research to 4GB and 5GB each respectively.

5.13 Ethics observed during crawling

Some of the ethics this research observed include not flooding one particular website with requests hence the research only did one website and paused for 3 seconds and each seed url was limited to fetching only 100 pages in its parent pages only. The research also observed ROBOTS.txt by excluding all the pages indicated in the given file for all the particular websites. The research did a test to each website for 1 minute to see if the response of downloading was not overloading the website and if the results were not favourable the website was excluded from consideration. Some of the websites that showed negative responses include www.redhotcleaning.co.za and this was proven by visiting the website before, during and after crawling. The research also avoided going to company proprietary websites for fear of downloading confidential information accidentally exposed on the internet. Whilst there is no law governing crawling in South Africa, the research did its best to observe to the ethical guidelines governing crawling.

5.14 Conclusion

The chapter made a detailed outline of the crawling methodology used in this research. Core to this are the key steps in crawling namely fetching the web url, validating the web url, downloading the web url, pre-processing the file through removal of stop words and then saving the file and its time parameters for implementation of the PLSA algorithm to analyse the text in these documents. Various challenges ranged from timeouts during downloads, high number of rejected urls as well as database connection problems. The research outlined the 11 websites falling mostly in sports, politics, aviation and medicine. Brief analysis focusing on the performance of the crawler in downloading the excess of 3000 documents was done as was the analysis of the exception encountered and the pre-processing of the data.

Chapter 6

PLSA Algorithm

6.1 Introduction

The chapter details Probabilistic Latent Semantic Analysis(PLSA), a modelling tool for co-occurrence information under a probabilistic framework in order to discover the underlying semantic structure of the data. PLSA [**?**] characterizes the generation of co-occurrences of words and documents as a probabilistic mixture model or aspect model and associates unobserved latent variables. PLSA algorithm is used as an unsupervised learning algorithm that includes training the data, which is a document matrix from the set of crawled data of about 3500 documents. The PLSA algorithm after analysing the data, will assign probabilities to the documents and also the words by specific categories. Whilst the previous chapters detailed the successful work done using the PLSA algorithm as a machine learning tool to do clustering and classification and also the algorithm structure, the current chapter will detail implementation of the PLSA algorithm as unsupervised learning algorithm for text classification[15]. The crawled data derived from the classification detailed in the previous chapter is used as the source data for implementation. The chapter details pre-processing steps that include removing stop words and reading the documents and texts from a MySQL database. The data is then used to generate a Document Term Matrix before being implemented using the Expectation Maximization iterations until convergence is achieved. PLSA, which was proposed by Thomas Hofmann, boasts a solid statistical foundation and defines a proper generative model [15]. Automated document indexing is achieved through PLSAs usage of a generalized Expectation Maximization algorithm. A key strength of PLSA is its suitability for processing large data sets, as PLSA scales linearly with the number of data points and the number of latent classes. The chapter will detail the implementation of the PLSA algorithm and at end of chapter an analysis of the results of the Topic Term and Topic document matrices are then done to try to deduce the latent variables in the Document Term matrix generated from the crawling data detailed in the previous chapter.

6.2 Pre-Processing the Web Data

The web data obtained in the previous experiment is stored in a MySQL database once processing is done and grouped together for further analysis. Topic document and term matrices are the cornerstone of this EM algorithm hence these are the output of the PLSA algorithm. The data should have been removed of stop words and any illegal characters.

6.2.1 Data Description

The test data is a product of the crawling detailed in the previous chapter. A total of 3600 documents are used in the research and the number of post processed words being 200 000 excluding stop words. The processing and reading of the data takes a lot of processing time since as the Java program reads the database to create the document term matrix the Java heap and memory will cause gradual degradation of processing speed CPU as well as RAM due to the non-parallelization setup of the program. The data extraction is done using a Java program reading the crawled data from a MySQL database into a Java array-list. A total of 608 stop-words from the stop list is filtered out. The documents are each read from a binary large object entry in the database and consists of a sequence of arrays of words extracted from the web crawled documents less stop words and HTML tags. The research decided of doing stemming by removing the last 's' of each word but after careful consideration backed out this decisions for fear of losing meanings of lots of words. The research discards all words length greater than three and less than 20 letters only due to argument that a 2 letter word is mostly not a meaningful word for the given topic to be analysed.

6.2.2 Reading documents from MySQL Database

The research used the crawling data as its input data for the PLSA algorithm and the following table summarizes all the data the crawling program had downloaded on the respective websites used. The soccer website espnfc.com had the highest number of websites downloaded and the golf channel website and BBC Africa channel had the least but primarily because the websites downloads cancelled during poor Internet connectivity at the time. For this research the document term matrix is built by doing a selection of one website and its documents from the given rows or a combination of the rows hence different categories. The data is stored in the 'crawled Websites' table listed and discussed in the previous chapter and the data processed on each row comes from the 'refinedWebDoc' which is the blob object containing the post processed text data for the research.

Number of documents	Web URL crawled
190	http://aviationweek.com/
158	http://espn.go.com/boxing/
230	http://espn.go.com/nba/
101	http://espn.go.com/tennis/
197	http://talksport.com/football
51	http://www.bbc.co.uk/news/world/africa/
1	http://www.bbc.com/news/world/africa/
159	http://www.bloomberg.com
51	http://www.cbsnews.com/politics/
197	http://www.cricbuzz.com
101	http://www.espncricinfo.com
268	http://www.espnfc.com
208	http://www.espnscrum.com
193	http://www.flightglobal.com/news/aviation/
101	http://www.formula1.com
197	http://www.goal.com
199	http://www.golf.com/news
8	http://www.golfchannel.com/news/
93	http://www.hopkinsmedicine.org/news/media/releases
98	http://www.medicalnewstoday.com/
202	http://www.reuters.com/politics
194	http://www.seashipnews.com/
98	http://www.space.com/news/

Table 6.1: Documents crawled:PLSA Input Data

6.2.3 Supporting tables used

To implement the algorithm, the research used two key tables namely the PLSABatchRuns table showing all the tests batch runs done, either by varying the number of documents in the training PLSA implementation given the list in the previous table or by changing the values of k, the number of topics or the value of the likelihood comparator from the research default value of 0.000001. The PLSA batch run table indicates in summary the total time taken to run each step of the algorithm from generating the document term matrix, initialization to running the EM algorithm and then also the printing of the resultant matrices namely the topic-term and topic-document matrices.

Filed	Description
testrunId	varchar(255) PK
subjectMatter	varchar(255)
start_date	timestamp
end_date	timestamp
timeTaken	double
numberOfTopics	int(11)
numberOfDocuments	int(11)
startTimeEM	timestamp,indicating the start time of the EM algorithm for each test
endTimeEM	timestamp
timeTakenEM	double, this is the time taken in milliseconds
startTimePLSAInit	timestamp, indicates the start time in milliseconds
endTimePLSAInit	timestamp
timeTakenPLSAInit	double
vocabularySize	int(11)
startTimeDocTermMatrix	timestamp
endTimeDocTermMatrix	timestamp
timeTakenDocTermMatrix	timestamp
DocTermMatrixBlob	blob
logLikelihoodComparator	int(11)
javaHeapSize	varchar(45)
CPU	varchar(45)

Table 6.2: PLSA Batch Run Reports Table

6.2.4 Generate Vocabulary

To be able to build the document term matrix there is need to generate unique words in all the documents under consideration hence these are put in a Java set which avoids duplicate entries. The vocabulary size used in this research had 20 000 unique words considering the post processed data from web crawling. The vocabulary is implemented using the 'generatePLSAVocabularies' method listed in Appendix 1. The design is to use a Java set and assign all words that appear in a document in the set and prevent any duplicate into the set. The generation of the Document Term matrix is solely dependent on the vocabulary as this becomes the horizontal column and the document the row of the matrix. The following shows the vocabulary size for the tests carried out. The vocabulary will exclude any words with length less than 4 characters as they were not considered meaningful enough for this research unless it was an abbreviated word.

6.3 Implement the PLSA Pseudo code

This implementation required the word-document co-occurrence matrix, the number of latent classes and the number of iterations as input arguments. In the ideal case the PLSA algorithm would iterate until the log-likelihood of the PLSA model converges. As the convergence could go on forever the iterations approximate number was supplied for each test carried out by the research. The research varied the value of the iteration estimate as well. The

number of topics for each experiment was also varied and values ranged from 5 to 50 for documents ranging from 100 documents to 1000 documents. The research could not do any PLSA algorithm implementation for documents larger than 1000 documents due to memory limitations since the vocabulary and document term matrix became so big hence the Java heap size ran out. The number of latent classes k therefore was forced to take the values k=5, k=10, k=20 and k=50 and k=100. The research felt it unnecessarily complex to have latent classes greater than 100.

The algorithm implementation involved generating the Document Term matrix with the sentences from the data supplied in the MySQL database. Thus for each entry in the Document Term matrix denoted by DT(x,y) where x is the term frequency and y the document under consideration. The document term matrix took as its input the list of documents generated each with its own set of words appearing in that document. Thus the first part of the implementation involved the document term matrix generation and results from the experiment also indicate high memory and CPU requirements for this part of the process due to big Java objects generated. Any word whose frequency is zero on that document is then given a value of zero on the document term matrix.

The PLSA model was on various occasions trained on the generated document term matrix. Various tests carried out included varying values of k the topics, varying the log likelihood iteration estimate value which by default was set to 0.000001. Some test had to take a value of 0.00001 and 0.00000001 to be able to see if the number of iterations increased or not due to the change. The aim of the research was to be able to deduce the hidden or latent topics in each document or across documents denoted by letter z. The model then provided a representation of the documents in the form of probability representations of the form P(z—d) where z denotes number of latent topics and d, the number of documents. Thus this meant that the document represented a multiple number of topics.

After generating the Document Term matrix, the research implementation provided this as its input to the Expectation Maximization training algorithm. The algorithm began with the E-Step followed by the M-Step alternating between the two until convergence was achieved[10][23]. The EM algorithm's aim was to calculate posterior probabilities. After convergence the algorithm would then generate two key matrices namely Topic-Term matrix and Topic-Document matrix.

6.3.1 PLSA Algorithm

Data: Document and words terms
Result: Latent topics in the document structure
initialization ;
Step 1: Remove all stop words from the document;
Step 2: Build Term-Document Matrix ;
Step 3: Start training of the PLSA algorithm and initialize randomly ;

Step 4: Initialize probabilities P(z), P(d—z), P(w—z) randomly.
while *no convergence* **do**
    step a : E-step: Compute posterior probabilities based on the latent dataset variable. Missing values will be estimated by estimated parameter values as actual true values ;
    step b: M-step: Maximize the expected complete log-likelihood and estimate parameters using estimated values as the true observed values ;
end
Step 5: Print the Topic Term matrix;
Step 6: Print the Topic Document matrix;

Algorithm 7: PLSA Algorithm[**?**]

6.3.2 PLSA Methods implemented

As part of the Java implementation the following methods are used to implement these algorithm in java.

Method	Description
generatePLSAMatrixTable	This method generates the Document Term Matrix by reading the data from the database for web crawled data. It filters its searches by seed urls and can combine more than one seed url
initialiseProbabilities	This method will initialize all probabilities pz_d and pz_w using random probabilities
EM Expectation Step	The iterative method for estimating the likely probabilities and identify likely hidden topics. This will be the Expectation step
EM Maximization Step	The maximizing step of the Expectation Maximization algorithm. Will run for each iteration till convergence.
Normalization Step	The probabilities are normalized so that they fall between 0 and 1
calculateLogLikelihood	The maximum likelihood function will estimate the best log likelihood per each iteration till it converges

Table 6.3: EM Algorithm implementation

6.3.3 Step 1: Build Term-Document Matrix

The document term matrix is a big grid of data showing each document against the terms or words in the crawled dataset discussed in the previous chapter. In the document-term matrix the rows correspond to the documents and the columns or the y-axis to the terms. The terms form the semantic part of the PLSA algorithm documents and the

document is typically a unit of retrieval which in this scenario is one simple web page downloaded from the web and stored as a blob object. The research builds the matrix by extracting documents from the MySQL database and counting the frequency of each word in the vocabulary generated to build the matrix[10][23]. To simplify the document term matrix the research assumes that the pre-processing of the documents has been successfully done namely removing stop words, converting all the words to small case to reduce chances of it document being treated differently as well as removing non-alphabetical characters forming part of a word. An implementation of this algorithm is shown in Appendix 1 but the table below shows an extract from one of the matrices build during running of this algorithm.

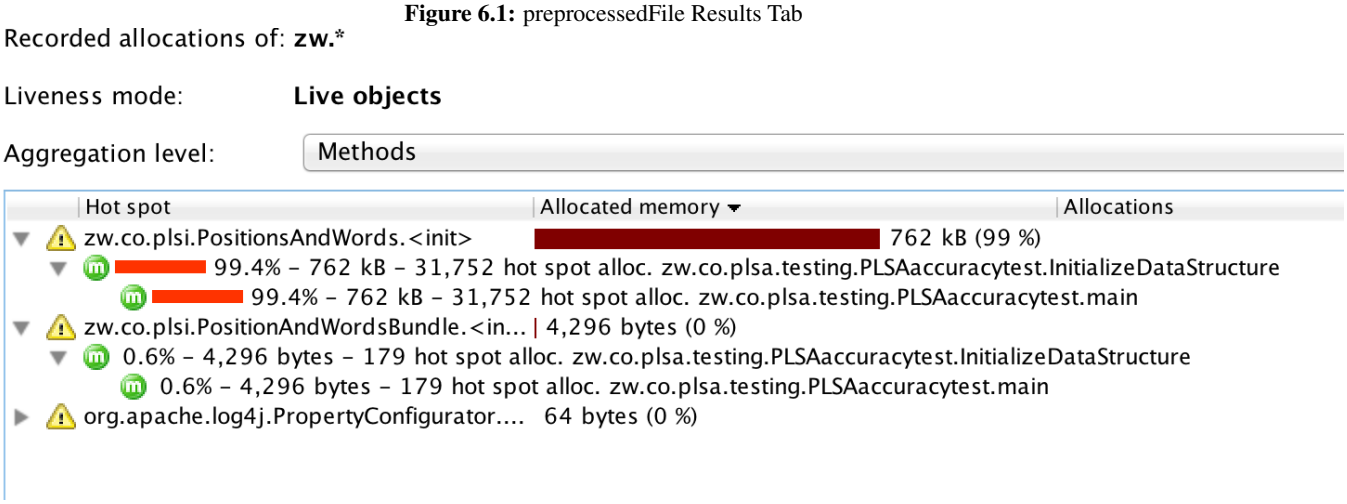
```
        t  t  t  t  t  t  t  t  t t  t  t  t  t  t  t  t  t
document 10 9  2  1  0  0  0 9  2  1  0  0  0 9  2  1  0  0  0
document 11 8  3  2  1  0  0 9  2  1  2  0  0 0  1  1  0  0  0
document 12 0  0  3  3  4  8 9  2  1  0  6  0 0  2  1  2  0  4
document 13 0  2  0  2  4  7 9  2  1  0  0  0 9  2  0  0  0  0
document 14 8  3  2  1  0  0 9  2  1  0  0  0 9  0  1  0  2  0
document 15 0  0  3  3  4  8 9  2  1  0  3  0 5  2  1  0  0  0
document 16 0  2  0  2  4  7 9  2  1  0  0  0 1  3  1  0  0  0
document 17 8  3  2  1  0  0 9  2  1  0  0  0 9  2  1  0  2  0
document 18 0  0  3  3  4  8 9  2  1  0  0  0 9  2  1  0  0  0
document 19 0  2  0  2  4  7 9  2  1  6  0  0 6  0  0  0  0  5
document 20 8  3  2  1  0  0 9  2  1  0  2  0 9  2  1  0  0  0
document 21 0  0  3  3  4  8 9  2  1  0  0  0 7  2  1  0  2  0
document 22 0  2  0  2  4  7 9  2  1  0  0  0 9  0  1  0  0  0
document 23 2  0  1  1  0  3 9  2  1  0  0  0 9  2  1  0  0  2
```

6.3.4 Step 2: Initialize probabilities P(z), P(d_z), P(w_z) randomly.

Of the two options of initializing the probabilities namely random probabilities or using a uniform distribution, the research opted to use random values to initialize these probabilities. Both the probability values P(z) for the hidden variable or latent topic, P(d—z) and that of the term given z the latent variable are initialized at random initially. The research found huge response times when running this phase of the process that is initializing the probabilities as shown by the VisualVM and JProfiler logs and graphs. The java utility, Math.random()is used to initialize the variables.

```
Pz_d[z][d] = (double) (rnd.nextInt(100)) / 100.0;
```

Figure 6.1: preprocessedFile Results Tab



6.3.5 Step 3: E-step: compute posterior probabilities for the latent variables

The e-Step or Estimation step will be used to compute by estimation posterior probabilities for given hidden or latent variables denoted by z. The following equation is used in this regard. The java implementation used for this part of the algorithm is shown in Appendix A. The step estimates the distribution of the latent variable by identifying each of the word[15][1]. word)

$$P(z_k|d_i,w_j) = \frac{P(w_j|z_k)P(z_k|d_i)}{\sum_{l=1}^{2} P(w_j|z_l)P(z_l|d_i)} \tag{6.1}$$

6.3.6 Step 4 : M-step: maximize the expected log- likelihood

In the m-Step, parameter estimation is done by re estimation of all parameters which maximize the Q-function. With this recalculated generation of parameters the E-Step is repeated and M-Step again until likelihood convergence occurs. This is where local maxima is reached. Guessing the latent variables then recalculating by approximation does the iterative data augmentation. The major assumption is that the greatest value is always the most complete value. The estimation will look at how often is term w associated with a latent variable z given by value P(w—z)[15][1]. Thus the step tries to maximize with respect to P(w_j—z_k) and total probabilities sum equals 1 for k=1 up to K. The M-Step will update the values of the posterior probabilities based on the formulas below. The implementation of this algorithm in java used in this research is shown in the Appendix 1

$$P(w_j|z_k) = \frac{\sum_i n(d_i,w_j)P(z_k|d_i,w_j)}{\sum_{m=1}^{M}\sum_{n=1}^{N} n(d_n,w_m)P(z_k|d_n,w_m)}$$
$$P(z_k|d_i) = \frac{\sum_{j=1}^{M} n(d_i,w_j)P(z_k|d_i,w_j)}{n(d_i)}$$
$$P(z) = \sum_{d=D}\sum_{w=W} n(d_i,w_j)P(z_k|d_i,w_j)$$
$$where --> n(d_i) = \sum_{j=1}^{M} n(d_i,w_j) \tag{6.2}$$

6.3.7 Calculate the log likelihood

The research infers hidden parameters of the PLSA algorithm using the Maximum Likelihood Estimator. The MLE finds one or more parameters for the given statistic where the likelihood distribution will be a maximum. In this scenario [15][1], the known variables are considered fixed and the selected function will focus on the variable changing frequently. The implementation of the log likelihood in java is shown in Appendix 1.

Figure 6.2: logLikelihood Tab

```
private double computeTheLoglikelihood(double[] Pz, double[][] Pz_d,
        double[][] Pz_w) {
    double L = 0.0;
    for (int d = 0; d < numberOfDocs; ++d) {
        for (PositionsAndWords pw : row1[d].getPosAndWords()) {
            double sum = 0.0;
            for (int z = 0; z < numberTopics; ++z) {
                sum += Pz[z] * Pz_d[z][d] * Pz_w[z][pw.position];
            }
            L += pw.nWords * Math.log10(sum);

        }
    }
    return L;
}
```

6.3.8 Normalize the data

Normalization of the data using the statistical model will involve transforming the data into values between 0 and 1 with mean being 0 and standard deviation 1. The algorithm involves rescaling all the probabilities by the minimum and range to make all the elements lie between o and 1. The implementation of the EM normalization is shown in the appendix section of the report.

```
for (int z = 0; z < numberOfTopics; ++z) {
          normValue += Pz_w[z][w];
      }
      if (normValue <= 0.0)
```

```
            normValue = 1.0;
        for (int z = 0; z < numberOfTopics; ++z) {
            Pz_w[z][w] /= normValue;
        }
```

6.3.9 Step 5:Generating the Topic Document Matrix generated

The topic document matrix is generated during the execution of the EM algorithm and the estimated values of Pz_d the probabilities of z the latent variables given a document d[15][1]. This data is stored in the TopicDocument table for processing. This is then printed to a text file using log4j. The document term matrix also selects the top 20 documents that have the highest probability Pz_d. The filter used is both topic and batch run. The sort order is done on the probability with descending order. The java implementation of generating the topic document matrix is shown in Appendix A.

```
SELECT topic,pz_w,term FROM test.TopicTermMatrix where
    testrunId='093e6427-14d6-4226-8abf-8d80649696cc0' and topic=3 ORDER BY pz_w DESC limit 30
```

Field	Description
Documentmatrix_id	int(11) AI PK
subject	varchar(255)
start_date	timestamp
end_date	timestamp
testrunId	varchar(255)
pz_d	double
topic	int(11)
documentId	varchar(255)
description	varchar(200)

Table 6.4: PLSA Topic Document Table

6.3.10 Step 6: Generating the Topic Term Matrix generated

The topic term matrix, after being computed from the PLSA algorithm is stored in a MySQL table called TopicTermMatrix table. The research implemented a printout of the topic matrix by selecting the top 30 words with highest probability for each given topic. The search criteria is both topic and batch run number and this returns the data in two columns namely the probability given and also the term with that probability. The sort order is done on the probability with descending order. The matrix variables returned by the search are the probability z given w pz_w as well as the respective term. By sorting in descending order of probability, the results will indicate the most likely words to be in the topic.

Field	Description
topictermmatrix_id	int(11) AI PK
subjectTitle	varchar(255)
start_date	timestamp
end_date	timestamp
testrunId	varchar(255)
pz_w	double
topic	int(11)
term	varchar(255)
description	varchar(255)

Table 6.5: PLSA Topic Term Table

6.4 Experimental Results and data generation

6.4.1 Training the data

Once the data was stored in MySQL database, the second milestone involved analysing this data using the PLSA algorithm by first building a document term matrix using the documents downloaded by the crawling method and then using this data to train it and identify latent topics in these documents. The number of topics to be identified , denoted k, was varied and performance and results varied. The algorithm implementation involved, 'buildDocumentTermMatrix' method, 'initialiseProbabilities' method, 'ExpectationMaximisation' method which iterated until convergence by calling the subroutines namely, 'Expectation and Maximization' as well as the 'calculateLogLikelihood' subroutines, till the latter's log likehood value converged to the given accuracy of 0.000001. The normalization of the resultant probabilities were then done using the 'normalizePLSAProbabilities' before the algorithm produced two resultant matrices, namely the TopicTerm matrix and TopicDocument matrix thus producing the resultant latent topics in the supplied data. Various tests were done and by varying the value of k, or the number of topics required, maximum number iterations(i) of the 'ExpectationMaximisation' method as well as the number of documents under consideration on each implementation of the PLSA algorithm. A total of 50 tests were run with varying values of k, i and nDocs, the number of documents. The following table shows response times for a selected implementation of the algorithm as well as the topics identified in that scenario where k took the value of 10 topics, and nDocs took the value of 300 documents. The other table also shows times taken to execute each method as part of the algorithm including highest time taken, average and number of iterations taken as well as the. The data the research is using will be selected using SQL from the following table showing all the data the research crawled.

6.4.2 Generating Topic Document and Topic Term Matrix

To generate the topic term matrix the research gets data after the EM algorithm has been run from the TopicTermMatrix table filtering by the topic number, testrunid for the batch of input documents that were involved in this PLSA algorithm run as well as returning pz_w and the value of the term from the database. The results will filter only the top 30 documents with highest probability of being in that topic. Thus for using value k=10 as the number of topics in a document list of 450 for the given test run 'ssss', the research will return all the 10 topics with their top 30 words as well as their corresponding probabilities. The first select statement gets the topic and probability of z given w or the term probability. The TopicTerm matrix will use the TopicTerm table to get all the probabilities and their corresponding words in descending order. The research opted to take only the first 30 words with the highest probabilities for each topic as shown.

```
SELECT topic,pz_d,documentId FROM test.TopicTermMatrix where
    testrunId='093e6427-14d6-4226-8abf-8d80649696cc0' and topic=3 ORDER BY pz_d DESC limit 30
```

The following table was a result of running the Expectation Maximization algorithm data on a dataset of 458 documents composed of 258 documents from the www.espnfc.com and the remainder from the cricket website

www.cricbuzz.com. The algorithm was running on topic of 10 and log likelihood comparator value of 0.000001. Out of the 10 topics found by the PLSA algorithm, notably topic 0 was found to have mostly English football teams with teams like Coventry and Ipswich being popular. Topic 2 is made up of mostly global soccer names like Brazilian footballer Ronaldinho and the French David Ginola as well as Michel Platini and the Dutch former great soccer player Johann Cruyff. Topic 4 is made up Spanish teams like Almeria and players like Benzema who plays(currently) for Real Madrid and Topic 6 is made up of Arsenal FC soccer plays like Rosicky, Chambers and Hector Belerrin and Fabianski who though has left the team this season. As the table [**?**] shows the highest probability entries are shown at the top.

topic 1	topic 2	topic 3
1 balancing(0.9999999999975403)	1 gocricket(0.9999999999999996)	1 customer(0.9999999999999999)
2 curiously(0.99999999999723)	2 parvinder(0.9999999999999993)	2 platini(0.9999999999999998)
3 combinations(0.9999999999950712)	3 purkayastha(0.9999999999999991)	3 essentially(0.9999999999999963)
4 wallace(0.9999999998627458)	4 appellate(0.9999999999999981)	4 masia(0.9999999999456267)
5 yorker(0.9999999962034808)	5 categories(0.9999999999999906)	5 faitelson(0.9999999997151485)
6 awkwardly(0.9999999960748266)	6 crore(0.9999999999999905)	6 repeating(0.9999999984283047)
7 underestimates(0.999999329469435)	7 taxable(0.9999999999999875)	7 conceivably(0.9999999984283047)
8 rovers(0.9864216782487725)	8 apprised(0.999999999999983)	8 distinctive(0.9999999972778529)
9 russell(0.9843068471294238)	9 cestat(0.9999999999999689)	9 flights(0.9999999819953493)
10 mcgeady(0.9772470686411642)	10 accrued(0.9999999999999646)	10 ironically(0.9999997733968599)
11 portsmouth(0.9560813235575092)	11 payable(0.9999999999999639)	11 examples(0.9999997695375432)
12 cottrell(0.9526324957583776)	12 treasurer(0.9999999999999564)	12 capping(0.9991661591613988)
13 torturous(0.9508983118916109)	13 challenged(0.9999999999999432)	13 vegas(0.9983717540616771)
14 brighton(0.9483409050386272)	14 tribunal(0.9999999999999397)	14 rant(0.9719190694394951)
15 norwich(0.9483407124651722)	15 objection(0.9999999999999133)	15 honorary(0.9559857903602881)
16 blackburn(0.946819258270919)	16 levied(0.9999999999999032)	16 nominated(0.9349556407948375)
17 bournemouth(0.9439080472705587)	17 categorising(0.999999999999844)	17 morocco(0.9283152707431312)
18 ipswich(0.9416560734163967)	18 excise(0.9999999999998432)	18 nomination(0.9279701539588738)
19 stambouli(0.9395815340712311)	19 filed(0.9999999999998308)	19 honor(0.925684337070635)
20 scunthorpe(0.9386430153061216)	20 incomes(0.9999999999998063)	20 politics(0.9248123112457677)
21 doncaster(0.9386430153061216)	21 aniruddh(0.9999999999997427)	21 malian(0.9217870745658601)
22 samuels(0.9371035648932646)	22 approx(0.999999999999621)	22 advert(0.9112201076365462)
23 farrell(0.9335216435226934)	23 franchisee(0.9999999999995722)	23 dictates(0.9103866665320417)
24 flair(0.9334332572598587)	24 rubbed(0.999999999999536)	24 ronaldinho(0.9091471503966787)
25 fulham(0.9329405432890424)	25 reactions(0.9999999999944525)	25 ginola(0.9073768435695492)
26 crewe(0.9322872006946415)	26 sehwag(0.9999999999874315)	26 spectacle(0.9073722950426439)
27 coventry(0.9322872006946415)	27 saini(0.9999999989919921)	27 unjust(0.9073587168448461)
28 dagenham(0.9322872006946415)	28 sunil(0.9999999932116826)	28 rummenigge(0.8985608549367298)
29 oxford(0.9322872006946415)	29 quadri(0.9999999585746101)	29 community(0.8984832322149107)
30 redbridge(0.9322872006946415)	30 chand(0.9999999068584364)	30 cruyff(0.8975742702800088)

topic 4	topic 5	topic 6
1.celta(0.9999999999999999)	1.eats(0.9999999184710473)	1.bellerin(0.9999999999999999)
2.buczko(0.9999999646768502)	2.uacute(0.9859173737243206)	2.unavailable(0.9999999999999994)
3.espanyol(0.9986056574620574)	3.wise(0.974769134948994)	3.exchanged(0.9999999999999979)
4.cordoba(0.9862105416416641)	4.arab(0.9632875646430983)	4.slot(0.9999999999999977)
5.vigo(0.9760247195301934)	5.pivotal(0.9606228837991432)	5.exiled(0.999999999999996)
6.atleti(0.9653013913380577)	6.jake(0.9560043545628523)	6.oxlade(0.9999999999999903)
7.nicholas(0.9646230861420124)	7.hertha(0.9525033585002535)	7.chamberlain(0.9999999999999903)
8.almeria(0.9637336801739939)	8.freiburg(0.9518167752475127)	8.chambers(0.9999999999999885)
9.benzema(0.9577522708739982)	9.eintracht(0.9484033847473442)	9.mere(0.9999999999999787)
10.bilbao(0.9430520186252918)	10.alaba(0.9451572750374607)	10.rose(0.9999999999999617)
11.stefan(0.9385184497956939)	11.como(0.9451191094320676)	11.mason(0.9999999999999617)
12.ledwith(0.9317430769109942)	12.china(0.9436859280459022)	12.cold(0.9999999999986827)
13.torres(0.9235993156231183)	13.hannover(0.9435286176427407)	13.rosicky(0.9999999999963367)
14.lowe(0.9125531721589859)	14.socceroos(0.9414959237106652)	14.sizzles(0.9999999999899233)
15.deportivo(0.8988224326566386)	15.shouts(0.9385041463405058)	15.inverse(0.9999999999806959)
16.gabriel(0.8959773149325851)	16.spirited(0.9340985546994166)	16.exhilarating(0.9999999998941341)
17.simeone(0.8933050641768014)	17.iacute(0.9298304096653707)	17.crack(0.9999997697254178)
18.hunter(0.8929880645485399)	18.davidson(0.9235481341910502)	18.jelavic(0.9999993894668203)
19.villarreal(0.8887497918803668)	19.lovell(0.9227914845215494)	19.mangan(0.9999974703659907)
20.blaugrana(0.8868206679929568)	20.gulf(0.9220992246900753)	20.mcnicholas(0.9999638464032292)
21.rakitic(0.883328044116797)	21.plot(0.9157796481250756)	21.pieters(0.9998199928110125)
22.eibar(0.8829237491731895)	22.kelvin(0.9157168027866137)	22.thrusting(0.9980636578682706)
23.ames(0.8786679613249788)	23.gritty(0.9132863002135124)	23.mertesacker(0.9881028897523594)
24.shirts(0.8779513505508068)	24.massimo(0.9093971689945106)	24.coquelin(0.9711144613813855)
25.anytime(0.8721683159596485)	25.burns(0.9051473479396882)	25.distin(0.94737892124727)
26.sassuolo(0.8679259255661438)	26.owen(0.902818666116898)	26.nikica(0.94737892124727)
27.smiled(0.8652325781512509)	27.sung(0.8995504922282137)	27.ciaran(0.94737892124727)
28.coppa(0.8636470610268202)	28.aussie(0.8987536603224012)	28.calf(0.9373051997109829)
29.catalans(0.8631861211724139)	29.badstuber(0.8963524400226299)	29.peak(0.9224984675149983)
30.uphill(0.8609736628570618)	30.este(0.8932116678358972)	30.fabianski(0.9212822510680737)

Table 6.6: Small Document Set Topic Term Matrix

6.4.3 With Medium size document sets

This test was a document set of 465 and deemed to be medium sized document set. The resultant topics and terms show that topic 0 relates to Spanish football teams like Almeria, Malaga and Levante as well as Cordoba which are all in the Spanish Premier top division.Topic 17 is a topic of English football lower division teams like Bolton, Crewe and Portsmouth. Topic 11 shows a cricketing world scenario with words like off spinner, tail ender and bowls making up the top 30 list. Interestingly topic 16 picked arsenal players of 2014, which is my favourite soccer team and names like Ozil, Giroud, Coquelin and Rosicky featuring in the top 30 list as well. Topic 5 is a topic made up of mostly Italian soccer names with players like Montolivo, an AC Milan player featuring as well as teams like Genoa and Cagliari both Seria A(Italian top division) teams. Topic 18, interestingly shows names of cricket playing cities like Dhaka(Bangladesh Cricket Team), Harare(Zimbabwe cricket team), and Durban(South Africa Cricket team) as well as other cricket names hence its a topic of cricket places and names. Topic 10 was identified as a German football names topic, whilst topics 8 and12 and is difficult to classify due to its huge mixture of words from different fields. Topic 13 was also found to be difficult to classify due to the fact that the research had no prior knowledge of most of the words in the list like 'dermotmcorrigan, sich, parisien, gabbiadini, klute, manolo, uuml, usmnt'. Some of the words could have been affected by the pre-processing of the data where some characters would have been removed then the eventual word formed being difficult to comprehend. Whilst a word like 'susofernandez' or 'acmilan' shows a word affected by pre-processing combining the 2 words Suso Fernandez, a footballer and AC Milan an Italian football team, the research could still comprehend the words.

Topic	Document List
Topic0	336,336,288,288,9,9,69,69,170,170
Topic1	499,6,499,6,153,153,364,364,10,10
Topic2	78,78,125,125,13,13,196,196,295,295
Topic3	306,306,199,199,200,200,117,117,181,181
Topic4	96,96,87,87,123,123,208,208,101,101
Topic5	202,202,264,264,392,392,256,256,255,255
Topic6	13,13,188,188,13,13,99,99,223,223
Topic7	200,200,128,128,680,680,564,564,271,271
Topic8	304,304,630,630,208,208,329,329,551,551
Topic9	480,480,70,70,418,418,473,473,465,465
Topic10	64,64,549,549,545,545,329,329,70,70
Topic11	167,167,593,600,600,593,177,177,553,553
Topic12	187,187,165,165,152,152,170,170,41,41
Topic13	115,115,295,295,600,593,600,593,717,717
Topic14	305,305,376,376,304,304,374,374,308,308
Topic15	99,99,83,83,129,129,59,59,19,19
Topic16	82,82,683,683,101,101,472,472,391,391
Topic17	58,58,18,18,123,123,47,47,26,26
Topic18	173,173,168,168,535,535,176,176,182,182
Topic19	46,46,180,180,36,36,30,30,26,26

Table 6.7: Medium Table Topic Document Matrix

Topic	Term List
Topic0	halilovic, cordoba, celta, submarine, mattered, rojiblancos, laurels, charisma, vigo, levante, almeria, nicholas, malaga, strife, copa, eibar, bilbao, atleti, ruptured, espanyol, deportivo, lapse, benzema, reminds, bernabeu, torres, jeering, gijon, urges, uphill
Topic1	iacute, motta, stefan, como, cisse, watzke, registering, federation, este, stade, bastia, campeonato, rennes, maurice, todo, roja, aacute, klopp, universidad, internacional, philadelphia, maurizio, kampl, pallotta, entrevista, mejores, boggling, westfalenstadion, coupe, osasuna
Topic2	laurence, coutinho, dejan, whoscored, lazar, okwonga, folly, scotsman, bridcutt , comparing, patterson, lesson, upsets, unreasonable, markovic, guadalajara, blind, column, scott, graziano, stubborn, tadic, recapturing, security, brown, usher, fanzine, dusan, trends, baffling
Topic3	zouma, cartoon, larry, dinamo, approve, sponsorship, zambia, arbeloa, prodigy, indomitable, haidara, rummenigge, cruise, hotel, starters, winding, qualified, gabon, herve, burkina, gent, oblak, expressly, absences, episode, politics, firmly, recurring, ludicrous, remarkably
Topic4	soorma, otago, gadha, dayaben, testing, laxmanism, redefining, bhopal, egged, entertainers, latham, bhuvneshwar, dilshan, overseas, whites, jerseys, fastest, tillakaratne, mendis, contemplated, dismissing, milne, pitches, angelo, elude, williamson, jeevan, senanayake, cupboard, resume
Topic5	filipenko, edimar, ighalo, jude, odion, montolivo, massimiliano, allegri, rzouki, raiola, coppa, origi, grimes, schwarzer, instilling, cagliari, turin, swansofficial, genoa, kimmich, chievo, kone, loris, italia, wesley, pogba, cesena, stevie, assaidi, divock
Topic6	woolloongabba, graphs, riverside, yorkshire, blast, lawrence, gardens, queensland, perth, glenelg, county, hobart, grace, dubai, scarborough, street, drawn, royal, division, ground, queen, shield, adelaide, hurricanes, gwalior, memorial, worcester, chester, zimbabwe, chelmsford
Topic7	shayk, rugby, strictly, casino, showbiz, beach, weird, film, sections, date, bacary, science, chaos, location, rampant, crime, info, alarm, racing, dealt, health, heavyweight, sagna, closes, postponed, path, raid, morrison, horse, ravel
Topic8	bookies, illustrations, flanagan, checked, physio, listed, venezuela, vidal, congratulations, kingdom, meets, compares, activate, subscribers, recruiting, customer, insider, azerbaijan, lotito, belarus, estonia, generations, breathtaking, converted, deportes, conversation, trinidad, consensus, slovenia, laurentiis
Topic9	kuwait, duerden, china, arab, pivotal, uzbekistan, brooks, jake, arabia, saudi, wise, socceroos, blue, massimo, plot, davidson, burns, mile, gritty, aguirre, spark, wilkins, gear, sung, spirited, aussie, abdul, kelvin, spots, mike
Topic10	emanating, alaba, lovell, owen, cast, redemption, presidency, hoffenheim, paderborn, pocket, ames, rounding, hertha, lewandowski, eintracht, smoked, fifa, awesome, concept, poke, freiburg, hannover, reina, climb, showcase, upstaged, holger, welt, stream, mexicans
Topic11	gilchrist, workload, offspinner, officiating, bowls, compile, tailenders, gloomy, truncated, afridi, shahid, marsh, reduce, sandhu, countdown, topics, saeed, balwinder, honed, bailey, doherty, brett, pietersen, gordon, coverage, taker, brien, selectors, misbah, kapil
Topic12	thwart, dale, thrash, larsson, naismith, bets, calf, prediction, hour, groin, prem, boot, elaborate, cole, citizens, oliveira, zabaleta, entries, fabianski, assumed, nasri, gain, dogged, involvement, kolarov, pelligrini, clark, besic, jagielka, sigurdsson
Topic13	potters, dermotmcorrigan, sich, acmilan, tempt, parisien, trial, gabbiadini, villans, klute, manolo, uuml, usmnt, reunion, ings, mehr, wishes, auml, correa, leiva, requirements, diskerud, echo, tune, rich, cats, guido, lacazette, feruz, islam
Topic14	sudhir, punia, manish, maharaja, refresh, scorecard, trail, toyota, malik, write, purge, cheer, sponsored, cracked, singh, sarkar, pathan, yadav, chand, sanjay, whopping, gallery, nieuws, kaushik, voetbal, cheteshwar, gupta, patel, varun, overseeing
Topic15	punishment, appoints, luiz, salzburg, carr, nunes, shakhtar, suso, adriano, ricardo, seattle, seals, canadian, shaw, tips, song, reschke, montreal, susofernandez, fcvtsfdayu, stromsgodset, bender, suits, transfert, safc, restructure, officiel, fcmetzofficiel, avant, commute
Topic16	intangibles, crack, rematch, sorely, coquelin, monreal, walcott, conundrum, mertesacker, ozil, giroud, unsung, chambers, styles, ospina, wally, olivier, rosicky, mikel, bellerin, gunnerblog, mangan, debuchy, contrasting, boardroom, cazorla, arnautovic, hoodoo, santi, scratching
Topic17	portsmouth, sordell, farrell, wallace, ipswich, rovers, bournemouth, bolton, mcgeady, brighton, scunthorpe, peterborough, crewe, wycombe, dagenham, crawley, coventry, redbridge, blackburn, stambouli, doncaster, gateshead, oxford, wolverhampton, sevens, hails, robles, southport, aiden, carroll
Topic18	kingsmead, harare, russell, cottrell, morne, bravo, johannesburg, dhaka, slower, saxton, elizabeth, rounder, windies, connects, behardien, drags, flick, gayle, pitched, imran, farhaan, university, morkel, proteas, villiers, justin, steyn, afghanistan, durban, bridgetown
Topic19	crippling, topped, eagerness, replays, parachuted, mettle, samba, stephanie, coronation, settling, convincingly, roden, annoyance, gaston, shine, cuevas, murphy, vickery, hesse, regard, calmed, prizes, gabriel, vintage, brassell, excuses, lists, reasonable, perceived, grows

Table 6.8: Medium Table Topic Term Matrix

6.4.4 With large number of document entries

To train the EM algorithm with almost 765 the research takes articles from the following websites namely basketball, rugby, aviation as well as ship and sea news. The plsa run is done with a value of 0.000001 forthe LogLikelihood calculator as well.

```
String query ="SELECT * FROM test.crawledWebSites where seedUrl='http://espn.go.com/nba/'or
    seedUrl='http://www.espncricinfo.com'"
          + " or seedUrl='http://www.espnscrum.com' or seedUrl='http://aviationweek.com' or
              seedUrl='http://www.seashipnews.com/' ;";// "select * from crawledWebSites
              order by startDownloadTime "
```

Since this was a huge data scenario, out of it came 20 topics and some of the topics picked from the 20 are shown in table x. The topic document matrix table [**?**] shows the most likely document in the hidden topic z. Topic 11 or the first topic has document 167, 600, 553 and 289 as its most visible documents falling into its topic category. Table y shows the topic term matrix with terms meeting the top 30 highest probability being shown. Topic 11 on the document term matrix shows its a list of names of cricket players like Lasith Malinga and Angelo Mathews both from Sri Lanka. Topic 11 is made of mostly cricket names like finch for Aaron Finch, lasith for Lasith Malinga and maxwell for Glen Maxwell the Australian cricket batsmen. Topic 10 shows tennis words with serena for Serena Williams and nishikori both tennis players. Topic 19 shows African names for countries and some of their currencies with words countries like Ethiopia, Tunisia, Rwanda, Djibouti as well as Sierra Leone(from the words leone and sierra) and African currencies like Shilling which is used in Rwanda appearing in the top 30 probability list.

Topic 7 is difficult to categorize into a specific topic since the words looked like Spanish or Portuguese words like 'camacho, malibu, coulibaly, ccedil, bolsa, iacute, estados, unidos, sobre, latina, negocios, canad, frica, reino, unido, janeiro'. With the research only well versed in English, it was difficult to be able to tell if a topic of consistent meaningful words were formed. Thus topic 7 was categorized as a topic of either Spanish or Portuguese words. Topic 2 is composed of mostly Indian and Pakistan words like 'abhimanyu' which the research failed to know if they were people's names or just meaningful words. Topic 9 is mostly football names in the English league like Everton FC and topic13 was found difficult to classify due to diversity in the meaning of the words.

Topic	Document List based on PLSA Algorithm
Topic0	336,336,288,288,9,9,69,69,170,170
Topic1	499,6,499,6,153,153,364,364,10,10
Topic2	78,78,125,125,13,13,196,196,295,295
Topic3	306,306,199,199,200,200,117,117,181,181
Topic4	96,96,87,87,123,123,208,208,101,101
Topic5	202,202,264,264,392,392,256,256,255,255
Topic6	13,13,188,188,13,13,99,99,223,223
Topic7	200,200,128,128,680,680,564,564,271,271
Topic8	304,304,630,630,208,208,329,329,551,551
Topic9	480,480,70,70,418,418,473,473,465,465
Topic10	64,64,549,549,545,545,329,329,70,70
Topic11	167,167,593,600,600,593,177,177,553,553
Topic12	187,187,165,165,152,152,170,170,41,41
Topic13	115,115,295,295,600,593,600,593,717,717
Topic14	305,305,376,376,304,304,374,374,308,308
Topic15	99,99,83,83,129,129,59,59,19,19
Topic16	82,82,683,683,101,101,472,472,391,391
Topic17	58,58,18,18,123,123,47,47,26,26
Topic18	173,173,168,168,535,535,176,176,182,182
Topic19	46,46,180,180,36,36,30,30,26,26

Table 6.9: Big Data Latent Topic Document Matrix

Topic	Terms
Topic0:	, chevron, unitedhealth, merck, towers, insane, lynda, milian, sovereign, weakens, intraday, payrolls, queue, nicest, feeding, pulse, variable, lendingclub, advisers, nivea, satisfy, renters, gopro, billionaire, takeover, frenzy, zillow, cornerstone, gamestop, royalties, flats
Topic1:	, weigel, credits, rushdie, harmful, graphic, scramble, trucks, published, capita, sedans, gasoline, reset, wages, criteria, inequality, firefox, pitting, lawyer, incomes, metrics, consumption, neighbors, upgrade, vehicles, subsidies, leonid, sized, buttons, contributors, screen
Topic2:	, interestingly, counterattacked, ovation, doggedly, uphill, prabhakar, rohtak, vishaal, ajith, loganathan, swapnil, madhya, purge, shreyas, yusuf, pathan, chirag, stalwart, facile, flatten, photoshoot, abhimanyu, uttar, crores, deepak, shrikant, manish, maharashtra, kerala, signup
Topic3:	, martial, fivethirtyeight, shortstop, knicks, raiders, rivers, cowboys, coordinator, bucks, playoff, redskins, jefferson, hockey, belichick, berry, angels, tampa, minors, playoffs, orange, draft, patriots, garrett, manning, offseason, bowden, thorpe, bulls, trash, underdog
Topic4:	, norah, donnell, gubernatorial, biden, moneywatch, latino, montana, restoration, stumble, donors, adopting, arrest, glitter, argues, enemies, cornell, recorder, views, protecting, overtime, petraeus, asylum, zimmerman, schwartz, turnabout, protective, romney, amendment, dickerson, ernst
Topic5:	, wiese, samuels, plessis, fletcher, hendricks, sammy, vines, duminy, edberg, norman, bjorn, extras, jaroslav, boris, flick, dwayne, herbert, simmons, william, santana, russell, johnston, lendl, nurse, slaps, tossed, outclassed, shoaib, steffi, morne
Topic6:	, earthquake, diplomacy, telecoms, wasik, buyouts, breakingviews, screener, linda, pehub, crafting, edited, faithworld, unstructured, workstation, incorporating, adchoices, macro, investigates, aluminum, settlements, currie, gundlach, dividend, indices, summits, nasdaq, photographers, flagship, aerospace, hutchinson
Topic7:	, streisand, reiner, ejection, selmi, camacho, malibu, coulibaly, ccedil, bolsa, iacute, estados, unidos, sobre, latina, negocios, canad, frica, reino, unido, janeiro, mundo, ayuda, desde, cierra, uacute, francia, jueves, rusia, franco, alemania
Topic8:	, shinpads, darlings, vader, shrunk, obscurity, trotted, toiling, inexact, belonged, equalled, recriminations, collar, palpatine, darth, grandstanding, patellar, insinuations, baubles, demolished, antics, cedes, tendonitis, gritting, limped, preening, liberally, trope, grownup, clutching
Topic9:	, berahino, arnhem, flanagan, boateng, hammers, hicks, farrell, laurence, villa, spurs, cambridge, tedious, borough, wigan, coventry, luton, everton, pochettino, bolton, telford, thorne, pardew, madden, albion, bacca, weston, nerve, martinez, peterborough, dreaming
Topic10:	, serena, bacsinszky, espntennis, kukushkin, wrist, ortiz, forums, nishikori, garber, melissa, potro, withdraws, sania, dodge, farrington, heineken, tennis, wilder, pacquiao, soprano, dunham, temper, https, chronicles, gilbert, quarterfinals, dunlop, baghdatis, sister, huffington
Topic11:	, lasith, alien, mccullum, finch, karunaratne, ronchi, dimuth, lankans, mathews, justify, starc, brigade, maxwell, positives, vettori, guptill, dilshan, excite, selfless, exaggeration, brendon, fashioned, whirlwind, haddin, sundry, nathan, behrendorff, dobell, posted, overseeing
Topic12:	, grauer, doctoroff, mazzeo, customizable, succeeding, rembrandt, messenger, justices, collaboration, offerings, envoy, humanitarian, pharma, nyusi, latam, payroll, expertise, crises, applicant, clients, jacobson, customized, codes, advisory, pricing, releases, analytics, eclipse, principles, bender
Topic13:	, dread, removes, roiled, koreans, tickers, consoles, roadblocks, reaffirmed, tumbles, swatch, schoolchildren, coordination, ibovespa, davos, blasio, exporters, saitto, unexpectedly, nbcuniversal, greenspan, hollen, obstacle, pilgrim, lowering, hannah, spoils, penney, scrap, snapshot, yield
Topic14:	, summarised, licence, wallpaper, rosberg, onboard, sauber, ferrari, timetable, premio, sutton, notices, edits, paddock, glossary, magnussen, vettel, formula, kobayashi, alonso, ricciardo, bianchi, schumacher, bernie, vergne, drivers, technical, rosso, jenson, races, kamui
Topic15:	, edging, disjointed, configured, injure, dazzling, equalising, betraying, gardos, coughed, smiled, fabian, routing, ribery, observer, turin, enyeama, biancocelesti, giallorossi, laurentiis, aurelio, ballon, zidane, corrigan, palestine, dermot, borussia, redknapp, delaney, ligue, dortmund
Topic16:	, ressler, engages, aired, roundtable, newsmakers, karen, moritz, attorneys, quinn, professor, kurobuta, halts, intricate, timepieces, fashions, terminal, chipotle, guests, strategist, profession, heathrow, methane, emily, russians, bolsters, analyzed, informative, megan, cosme, mcardle
Topic17:	, paarl, envision, mommsen, halsall, gloucs, worcs, lancs, mitchel, warks, yorks, patiently, gavskar, groundsman, sixers, coetzer, berrington, bloemfontein, scorchers, renegades, alasdair, leask, machan, kingston, rajkot, knights, okavango, sharif, gardiner, safyaan, hamish
Topic18:	, srinivasan, gavaskar, astonishingly, aggressively, refuses, sportsmanship, ganga, chopra, ipads, departments, maligned, realise, lorgat, heartbreak, angry, transformation, commit, surpasses, usman, sidharth, tablets, manjrekar, monga, spite, dravid, widgets, movers, sohail, allan, headache
Topic19:	, foyle, handpicked, ulster, detectives, hippo, territories, tunisia, leone, sierra, islands, ethiopia, churchill, fruit, upgrading, enabling, djibouti, burkina, dinar, belarus, newsbeat, reddit, murdered, tanzania, rwanda, shilling, programmes, soars, rupee, insanity, centcom

Table 6.10: Big Data Latent Topic Term Matrix

6.5 Results of PLSA tests

Whilst the previous section shows successful deduction of several topics through generation of the topic term and topic document matrices by running the PLSA algorithm, the research also looks at the efficiency of running this algorithm by considering various factor changes. Some of the tests included increasing the value of k, the number of topics and also changing the number of documents in the set to see if an increase in the document will change the speed and efficiency as well as accuracy of the PLSA algorithm. A total of 20 tests were done considering various combinations as shown in the following table. Notably some tests like test 8 indicate a failed test where due to using 1000 documents in the set, the PLSA algorithm implementation in java ran out of heap memory during processing and gave out of memory error. On increasing the memory for the java heap sizes by 3GB, a repeat of the test still failed. Another repeat test 9 was then done on the new memory parameters set and minimal logging removed and the test succeeded as test 9. Raw data or web documents with no processing meaning they have bigger vocabularies since stop words were not removed. Thus raw web documents or pre-processed files have more time to execute the Expectation Maximization procedure than the post processed files, though the former doesn't suffer from information loss since no data was filtered.

Table 6.11: PLSA Tests List

Test Type	Status	Details
Test 2	Success	Run with processed data for k=10,200 documents for topics. Seed urls are espnfc.com and espncricinfo.com
Test 3	Success	Run with processed data for various values of k=10, 20 and 5 and constant 202 documents for topics. Seed urls are espnfc.com and espncricinfo.com
Test 4	Success	Run with processed data for various values of i=0.01, i=0.001, i=0.00000001 and constant 202 documents for topics. Seed urls are espnfc.com and espncricinfo.com
Test 5	Success	Run with raw data for various values of k=10,20 and 5 and constant 202 documents for topics. Seed urls are espnfc.com and espncricinfo.com
Test 6	Success	Run with processed data for various values of k=10, 20 and 5 and varying documents list(4) for topics
Test 7	Failed	Run with processed big data (1000 documents) Out of Memory Error on Generate Matrix Table. Another test to be done with more memory from -Xms1024M -Xmx1024M to -Xms2048M -Xmx2048
Test 8	Failed	Run with processed big data(1000 documents)Repeat of test 7 with new memory resources ,and reduced logging
Test 9	Success	Run with processed big data(1000 documents) for various k=10, 20 and 50
Test 10	Success	Run with processed big data(1000 documents) for various k=10, 20 and 50 and various iterator comparator values of 0.01, 0.0001 and 0.00000001
Test 11	Success	Run with raw data for various values of k=10, 20 and 5 and varying documents list(4) for topics.

6.5.1 Performance of the algorithm with 202 documents batch

From the performance metrics of the test runs against the 202 documents batch, the research can deduce that matrix generation process is dependent to the number of documents and subsequently the vocabulary. Bigger vocabulary also meant more time during the matrix generation phase. Slight differences for documents having same vocabulary is probably due ad hoc slow responses on the server due to other uses of the server when running or poor responses due to logging massive amounts of output data. The time to run the Expectation Maximization is a broader aggregate of the Expectation and Maximization phase as well the calculation of the log likelihood. The

total Expectation Maximization time increases the number of topics is increase. An increase in the value of the log iterator value from 0.000001 which is the default value will reduce the number of iterations as well reduce the time to execute Expectation Maximization algorithm. The time to get to the maximum likelihood of the log likelihood is also easily calculated faster and in a shorter time since the value to converge has been reduced.

Table 6.12: Performance of PLSA 202 Documents

Topics	Documents	Iterations	Time EM	E-Step Time	M-Step Time	Log Likelihood	Matrix Generation	Initialisation	Iterator Comparator
5	202	173	11450	494	1369	431	6061	6	0.000001
10	202	371	14459	1900	1645	1288	6025	8	0.000001
20	202	439	31742	5639	14113	2584	6242	16	0.000001
20	202	2135	134081	29702	74993	15287	6131	26	0.00000001
20	202	48	16493	721	1779	326	6205	19	0.0001
20	202	2	1007	61	192	47	6221	19	0.01

Table 6.13: Performance of PLSA with raw data(202 Documents)

Topics	Vocabulary	Iterations	Time EM	E-Step Time	M-Step Time	Log Likelihood	Matrix Generation	Initialisation	Iterator Comparator
10	9489	283	19669	1826	5394	1207	8369	11	0.000001
20	9489	247	24084	3633	8900	1530	8909	14	0.000001
5	9489	350	15792	1141	3069	1530	7355	4	0.000001
20	9489	2589	158697	36724	89895	16739	8287	18	0.00000001

CPU and Memory Utilization

The performance of this run is shown in the table [**?**]. The average CPU used for classifying the 202 documents is 20 per cent and the java heap memory also peaked at 720MB. The most effective tests saw the CPU usage only peak to 18.5 per cent and java heap size peaking to 720MB. Reducing the iterator comparator by a factor of 100, saw the total Expectation Maximization time to execute increases ten fold as well as the log likelihood converging at a time 10 times slower than when the value was 0.000001. The number of iterations also increased 10 times when the iterator comparator was reduced by a factor of 100.

Table 6.14: CPU and Memory of PLSA

Topics	Documents	Iterations	Time EM	CPU	Java Heap Size Memory	Iterator Comparator
5	8595	325	19516	18.5percent	500MB	0.000001
10	8595	371	32523	21.5percent	650MB	0.000001
20	8595	439	106364	22percent	720MB	0.000001
20	8595	2135	22136	29.5percent	820MB	0.00000001
20	8595	48	19516	17.5percent	790MB	0.0001
20	8595	2	22136	16.5percent	810MB	0.01
10	9489	283	106364	22percent	720MB	0.000001
20	9489	247	22136	22.5percent	720MB	0.000001
5	9489	350	19516	18.0percent	550MB	0.000001
20	9489	2589	15869	28.5percent	900MB	0.00000001

Figure 6.3: PLSA Execution Tab

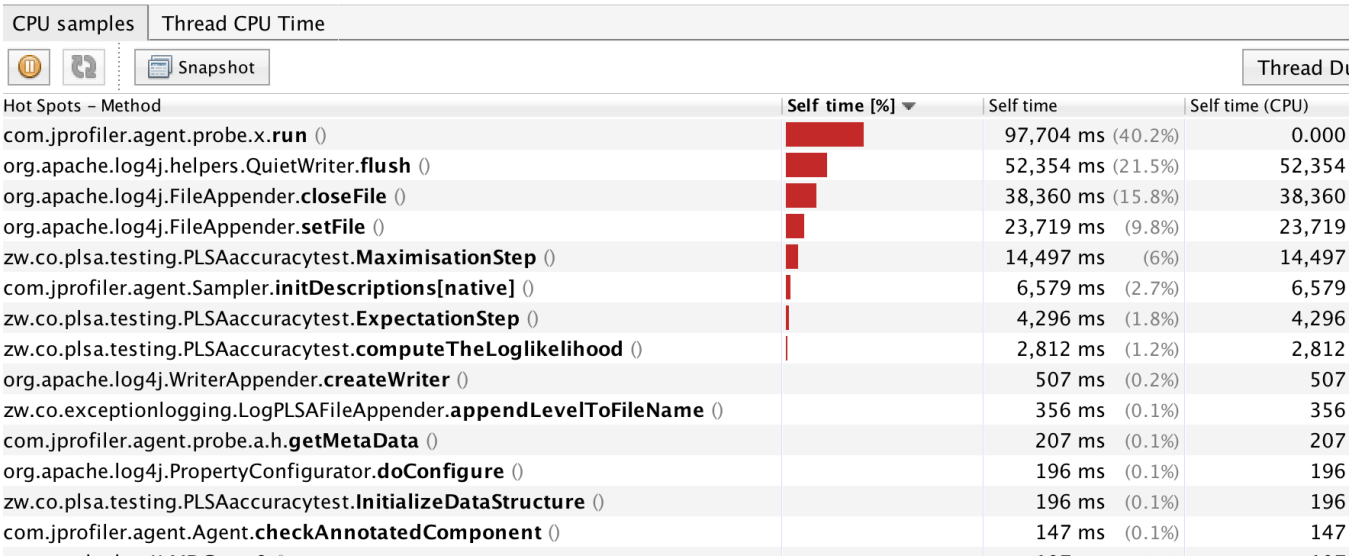
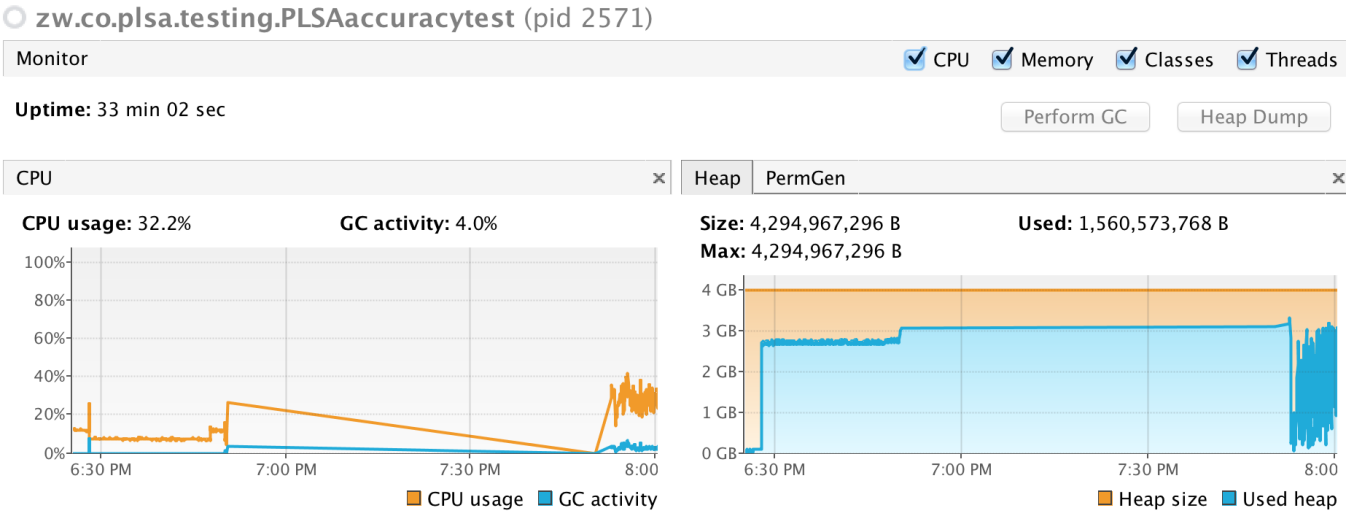


Figure 6.4: Documents by CPU



6.5.2 Performance of the algorithm with a mixed document model batch

The mixed batch consists of all urls from bloomberg.com, cbsnews.com/politics and talk sport football news. A total of 407 documents made up the batch with 51, 159 and 197 documents for the three seed urls respectively. The combined word vocabulary numbered 14075. The second batch adds medicalnewstoday.com website as well space.com as new seed urls. A SQL search to the MySQL database is done for these 5 websites and this forms the data to be run against the PLSA algorithm. To speed up the SQL search table index on the seedUrl was created on the MySQL database. This set had 603 documents and a vocabulary of 21560 words. The other batch set is a query of documents from espnfc.com as well as espn.com/nba for basketball news and the batch had 765 documents. The 450 document batch had espnfc.com and cricbuzz.com.

CPU and Memory Utilization

The performance of this run is shown below:

Table 6.16: Heap Size and CPU of PLSA

Topics	Documents	Iterations	Time EM	CPU	Java Heap Size Memory
10	765	406	112026	31percent	3.9GB
20	765	406	113402	28percent	4.2GB
30	765	406	114019	31percent	4.6GB
40	765	406	114912	46percent	4.9GB
50	765	406	115178	55percent	5.6GB

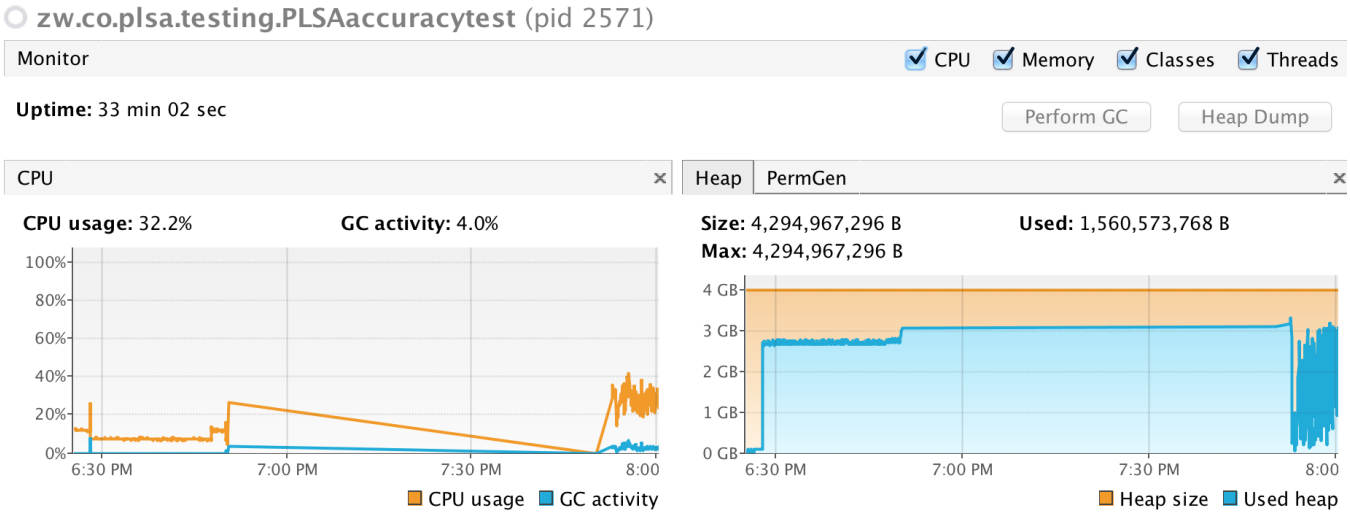
Table 6.15: Performance of PLSA

Topics	Docs	Iterations	Time EM	E-Step Time	M-Step Time	Log Likelihood	Matrix Generation	Initialization	Iterator Comparator
20	407	193	44880	7269	17135	3596	29875	43	0.000001
50	407	164	66007	14146	31080	5937	33789	133	0.000001
10	407	157	21927	1946	1563	1351	2841	13	0.000001
20	450	436	1261866	52193	1050690	70521	45428	62	0.000001
50	500	613	532620	76264	183521	249244	79309	255	0.000001
10	603	130	51582	5201	14452	3699	120519	56	0.000001
20	603	141	77027	12407	132305	6000	117506	131	0.000001
20	765	553	158306	32248	84273	18274	81594	31	0.000001
10	765	325	19516	1358	3872	1399	11227	5	0.000001
20	765	346	32523	8853	3209	2245	11658	11	0.000001
30	765	325	19516	1358	3872	1399	11227	11	0.000001
50	765	450	106364	11063	7681	5347	140720	22	0.000001
20	765	325	19516	1358	3872	1399	11227	22	0.000001
20	765	325	19516	1358	3872	1399	11227	22	0.000001
10	1000	613	532620	76264	183521	249312	79309	255	0.000001

Topics	Documents	iteration Value	Taken EM	timeTaken PLSAInit	timeTaken DocTermMatrix
1.42176E+12	20	566	5219660	8.12296E+11	18543
1.42176E+12	10	566	34918	26863065000	19220
1.42175E+12	50	566	218890	30013918000	19220
1.42175E+12	10	566	25667	25178070000	19220
1.42175E+12	20	566	168926	25026738000	19220
1.42175E+12	20	566	68157	24398616000	19220
1.42174E+12	15	566	28680	8.04537E+11	18543
1.42174E+12	20	566	12812	8.37515E+11	18543
1.42174E+12	20	566	284698	8.20655E+11	18543
1.42173E+12	10	566	25837	8.08076E+11	18543
1.42173E+12	20	566	93799	8.34755E+11	18543
1.42172E+12	50	566	784300	8.06667E+11	18543
1.42172E+12	20	298	56658	2.12291E+11	9250
1.42172E+12	10	298	11646	2.10851E+11	9250
1.42172E+12	10	202	41820	1.79878E+11	11700
1.42172E+12	15	202	16114	1.80326E+11	11700
1.42172E+12	10	202	7527	1.82528E+11	11700
1.4217E+12	5	202	4229	1.79253E+11	11700
1.4217E+12	5	202	4316	1.76788E+11	11700
1.42169E+12	20	202	15148	1.80982E+11	11700
1.42168E+12	10	202	5852	1.8063E+11	11700
1.42168E+12	10	309	6857	1.60636E+11	6919
1.42168E+12	10	432	17153	6.10773E+11	17695
1.42164E+12	10	432	19443	4.15899E+12	17695
1.42164E+12	10	432	20699	5.85391E+11	17695
1.42163E+12	50	432	117776	6.27228E+11	17695
1.4216E+12	20	733	136536	1.18161E+12	21226
1.42159E+12	20	465	35514	5.62082E+11	15897
1.42159E+12	10	465	1175414	1.07969E+12	15897
1.42147E+12	10	500	134144	6.90738E+11	18226
1.42146E+12	5	200	3003	85684897000	5521
1.42144E+12	50	500	511951	1.84256E+12	20741
1.42143E+12	20	200	34118	1.79922E+11	11561
1.42142E+12	20	1000	204355	2.38892E+12	30714
1.42141E+12	10	1000	88375	2.37633E+12	30714
1.42141E+12	10	500	51910	7.92654E+11	20741
1.42141E+12	10	300	18335	3.10372E+11	13508
1.42141E+12	10	200	14379	1.77806E+11	11561
1.42141E+12	10	100	5781	62177773000	8146

Table 6.17: PLSA with Varying Topics

Figure 6.5: PLSA with Varying Topics



6.6 Log Analysis

Checking performance of the PLSA algorithm is facilitated by using tracing performance of the sub-procedures making up the Expectation Maximization algorithm as shown in table[**?**]. The program is then able capture the start and end times and get their execution times and also frequency of execution of the methods. The technologies used in this include AspectJ an Aspect Oriented Programming framework as well as Log4j for writing logs.

6.6.1 Log tracing using Log4J and AspectJ

Table 6.18: Gather PLSA Data using AspectJ and Log4J

Gather PLSA Data using AspectJ and Log4J		
Join	**Join**	**Execution**
before(),after()	ExpectationMaximisationAlgoImplementer()	execution(* zw.co.WebCrawling.PLSAWebCrawler. ExpectationMaximisationAlgoImplementer(..)).
before(),after()	expectationStep()	execution(* zw.co.plsa.testing.PLSAaccuracytest.ExpectationStep(..)).
before(),after()	maximisationStep()	execution(* zw.co.plsa.testing.PLSAaccuracytest.MaximisationStep(..)).
before(),after()	computeTheLoglikelihood()	execution(* zw.co.WebCrawling.PLSAWebCrawler. computeTheLoglikelihood(..)).
before(),after()	initProbabilities()	execution(* zw.co.WebCrawling. PLSAWebCrawler.initProbabilities(..)).
before(),after()	generatePLSAMatrixTable()	execution(* zw.co.plsi.Plsa2.generatePLSAMatrixTable(..)).

Table 6.19: PLSA AspectJ Log Analysis

15:20:32,130 WARN zw.co.exceptionlogging.AspectJAspect - computeTheLoglikelihood has ended: 130 milliseconds total Log Likelihood Computation Time=70129
15:20:32,131 WARN zw.co.exceptionlogging.AspectJAspect - ExpectationStep has started:
15:20:32,234 WARN zw.co.exceptionlogging.AspectJAspect - ExpectationStep has ended: 103 millisecondstotal Expectation Time=52097E Step Number= 435
15:20:32,235 WARN zw.co.exceptionlogging.AspectJAspect - MaximisationStep has started:
15:20:33,825 WARN zw.co.exceptionlogging.AspectJAspect - MaximisationStep has ended: 1590 millisecondstotal Maximization Time=1049149
15:20:33,826 WARN zw.co.exceptionlogging.AspectJAspect - computeTheLoglikelihood has started:
15:20:33,963 WARN zw.co.exceptionlogging.AspectJAspect - computeTheLoglikelihood has ended: 137 milliseconds total Log Likelihood Computation Time=70266
15:20:33,964 WARN zw.co.exceptionlogging.AspectJAspect - ExpectationStep has started:
15:20:34,061 WARN zw.co.exceptionlogging.AspectJAspect - ExpectationStep has ended: 96 millisecondstotal Expectation Time=52193E Step Number= 436
15:20:34,062 WARN zw.co.exceptionlogging.AspectJAspect - MaximisationStep has started:
15:20:35,604 WARN zw.co.exceptionlogging.AspectJAspect - MaximisationStep has ended: 1541 millisecondstotal Maximization Time=1050690
15:20:35,605 WARN zw.co.exceptionlogging.AspectJAspect - computeTheLoglikelihood has started:
15:20:35,732 WARN zw.co.exceptionlogging.AspectJAspect - computeTheLoglikelihood has ended: 127 milliseconds total Log Likelihood Computation Time=70393
15:20:35,741 WARN zw.co.exceptionlogging.AspectJAspect - computeTheLoglikelihood has started:
15:20:35,870 WARN zw.co.exceptionlogging.AspectJAspect - computeTheLoglikelihood has ended: 128 milliseconds total Log Likelihood Computation Time=70521
15:22:02,320 WARN zw.co.exceptionlogging.AspectJAspect - ExpectationMaximisation has ended: 1261866 millisecondstotalDocument Generation Time=45428totalInitializationTime=62

6.6.2 CPU Consumption during execution

Figure 6.6: CPU and Heap Size(450 Documents)

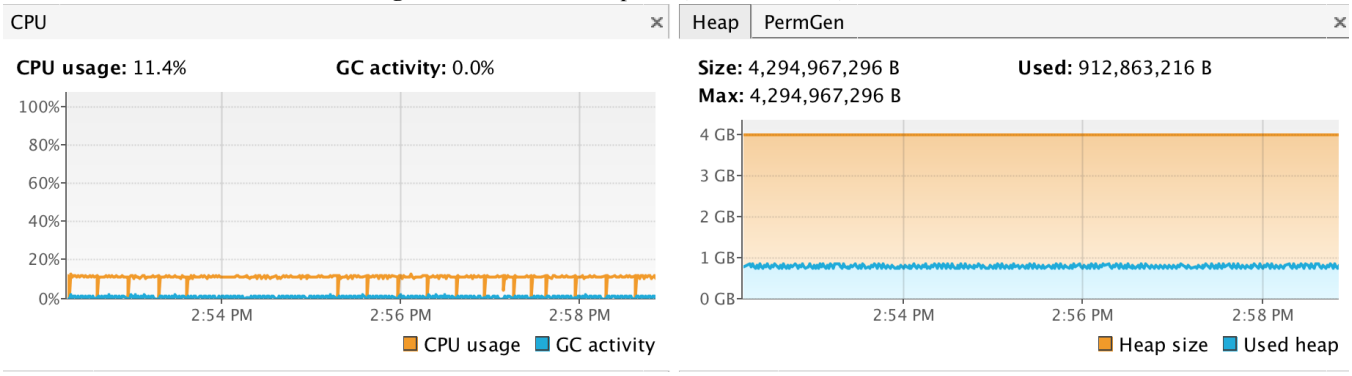
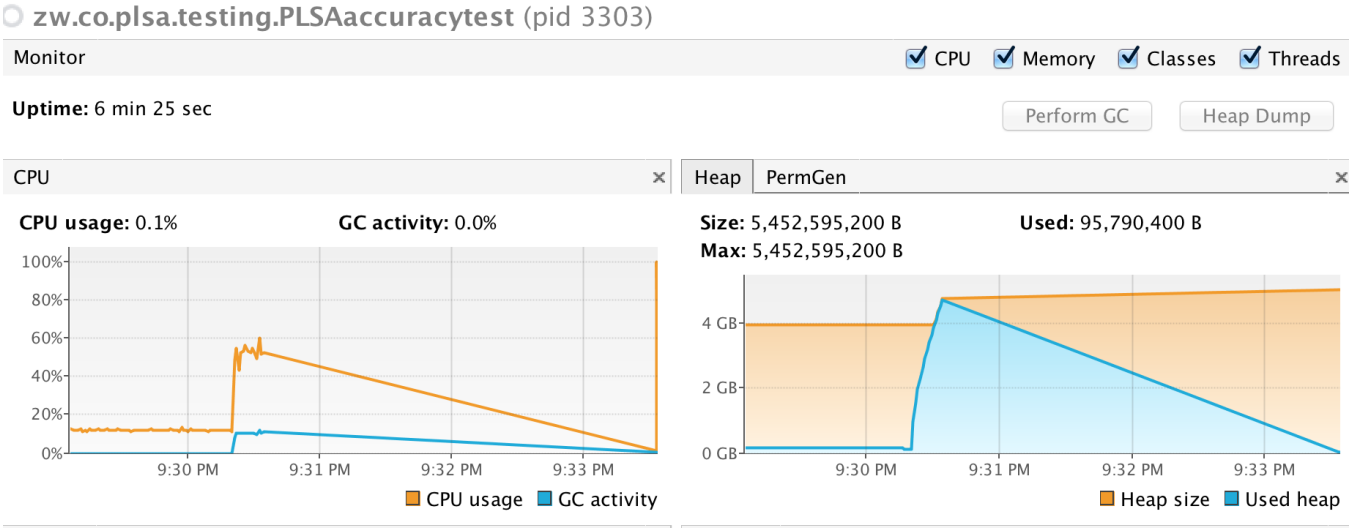


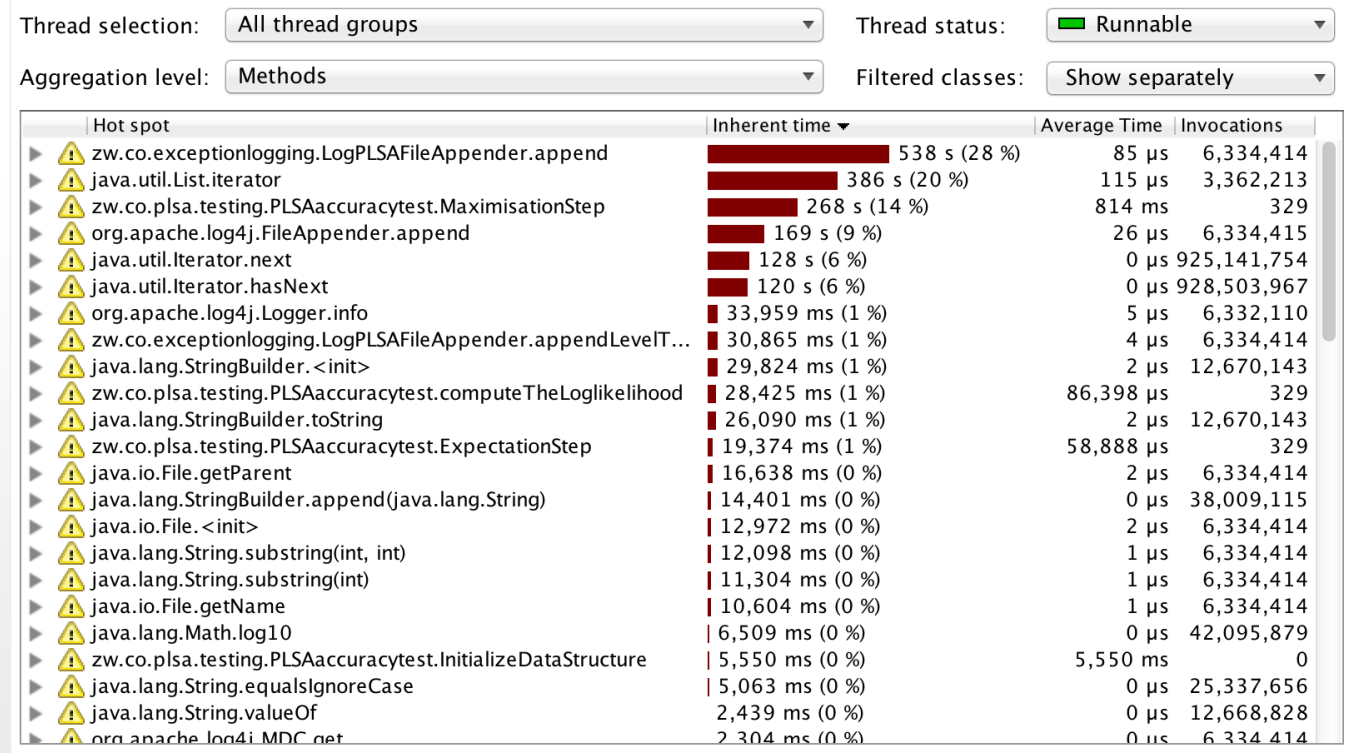
Figure 6.7: CPU and Heap Size(450 Documents)



6.6.3 Java threads during processing

Based on the memory results above, the tests with the 1000 documents failed when the java heap went above the allocated maximum heap size in the failed test case scenario.

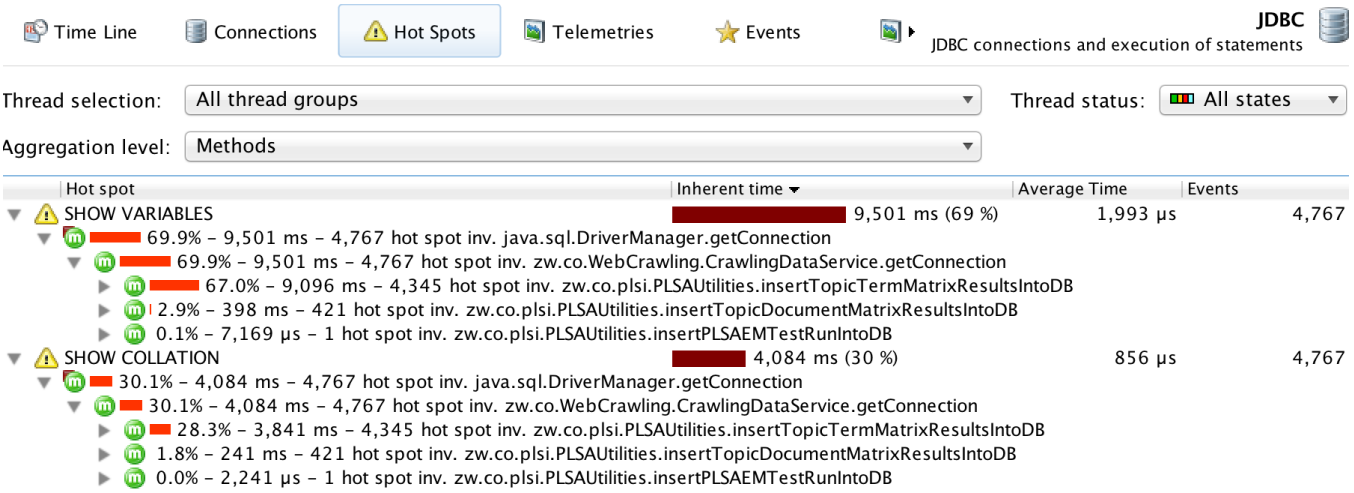
Figure 6.8: Java Heap Processing



6.6.4 Inserting data into MySQL Database

The PLSA algorithm implementation does huge database connection during the reading and extracting of the web crawled data. After the preparatory phase, there are also huge database inserts as the implementation creates both the topic document matrix as well as the topic term matrix. The java monitoring tool JProfiler shows huge response time during the 'generatePLSAMatrixTable' table phase due to theses massive matrix generation as shown in [**?**] .

Figure 6.9: Insert PLSA Topic Matrices



6.7 Analysis of PLSA implementation observations and findings

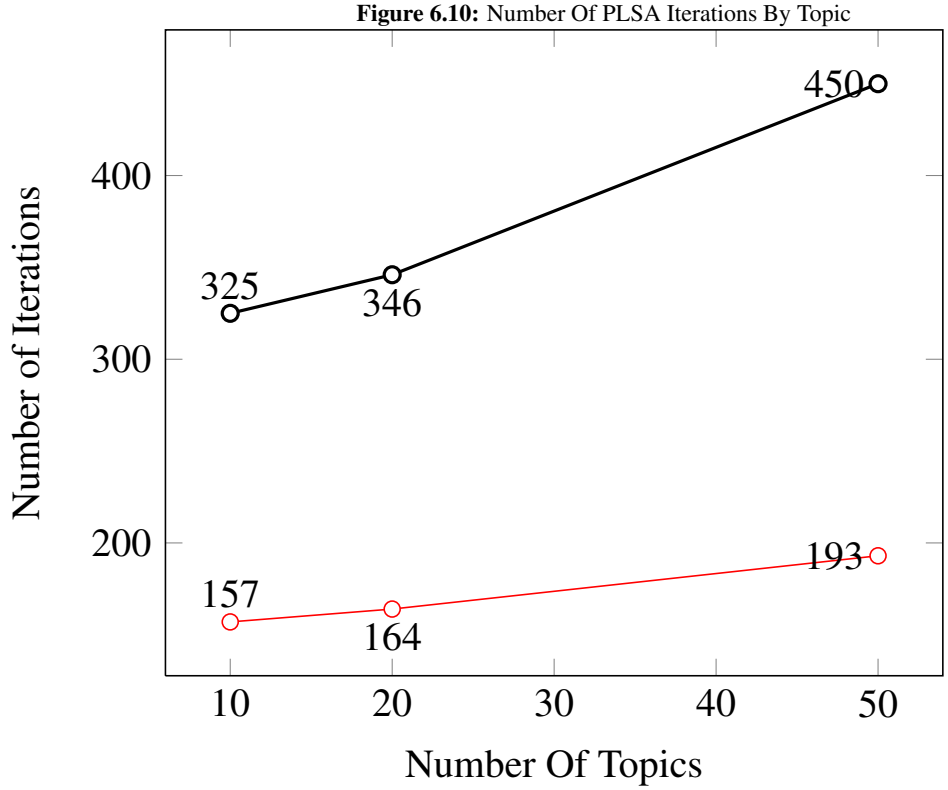
The experiments carried out on running PLSA algorithm showed a lot of findings. The research found out that the algorithm did successfully produce topics with very relevant meaning. To be able to understand the meaning of the topic did require in depth knowledge of the subject. For instance, some of the topics discovered in the research ranged from easy to understand fields like soccer to more complicated or non popular fields like space technology and aviation and marine news. Different languages' words also proved a challenge in that the research team only had an understanding in English. Having background knowledge in cricket, soccer and African news allowed me to easily identify topics' meaning and identify irrelevant words in the topic. Some of the words were found to be affected by preprocessing in that some crucial characters were removed and the words lost the real meaning.

Saving of the crawled web page data as well as the document matrix, topic term matrix and topic document matrix linked by a crawling and plsa implementation batch tables showing the statuses for crawling and PLSA implementation meant running the reports against these matrix tables in the MySQL table was easier to process and all the historical batch testing done was kept as record. Indexes were created on the MySQL database to speed up the search of the data from the crawling data table as well as doing inserts in the topic document and topic term matrix tables.

The parameters affecting the PLSA algorithm include k, the number of topics, n, the number of documents in the documents matrix and i, the log likelihood differential parameter. The research deduced that as the value of k increases, the quality of the topics above a certain threshold, most of the topics produced do end up with lots of irrelevant words. For a document set of 200, with a k-value of 20 was found to have better values and the best topic quality as opposed to when the algorithm was run with k-values of 10 and 25. Lower values of k like 5 were

also found to reduce the quality the topics as well. The higher the topics also increased the number of iterations as well as the processing time of the Expectation Maximization algorithm. The number of documents n , increased the time required to generate the document matrix as well as needed more iterations to get to the convergence value as well and due to the fact that there were more documents and the vocabulary had increased, the quality of the topics was also found to increase albeit slightly. The change in i, the value of the log likelihood convergence was found to cause better topic quality when it was reduced from 0.000001 to 0.00000001, due to the fact that the number of iterations to convergence increased. For high values of i, the number of iterations done was very low hence the quality of the words in the chosen topics and documents in the topics was found to be of poor quality. The following graph shows the number of iterations based on the number of topics chosen with the red line having 202 documents and the black line having 765 Documents. Thus the as the number of topics increases the number of iterations also increase and also as the number of documents increase the the number of iterations will be on a higher curve.

Figure 6.10: Number Of PLSA Iterations By Topic

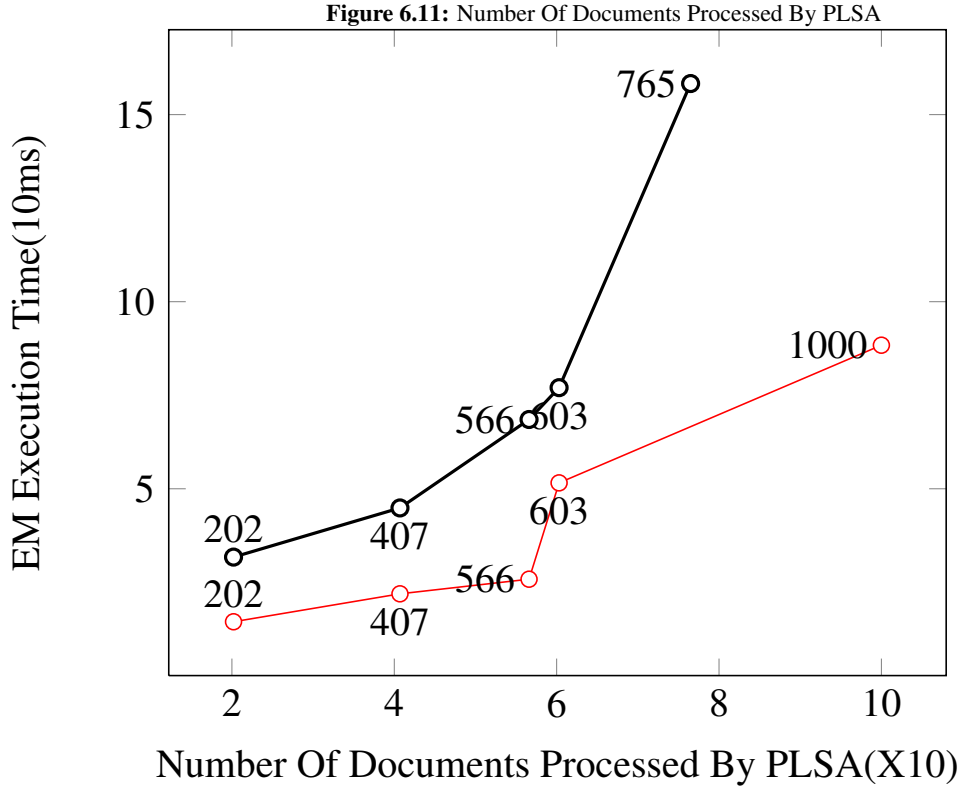


6.7.1 Performance bottlenecks

Processing of the algorithm without multi-threading brought challenges in prolonged processing times and in cases of documents greater than 700, the machines ran out of java heap and CPU. The response time for large document sets was mostly on probability initializing and building document term matrix which was huge and took huge

memory to stay in memory as the EM algorithm. To simplify, and reduce memory, the program wrote all the output of the EM algorithm to a database and to find out the topic and terms reports, queries were run against these tables to filter the top 30 terms and documents per identified topic filtering in order of probability of z_w and z_d in descending order. The following graph shows the red line graph where k=10 and black line graph where k=20 detailing the impact of increasing the number of documents showing that the total processing time for the EM algorithm increases almost exponentially as the number of documents approaches 10006.11.

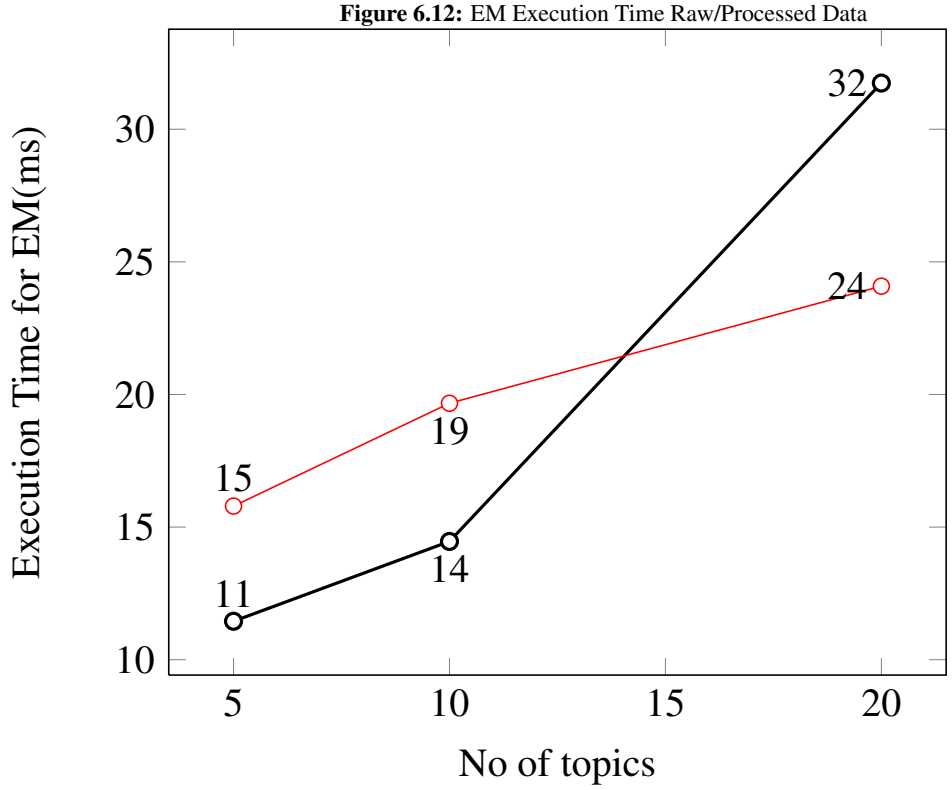
Figure 6.11: Number Of Documents Processed By PLSA



6.7.2 Raw Data against Processed Data

The performance of classifying web pages without processing them by removing keywords or stop words produced more correct topics with less meaningful words as compared to topics that came from the processed data. This could be attributed to the fact that there was no information loss. The raw or unfiltered documents classification took longer than the processed words though due to the fact that their vocabularies were much bigger than the raw data. The tests on raw data and processed data showed that the quality of topics derived from the raw data to be of better quality than the processed data. This is attributed to the loss of data and meaning of some words during preprocessing. The downside of the raw data though is the increase in processing time as shown in the next diagram 6.12.

Figure 6.12: EM Execution Time Raw/Processed Data



6.7.3 Time changing web data

One of the issues the research also found is that topic identification in fields which change a lot for instance politics, sports or general news where what applied a week ago might not necessarily be the same today. For instance the topic of Arsenal players on some articles included the current players but a month ago some players like Podolski left to join Inter Milan hence they no longer fit in that topic. The problem is that the due to the fact that topic identification does not consider date the article was produced on the web then doing topic identification with all the articles is not necessarily correct for topics where situations change a lot.

6.7.4 Topic identification is still judgmental

Whilst the research identified the topics through own background knowledge of the topics found, this still remains a subjective assessment and is a disadvantage of the algorithm as compared to other text mining algorithms. The iteration also is an approximation method hence the value in reality is not the exact value but a very close value.

6.7.5 Problems in source data

One of the major problems noted in the source data included lots of irrelevant web pages. The impact of this is seen on running the algorithm as some of topics have irrelevant words mostly due to the fact that there some couple of web pages downloaded that contain advertising content and chat or social content. Whilst the research did exclude content from major social networks like Facebook and Twitter, the small social sites' content did find its way into the list of crawled data used by the PLSA algorithm. Not only did this social network chat data present problems like informal language use, it also brought problems of analysing content from unobjective people not qualified as web publishers and such content was more of individual opinions

6.7.6 Different Languages on the Internet

One of the problems the research was downloading British English based websites and American English websites. The two website types' languages which interpret some words differently resulted in one word being treated as two different words in the PLSA algorithm with words like 'organiser' and 'organizer' being treated as two different words. It was left up to the researcher to use base English knowledge of the two, to be able to identify such scenarios in a manual way. Some words which came in as Spanish and Portuguese brought challenges as the research was not skilled in speaking such languages. There were also several cases of Indian and Eastern Europe words in cases of football and cricket articles which were difficult to identify as people's names or meaning some activity in the respective language.

6.8 Summary

The chapter summarizes the experiments carried out in the research with huge sets of data against the PLSA's Expectation Maximization algorithm. Whilst topic identification came out with with positive results in both topic terms and topic document classification, processing time as the documents exceeded 500 documents was a major concern hence the need to implement another way of running the iterations in parallel using technologies like Hadoop and Map Reduce. The output of the algorithm was written to a database and analysis of the results included extracting the top 30 words with the highest probability. Error logging and tracing even though it helped in identifying the more heavier parts of the algorithm also introduced a performance overhead and for huge document sets above 700, logging and tracing had to be disabled. The next chapter summarizes the findings of this research and makes recommendations of how the algorithm and topic identification in web data can be improved based on the results of the data found in this research.

Chapter 7

Discussions, conclusion and recommendations

7.1 Introduction

This chapter attempts to come to conclusions on the objectives of this research. The first past summarises the research findings to see if they have been achieved or not. This is both looking at both the milestones the research was looking which includes dirstly, crawling to download web data for classification and the secondly, implementation of the PLSA algorithm to analyse the crawled data. The research looks at the benefits and challenges of the PLSA algorithm as well as the crawling approach followed with focus on the challenges experienced during implementation as well as design. The chapter will look at future research suggestions and then a research conclusion will follow.

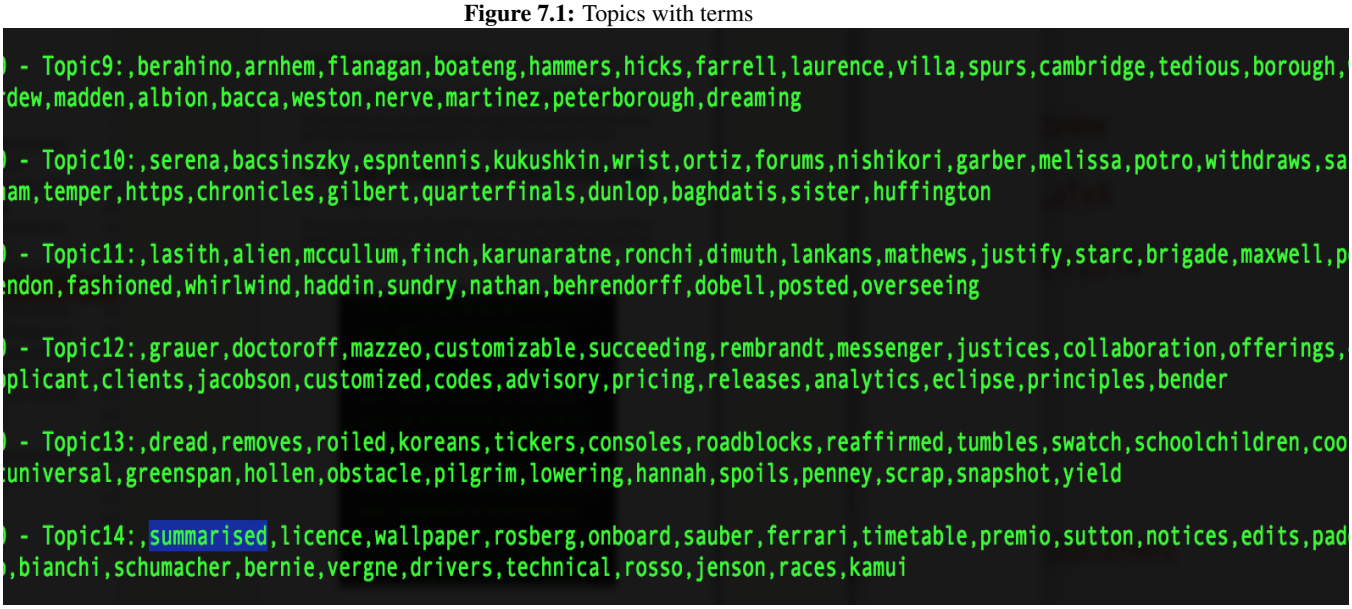
7.2 Summary of the research findings

The research concludes that PLSA is a good generative model which can be used to extract topics from a document collection. The research concludes that even if there was no word ordering, the PLSA algorithm still provided very good topic extractions from the documents supplied. However as evidenced by the implementation in this research PLSA is a highly computationally complex and from a resource perspective requires huge ammounts of processing memory and central processing unit resources. Whilst program deduces the topics, it still remains the duty of the research to do manual identification and verification of what each topic means and means. This means there is need for human involvement to do the topic verification and interpretation where the respective persons have to have broad knowledge of these topics in order to do proper identification. The research answers some of the key objective questions based on the results from this research.

7.2.1 Can we use PLSA algorithm to classify web documents and identify hidden topics?

The PLSA algorithm can be used to identify hidden topics through Expectation Maximisation algorithm based on evidence of this research's experiments. However there is need to understand the domain knowledge so as to validate if the generated topics are relevant. The research found interesting topics like below where cricket, british footballers, tennis players and formula one racing topics were found.

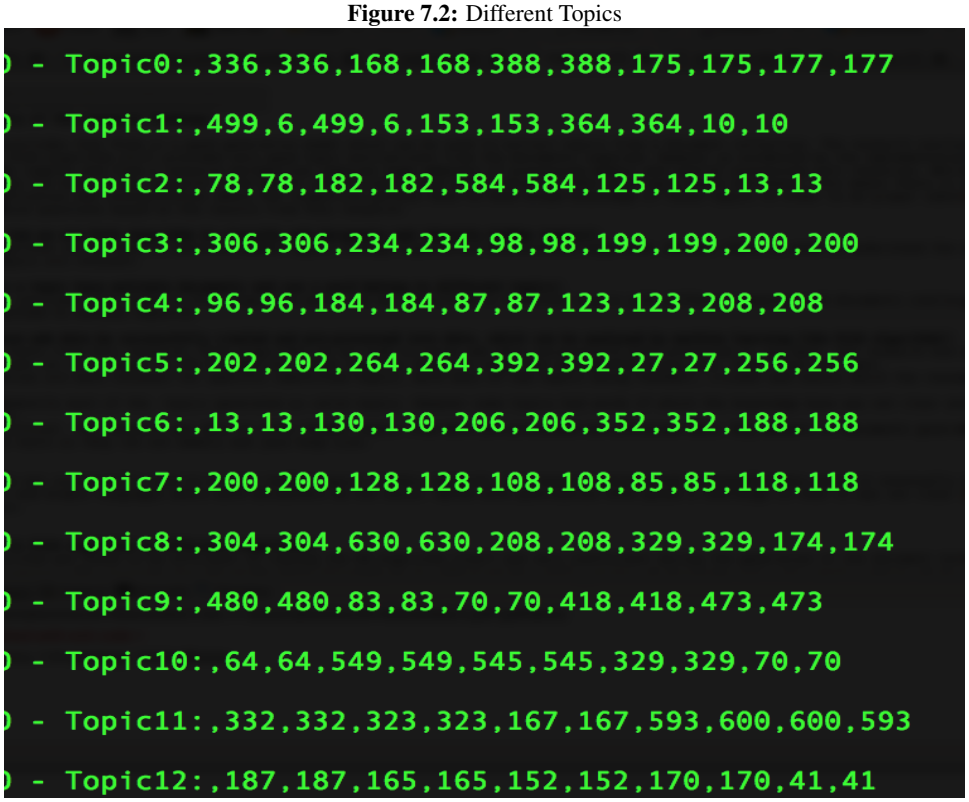
Figure 7.1: Topics with terms



7.3 Can a topic have multiple documents and can a word belong to different topics

The research concludes that based on PLSA algorithm words can be classified in different topics with different meanings and documents similary can be classified to various topics. Document 200 was found to belong to topic 7 and topic 3 as shown.

Figure 7.2: Different Topics



7.3.1 Can web data be successfully crawled and pre-processed into data, which can be analysed by machine learning like PLSA algorithm?

Based on the tests from this research, PLSA algorithm can be used to parse web data successfully both processed or raw data. The research did successfully download web data, pre-processed it and ran it as input to the Probability Latent Semantic Analysis algorithm implemented using java. Topic term and topic document matrices are both relevant for specific identified topics. With most of the topics being football, cricket and tennis which the research fully understands as a domain, the research was able to identify most of the topics generated as valid topics. However some topics had words of which the knowledge base was not clear and unknown, which means selection of initialisation domains are critical. The research was able to infer hidden topics from both pre-processed and processed data. Big amounts of documents generated massive amounts of vocabulary which caused failed tests as they ran out memory and java heap size like test 7 and test 8.

Whilst web data was successfully used to extract hidden topics in this research, the research encountered unsuitatable data which it eventually did not consider for classifying that included Asian and Arabic languages where word delimiters do not exist. Words with punctuation also posed a challenge in that it was not clear if this was the end of a sentence or an abbreviation.

7.3.2 Is PLSA algorithm an efficient algorithm?

The PLSA algorithm was found to be efficient in running the EM algorithm itself and very inefficient during the generation of the document term matrix and probability initialisation phase of the process. The research also found the PLSA algorithm to be efficient for small volumes of documents only. For small numbers of documents the PLSA algorithm performed well the 5 major subcomponents, namely generating the document term matrix, initialisation of the probabilities, running the expectation step, running the M-Step, running the calculate Log Likelihood function till convergence, and then normalization before topic terms and topics documents are generated. For documents sets of 200, the server being used for testing showed very good response times. The quality of documents and terms belonging to each term were also very relevant with only around 10% terms/words being irrelevant to the topics. For large document sets of 700 and above most of the tests failed with the reason being the server running ran out of java heap size and CPU shot over the roof to around 100 per cent. The research had to increase to 4GB and 5GB for the minimum and maximum java heap size for the algorithm to run to completion.

7.3.3 PLSA algorithm works well on processed data than on raw web data?

The research proved that the PLSA algorithm gives more accurate data on raw web data than pre-processed data. Of the tests carried out the research proved that raw data even though had a bigger vocabulary size resulting in longer processing time, had less words and documents, which were irrelevant to the topics in the topic term and topic document matrix. The reason that the research found was that preprocessing did remove some relevant words(information loss) and also stripped some words from being meaningful words to non-meaningful words. Some of the stop words removed were found to have a meaning in the identified topics as evidenced by their appearance in the top 30 words of topics when raw data was used as input to the PLSA algorithm.

7.3.4 Does number of documents, value of topics or iterations matter?

The number of documents and topics and iterations do affect the processing time of the algorithm as well as the quality of the topics. The other major factors affecting the PLSA algorithm include k the number of topics, i the value of iteration cut off. As k-value increase processing time of the EM algorithm increased and some topics returned had poor quality words. The number of topics also increases the number of iterations as well which will lead to more improved topic quality. The research discovered that as the number of documents increase, the processing time especially the building of the document term matrix is increased and there is high chance of the server resources running out and the algorithm failing to run to completion.

7.4 Crawling challenges and benefits observed

Some of the challenges experienced during crawling include the fact that only http links are supported by the implementation. HTTPS, FTP and other web technologies are not catered for due to the security, SSL and login challenges as well as privacy concern.

7.4.1 Wrong selection of implementation language

After experiencing numerous performance problems like out of memory issues and network timeouts using the java program, the research is of the opinion that a wrong selection of language was used. Perl or python would have been viable and simpler alternative web crawlers using less memory and CPU resources.

7.4.2 High number of rejected website Urls

The research found that a huge number of websites were rejected by not meeting criteria and most of these websites did not meet the criteria used to filter valid URLs hence they failed.

7.4.3 Database storage problems

The writing of the web data was fraught with lots of problems like java database connection errors. Truncation of the data also took place since blob or binary large object would truncate some of the longer web pages.

7.4.4 Language Detection Web Service

The crawling preprocessing used a language detector web service by calling a REST or Representation State transfer technology service. This only introduced more processing time for the crawling methodology in that the REST web service being called was a remote call hence there was time delays, timeouts as well as dependency on high availability on an outside system by the implementation.

7.4.5 New data formats flooding the Web

The 21st century has seen various formats coming to the web with 'XHTML, microformats, DC, RSS, Podcast, Atom, WSDL(Web Service Desciption Language), FOAF(Friend of a Friend), RDF(Resource Description Format)', XSLT(Exensible Style Sheets) and Extensible Markup Language(XML). The research did not have a crawler with capabilities to crawl such data formats hence it only restricted itself to HTML formats which were simplified in nature. This means the research did miss a lot of data with such formats and accuracy of the PLSA topics was heavily compromised.

7.4.6 Crawling Algorithm Errors

Java Out Of Memory Errors

Due to the high number of websites being crawled the java processes running on a JBOSS application server ran out of memory and CPU ran high on most occasions. Most of the errors listed in the Appendix B indicate most of the times errors were as a result when downloading huge number of webpages and the machine used which was running on 16GB on RAM failed to cope.The research then had to allocate minimum and maximum java heap memory sizes of the 4GB and 5GB respectively and response times improved greatly.

HTTP 4xx Errors

All the websites which were successfully downloaded returned an HTTP code 200 to show successful response from the website the crawler was downloading the website but there are instances when HTTP 4XX errors were received due to various reasons. The HTTP error code 401 was received when the web page needed authentication due to security but the crawler was not able to reach it hence it returned this 401 HTTP error. A 404 page not found was found on various cases to indicate that the page was not a dead link due to the web page being removed but its associated links to it was not being updated.

Database jdbc connection errors

The java application crawler on various occasions also failed to establish connection to the database used to store the crawling data and threw java database connection errors. This was caused mostly when either the database was not restarted, in cases when the research server had been restarted and in cases when the java application engine, JBOSS ran out of connection threads to connect to the MySQL threads. The number of connection threads was then increased on the JBOSS application server to 10 and this reduced the frequency of this error as well.

Network Errors on Crawling

Some of the websites failed to resolve the Domain Name Service of the website URL or timed out in their response when the crawler was running and this was mostly observed when the internet was throttled and during busy hours of the day like during midday. When crawling was done in the night this reduced this error greatly.

Irrelevant pages and Dead links

The research found lots of dead links and irrelevant data due to dead links on most static web pages. The research also did not cater for spelling errors of most websites hence considered the web data as is as well.

Preprocessing loss of data

The removal of all words less than 3 words meant the research lost lots of data hence the accuracy of the PLSA algorithm was affected. The research also had to remove all non-word characters and this meant a word like 'isn't'

was reduced to isnt, which in some cases means a different meaning in the English vocabulary.

7.5 PLSA challenges and benefits observed

PLSA used in this research has well defined probabilities. It is a better model selection and complexity control. Documents are not related to a single cluster and allows effective modelling. Probabilistic Semantic Analysis since it is based on the likelihood principle, defines a generative data model, and reduces word perplexity. Probabilistic Latent Semantic Analysis (PLSA) also has a good statistical foundation and uses the function of learning and memory to improve algorithmic accuracy. The major problem is that the model can not add a new term or a word not seen in the training set without going through another PLSA implemetation run or reclassification of the data again.The other problem is that updating the model requires running over the complete training and folded-in data which is not time efficient. PLSA is prone to overfitting as the number of documents approaches infinity. From this research, results that were obtained showed that the PLSA algorithm relies too heavily on the frequency of the terms in each document, which is the number of times a word is found in a document.

7.5.1 PLSA is training data dependent

PLSA depends on how the training model data was created. The types of documents used in the training model should mirror the type of documents the model will compare. Generating the model with few documents will not create a good meaningful sense but also using almost similar documenents will result in a model latent topics very sensitive to minor changes to the model. .

7.5.2 Computational Intensive and High Network Utilisation

As seen from this research PLSA algorithm brings high computational costs especially when not using multithreading environment. The biggest tool PLSA requires is using map reduce model per iteration. Due to the high network input output overhead map-reduce iteration. The challenge also is that Map-Reduce also create memory problems hence it will still bring perfomance overheads. PLSA is also a very complex algorithm and this has reduced its use in the general industry.

7.5.3 PLSA is Memory Intensive

The biggest problem encountered during the EM algorithm execution included out of memory errors due to the java heap and memory running out of space. The research changed the values of the heap sizes by increasing the value of -vmargs -Xmx256M to 4GB.

7.5.4 PLSA initialisation problems

Parameters of the PLSA model are trained based on the Expectation Maximization (EM) algo- rithm hence the output trained model's outcome will be heavily dependent on how the initialisation values were created. The

likelihood function will increase till with Expectation Maximisation iterations till it reaches the local maximum value as opposed to the global maximum. Thus the solution is heavily dependent on how the starting data was initialised.

7.5.5 Computational Complexity

Whilst for LDA its numerically complex it will calculate the exact values, PLSA uses EM algorithm to approximate the most likely or maximum likehood hence it is still an approximation.[16]. The relevant probability distributions are found by selecting the model parameter values that maximize the probability of the observed data which is in this case the likelihood function(MLE).

7.5.6 Polysemy Words

PLSA algorithms doesn't cater for polysemy words but this can be mitigated by using a word sense disambiguation tool on the input data so that each word can be be tagged in a sensible way.

7.6 Future Work for improved Crawling for data collection

The research recommends various strategies given the problems encountered in this research. To counter lots of irrelevant pages the research recommends using Genetic Algorithm to do focused crawling. To improve crawling speeds the research recommends using distributed crawling either with geographically located servers or multi-servers with load balancing capabilities. Separating URL fetching and web data downloading into separate threads is also recommended so is using web caching technologies to cache already fetched urls. The research also recommends adding ontological capabilities to the crawling framework to take advantage of the Resource Description Framework.

7.6.1 JVM Errors during crawling

Out of memory was the major bottleneck of the crawling implementation in java including timeout errors of websites with slow responses. Whilst implementing multithreading to resolve speed up crawling did put the little resources under strain, the research had to increase the minimum and maximum java heap size to 4GB and 5GB respectively. A more robust solution would have been to increase the CPU processing power as well as moving the data structure of the visited websites from a java bean object to a persistent data object in the database thus putting less strain on the java application engine. The research removed all the 'System.out.println' statements, which also consume huge memory with a logger Log4J as well as using AspectJ to track performance of the algorithm. This also helped to have cleaner code since all logging to do with exceptions, crawling, and PLSA text classification were written to separate logs.

The research recommends a more robust crawler implementation using Perl or python because not only does it reduce the complexity by having fewer lines of code, it also simplifies code changes as opposed to the java implementation used with resources.

7.6.2 Improving Focused Crawling With Genetic Algorithms

The research did not benefit from a blanket crawling approach in that a huge number of websites either returned irrelevant data or some websites were not relevant to the topics required. The research recommends a focused or topical crawler based on a genetic algorithm. Genetic algorithms or GA are search based algorithms founded on the principle of genetics and natural selection where principle of survival of the fittest is used to produce better approximations of the results desired[35]. The new generation selected would then have better individuals compared to their ancestors[35]. A generic-algorithm web crawler would then include intelligent agents browsing the web using user queries mimicking human browser behaviour and then this will in turn be used by the algorithm to focused crawling traversing only the relevant pages of the given topics by changing weights and page ranks[35].

7.6.3 Increase Speed of Crawling by Maximum Size Threshold

The crawler can also download a page in batches of bytes for instancw 24K, followed by another 24K and till all the bytes are downloaded which improve speed though this would lead to risk of information loss. Slow websites can also be blacklisted to avoid them in the future.

7.6.4 Distributed Crawling using parallelization policy

Running parallel processes or threads in crawling improves crawling greatly especially for major well known crawlers like Google and Spybot[9]. Whilst the research did a partial implementation by running 5 java threads to crawl data concurrently the research recommends for bigger crawling requirements, running a group of distributed crawlers where the crawler servers are geographically distributed and replicate their data across each other with the server closest to the website to be crawled being used to crawl thus benefit in terms of network speed and also security risks. The research also recommends running a crawler with multiple servers in an intra-site parallel crawler to benefit from load balancing and fail over capabilities in case on of the crawlers is having performance challenges.[9]

7.6.5 Crawling data background uploading

A more revolutionary implementation of the crawling algorithm would be to download crawling data in the background whilst the url retrieval still takes place to fetch the subsequent urls and this will improve performance. Thus as the separate threads can be used to crawl for new links and downloading of the data allowing them to run separately. The research also recommends using a more distributed and failover enabled database like DB2 Enterprise Edition or Oracle Database 11G which would allow for bigger data storage capacity and improved performance. For the analysis, the research since it had thousands of data would fare much better with indexing the given tables to improve searches of the data during analysis.

7.6.6 Persisting data in database rather in java objects

The research used java beans to persist crawled urls and another list of crawled urls, which means the java engine will persist the data for longer periods when the crawling is still running and this causes the java heap to fill up and CPU and memory on the java server to degrade performance. A more robust model is to persist the data on the database instead and this improve performance greatly though the java application engine to database engine will trigger a small volume java database connectivity timeouts and slow responses. .

7.6.7 Caching Results

Instead of also using java persistency frameworks like JPA or java persistence framework, a more better crawl batter would be to cache requisite data like the whole URL web pages and child pages in a database data and these will. Crawling is also improved greatly by caching in a dynamic subset of already seen URLs using caching

technologies like static cache, Least Replaced Unit(LRU)[6], infinite cache and random replacement cache. Major caches implemented in the java world include the use of terracotta, which is an open source technology as well ad dynacache from IBM. Crawling is also improved by using cache control headers.

7.6.8 Web Crawling High-Quality Metadata using RDF and Dublin Core

Modern faster web crawler combine resource description properties of ontological frameworks like Dublin Core and RDF model (Resource Description Framework) to create a faster web searching mechanism. The web crawler used by Bekett[3] created web crawl RDF/XML records which were indexed based on original RDF/XML subject gateway records.

7.6.9 Network-load reduction

Through downloading compressed web pages by certain web servers, the crawling algorithm can benefit from HTTP compression through better download speeds of the web pages as well as reducing bandwidth utilisation. In this scenario HTTP is compressed before download is done through compression techniques like gzip as well as deflate.

7.7 Future Work for improved PLSA algorithm

Some of the improvements recommended on the PLSA algorithm include implementing a 'Paralleled Probabilistic Latent Semantic Analysis Algorithms' [20] based on Hadoop and Map Reduce Model as well as Improving PLSA Parameter Initialisation with Principal Component Analysis.

7.7.1 Improving PLSA Parameter Initialisation with Principal Component Analysis

The PLSA algorithm is heavily dependent on the initialisation of the Expectation Maximization parameters. For a given initialisation, the log likelihood increases until a local maximum is attained during iterations[16]. The maximum is said to be a local maximum rather than a global maximum hence the initialisation of the model is very critical. Rather than using the Random initialisation approach used in this research based on Hofman, 1999[16], the research recommends adopting a combined LSA-PLSA combo approach where Latent Semantic analysis is used to initialise the PLSA model data since this has been proven to have less computational costs and better performance with smaller number of latent variables.

7.7.2 Paralleled Probabilistic Latent Semantic Analysis Algorithms Based on Hadoop and Map Reduce Model

The research also proposes a hadoop and map reduce implementation of a parallel Probabilistic Latent Semantic Analysis (PLSA) algorithm when it comes to a large dataset of 100 000 documents or more will see improved computing speeds with hadoop[20]. The traditional PLSA Expectation-Maximization (EM) algorithm is will estimate and identify latent topics in serial fashion whilst the parallel PLSA algorithm will implement the EM algorithm in parallel[20]. Whilst the traditional implementation has an Expectation step followed by the Maximization step iterating based on the new log likelihood values till convergence, the PLSA implementation implemented based on Hadoop will use the Map function to adopt and perform the E-Step and the Reduce function to perform the Maximisation Step. The challenge in this implementation though is that all intermediate results of the Expectation Step have to be sent to the Maximisation step and there will be huge transfers of data from Expectation step to the Maximisation step thus overburdening the network further[20]. Another recommended version of the PLSA algorithm would be to perform the Expectation and Maximisation steps simultaneously this reducing the data transfers between the two steps[20].

There are many implementations of Map Reduce. One of them is the Apache Hadoop project that is an Apache's Open Source implementation of Google's Map Reduce parallel processing framework. Running Hadoop on a cloud means that we have the facilities to add or remove computing power from the Hadoop cluster within minutes or even less by provisioning more machines or shutting down currently running ones.

7.7.3 Calculate plsa algorithm perplexity

The research should have used perplexity calculation to measure prediction of the PLSA algorithm. Low perplexity means would mean the model is a better topic predictor and thus a better model[16].

7.8 Conclusion

The research conclusion chapter made conclusions that PLSA algorithm is a good estimator and can be used to find hidden topics in list of documents with textual data with reasonable accuracy. Even though it is computationally intensive and complex in its design, PLSA enabled the research to group large groups of documents into topics using terms or words as well as classifying documents into multiple topics as well. Various problems affect PLSA starting with the random initialisation of the data having an impact on the eventual model derived which could be wrong if the initialisation is done wrongly. Other problems include the need to then using human knowledge still identify what kind of topic has been identified means the players involved would now need to have very good domain knowledge of the topics chosen as well. PLSA does not cater for polysemy words and does not account for time dependent data. The research also concluded that crawling data from the web is a good sample of data which can be classified to reasonable accuracy by the PLSA algorithm.

The research also concludes that crawling does bring its own challenges in that most of the data on the web is either old, redundant and either contains dead links. Social network and blogging data also brings challenges to crawling in that these are words which are informal and not expert people's words and not objective as well. Crawling is also computationally intensive and Map-Reduce algorithm is recommended to speed up crawling as well as using parallel crawling algorithms.

Several recommendations were also made including the need to use Map Reduce algorithm for both PLSA classification as well as crawling. Improving PLSA parameter initialisation with Principal Component Analysis is also recommended so is using PLSA and Latent Semantic Analysis algorithm together to get the benefits of both. The research also recommends using caching algorithms and tools like terracotta during crawling as well as persisting most of the data to reduce perfomance and network overheads as well.

Bibliography

[1] Gianni Amati and Cornelis Joost Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, 20(4):357–389, 2002.

[2] Bosch Anna, Zisserman Andrew, and Muoz Xavier. Scene classification via plsa. In Ale Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision ECCV 2006*, volume 3954 of *Lecture Notes in Computer Science*, pages 517–530. Springer Berlin Heidelberg, 2006.

[3] David Beckett. Web crawling high-quality metadata using rdf and dublin core, 2014.

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[5] Andrei Broder. A taxonomy of web search. *SIGIR FORUM*, 36(2):3–10, 2002.

[6] Andrei Z. Broder, Marc Najork, and Janet L. Wiener. Efficient url caching for world wide web crawling. In *In Proceedings of the twelfth international conference on World Wide Web (WWW2003*, pages 679–689, 2003.

[7] Henri Chen and Robbie Cheng. *ZK Step-By-Step: Ajax Without JavaScript Framework*. Apress, Berkely, CA, USA, 2007.

[8] Junghoo Cho and Hector Garcia-Molina. The evolution of the web and implications for an incremental crawler. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 200–209, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[9] Junghoo Cho and Hector Garcia-Molina. Parallel crawlers. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, pages 124–135, New York, NY, USA, 2002. ACM.

[10] Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. Efficient crawling through url ordering. *Comput. Netw. ISDN Syst.*, 30(1-7):161–172, 1998.

[11] Ayesha Choudhary, Manish Pal, Subhashis Banerjee, and Santanu Chaudhury. Unusual activity analysis using video epitomes and plsa. In *Proceedings of the 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, ICVGIP '08, pages 390–397, Washington, DC, USA, 2008. IEEE Computer Society.

[12] Roy T. Fielding and Richard N. Taylor. Principled design of the modern web architecture. *ACM Trans. Internet Technol.*, 2(2):115–150, May 2002.

[13] Yun Chen Flora S. Tsai and Kap Luk Chan. Probabilistic latent semantic analysis for search and mining of corporate blogs. in proceedings of the 2008 conference on applications of data mining in e-business and finance, carlos soares, yonghong peng, jun meng, takashi washio, and zhi-hua zhou (eds.). ios press, amsterdam, the netherlands, the netherlands, 63-73. *Applications of Data Mining in E-Business and Financ*, 4(3):175–246, 2008.

[14] Andreas Harth, Jrgen Umbrich, and Stefan Decker. Multicrawler: A pipelined architecture for crawling and indexing semantic web data. In *In 5th International Semantic Web Conference*, pages 258–271, 2006.

[15] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 50–57, New York, NY, USA, 1999. ACM.

[16] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Mach. Learn.*, 42(1-2):177–196, January 2001.

[17] Ziwon Hyung, Kibeom Lee, and Kyogu Lee. Music recommendation using text analysis on song requests to radio stations. *Expert Syst. Appl.*, 41(5):2608–2618, April 2014.

[18] Jenkov Jacob. Java web crawler implementation, 2011.

[19] Alex Rogers James McInerney and Nicholas R. Jennings. Improving location prediction services for new users with probabilistic latent semantic analysis. in proceedings of the 2012 acm conference on ubiquitous computing (ubicomp '12). acm, new york, ny, usa, 906-910. *Foundations and Trends in Information Retrieval*, 4(3):175–246, 2012.

[20] Yan Jin, Yang Gao, Yinghuan Shi, Lin Shang, Ruili Wang, and Yubin Yang. P2lsa and p2lsa+: Two paralleled probabilistic latent semantic analysis algorithms based on the mapreduce model. In Hujun Yin, Wenjia Wang, and Victor Rayward-Smith, editors, *Intelligent Data Engineering and Automated Learning - IDEAL 2011*, volume 6936 of *Lecture Notes in Computer Science*, pages 385–393. Springer Berlin Heidelberg, 2011.

[21] K. Sparck Jones, S. Walker, and S.E. Robertson. A probabilistic model of information retrieval: Development and status, 1998.

[22] Tuomo Kakkonen, Niko Myller, Jari Timonen, and Erkki Sutinen. Automatic essay grading with probabilistic latent semantic analysis, 2005.

[23] PS Kenkre. Web crawlers. 2010.

[24] Gerald Kowalski. *Information Retrieval Systems: Theory and Implementation*. Kluwer Academic Publishers, Norwell, MA, USA, 1st edition, 1997.

[25] Sangno Lee, J. Baker, J. Song, and J.C. Wetherbe. An empirical comparison of four text mining methods. In *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, pages 1–10, Jan 2010.

[26] Hu Li. 2014.

[27] Chenxi Lin, Gui-Rong Xue, Hua-Jun Zeng, and Yong Yu. Using probabilistic latent semantic analysis for personalized web search. In Yanchun Zhang, Katsumi Tanaka, JeffreyXu Yu, Shan Wang, and Minglu Li, editors, *Web Technologies Research and Development - APWeb 2005*, volume 3399 of *Lecture Notes in Computer Science*, pages 707–717. Springer Berlin Heidelberg, 2005.

[28] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[29] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[30] Melanie Martin and Hofmann T. Latent semantic analysis, 2014.

[31] Ali Mesbah, Engin Bozdag, and Arie van Deursen. Crawling ajax by inferring user interface state changes. In *Proceedings of the 2008 Eighth International Conference on Web Engineering*, ICWE '08, pages 122–134, Washington, DC, USA, 2008. IEEE Computer Society.

[32] Seyed M. Mirtaheri, Mustafa Emre Dinçtürk, Salman Hooshmand, Gregor V. Bochmann, Guy-Vincent Jourdan, and Iosif Viorel Onut. A brief history of web crawlers. In *Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '13, pages 40–54, Riverton, NJ, USA, 2013. IBM Corp.

[33] Christopher Olston and Marc Najork. Web crawling. *Foundations and Trends in Information Retrieval*, 4(3):175–246, 2010.

[34] SM Pavalam, SV Kasmir Raja, M Jawahar, and Felix K Akorli. Web crawler in mobile systems. 2010.

[35] Chain Singh, Ashish Kr. Luhach, and Amitesh Kumar. Article: Improving focused crawling with genetic algorithms. *International Journal of Computer Applications*, 66(4):40–43, March 2013. Full text available.

[36] Hao Sun, Cheng Wang, Boliang Wang, and Naser El-Sheimy. Unsupervised video-based lane detection using location-enhanced topic models. *Optical Engineering*, 49(10):107201–107201–9, 2010.

[37] Hu Wu, Yongji Wang, and Xiang Cheng. Incremental probabilistic latent semantic analysis for automatic question recommendation. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 99–106, New York, NY, USA, 2008. ACM.

[38] Zhi-Yuan Wu and Xue-Zhong Qian. Tag clustering research based on plsi. *Application Research of Computers*, 5:008, 2013.

[39] Yanzan Zhou Xin Jin and Bamshad Mobasher. Web usage mining based on probabilistic latent semantic analysis. in proceedings of the tenth acm sigkdd international conference on knowledge discovery and data mining (kdd '04). acm, new york, ny, usa. *Foundations and Trends in Information Retrieval*, 4(3):175–246, 2004.

[40] WANG Yi. Chinese text classification based on probabilistic latent semantic analysis. *Journal of Gansu Lianhe University (Natural Science Edition)*, 4:023, 2011.

Appendix A

Source Code and Perfomance Results

A.1 Crawling

A.1.1 Implemeation of Logging using AspectJ and Log4J

```
pointcut insertWebDocumentIntoDB() : execution(*
    zw.co.WebCrawling.PLSAWebCrawler.insertWebDocumentIntoDB(..));
before() : insertWebDocumentIntoDB() {
  PropertyConfigurator.configure("crawlingLog4j.properties");
     start4=System.nanoTime();
  log.warn("insertWebDocumentIntoDB for website has started: ");
}
after() : insertWebDocumentIntoDB(){
  PropertyConfigurator.configure("crawlingLog4j.properties");
  end4=end4+System.nanoTime()-start4;
  savedPAgesCounter++;
  log.warn("insertWebDocumentIntoDB for website has ended: "+end4/1000000+" milliseconds
      and "+savedPAgesCounter+"pages saved");
}
```

A.1.2 Fetching Websites and validations

```
if (!url.toLowerCase().startsWith("http")) {
     return null;
  }
if(url.contains(":80")
  return null
```

```
      private final static Pattern MYFILTERS = Pattern.compile(".*(\\.(css|js|bmp|gif|jpe?g"
+ "|png|tiff?|mid|mp2|mp3|mp4|wav|@|avi|mov|mpeg|ram|m4v|pdf|jpg" +
"|rm|smil|wmv|swf|wma|zip|rar|gz))$");

//check if URL has not been downloaded before on the database
public static boolean urlhasBeenVisitedBefore(String url)
{
   return false;
}



/**
     * You should implement this function to specify whether the given url.
     * should be crawled or not (based on your crawling logic).
     */

public static boolean shouldVisitNow(String url)
{
String href = url;
if(urlhasBeenVisitedBefore) return false;
//invalid link
return !FILTERS.matcher(href).matches() ;
}
```

A.1.3 crawling

A.1.4 Saving Web pages

```
  public static void saveWebPageRecord(String url1) {
     try{
     java.sql.Date oldTime=new java.sql.Date(0);
     Document doc = Jsoup.connect(url1).get();
     String title = doc.title();
     System.out.println(title+"---------------------------------------");

     CrawlBO crawlbo1=new CrawlBO();
     crawlbo1.setStrUrl(doc.baseUri());
     crawlbo1.setHashCode(Util.getHashValue(url1));
     crawlbo1.setPageContents(doc.title());
```

```
    crawlbo1.setProtocol(doc.location());
    System.out.println("writeToFile");
    //create crawled file
    //write the contents
    String newData =Jsoup.parse(doc.body().html()).text();
    java.sql.Date currentT=new java.sql.Date(0);
    //crawlbo1.setTimeTaken((int)currentT-oldTime);
    preprocessAndWriteToFile(newData,crawlbo1);


    }catch(IOException e){System.out.println("Exception occured");}
    }

  public static void insertWebDocumentIntoDB(File newFile,File parsedFile,CrawlBO crawlbo1) {
    Connection connection = null;
    PreparedStatement statement = null;
    FileInputStream inputStream = null;
    FileInputStream processedInputStream = null;
    try {
      //File image = new File("/Users/kennedy/Downloads/bpm_err2.txt");
      inputStream = new FileInputStream(newFile);
      processedInputStream = new FileInputStream(parsedFile);
      connection = getConnection();
      statement = connection
            .prepareStatement("insert into webdocs(idwebdocs, hashcode,strUrl,pageContents"
                  +
                      ",protocol,httpResponseCode,downloadTime,originalWebDoc,refinedWebDoc,timeTaken)
                      "
                  + "values(?,?,?,?,?,?,?,?,?,?)");
      statement.setString(1, java.util.UUID.randomUUID()+""+new java.sql.Date(0).getMonth());
      statement.setString(2, crawlbo1.getHashCode());
      statement.setString(3, crawlbo1.getPageContents());
      statement.setString(4, crawlbo1.getStrUrl());
      statement.setString(5, crawlbo1.getProtocol());
      statement.setInt(6, 200);
      statement.setDate(7, (Date) new java.sql.Date(0));
      statement.setBinaryStream(8, (InputStream) inputStream,
            (int) (newFile.length()));
      statement.setBinaryStream(9, (InputStream) processedInputStream,
            (int) (parsedFile.length()));
```

```
        statement.setInt(10, 90);

        statement.executeUpdate();

        System.out.println("don: - " );
    } catch (FileNotFoundException e) {
        System.out.println("FileNotFoundException: - " + e);
    } catch (SQLException e) {
        System.out.println("SQLException: - " + e);
    } finally {
        try {
            connection.close();
            statement.close();
        } catch (SQLException e) {
            System.out.println("SQLException Finally: - " + e);
        }
    }

}
```

A.1.5 E Step

```
/**
 *
 * @param Pz
 * @param Pz_d
 * @param Pz_w
 * @param Pz_dw
 * @return
 */
private Boolean ExpectationStep(double[] Pz, double[][] Pz_d, double[][] Pz_w,
        double[][][] Pz_dw) {

    for (int d = 0; d < numberOfDocs; ++d) {
        int counter = 0;
        for (PositionsAndWords posAndWords: row1[d].getPositionsAndWords()) {
            int w = posAndWords.currentposition;
            double nomarlizer = 0.0;
            for (int z = 0; z < numberTopics; z++) {
                Pz_dw[z][d][counter] = Pz[z] * Pz_d[z][d] * Pz_w[z][w];
```

```
            nomarlizer += Pz_dw[z][d][counter];
         }
         for (int z = 0; z < numberTopics; z++) {
            Pz_dw[z][d][counter] /= nomarlizer;
         }
         ++counter;
      }
   }
   return true;
}
```

A.1.6 Log Likelihood

```
private double computeTheLoglikelihood(double[] Pz, double[][] Pz_d,
      double[][] Pz_w) {
   double loglikelihooder = 0;
   for (int d = 0; d < numberOfDocs; ++d) {
      for (PositionsAndWords pwords : row1[d].getPositionsAndWords()) {
         double sumtotal = 0;
         for (int z = 0; z < numberTopics; ++z) {
            sumtotal =sumtotal+( Pz[z] * Pz_d[z][d] * Pz_w[z][pw.position]);
         }
         loglikelihooder = loglikelihooder+(pwords.nWords * Math.log10(sum));

      }
   }
   return loglikelihooder;
}
```

A.1.7 Crawling Exception Handling

```
            catch (FileNotFoundException e) {
      System.out.println("FileNotFoundException: - " + e);
   } catch (SQLException e) {
      System.out.println("SQLException: - " + e);
   }
   catch (MalformedURLException e) {
      return null;
   }
```

```
    catch (IOException e) {
       System.out.println("Exception occured");
    }catch (Exception e) {
       e.printStackTrace();
       return null;
    }
    catch(OutOfMemoryError e)
  {
     e.printStackTrace();
     return null;
  }
```

A.2 PLSA code documentation

A.2.1 Retrieving Web documents

```
public static List<Document> readWebDocumentsFromDB() {
    Connection connection = null;
    Statement statement = null;
    FileInputStream inputStream = null;
    FileInputStream processedInputStream = null;
     List<Document> allDocuments=new ArrayList<Document>();
    try {
       //File image = new File("/Users/kennedy/Downloads/bpm_err2.txt");
       connection = getConnection();
        statement = connection.createStatement();
          String query = "select * from crawledWebSites";
          ResultSet resultset= statement.executeQuery(query);
        //  System.out.println("Id Name Job");
          int count=0;
          while (resultset.next()) {
          // Then parse the results
            java.sql.Blob myBlob = resultset.getBlob("refinedWebDoc") ;
            java.io.InputStream myInputStream = myBlob.getBinaryStream();
            java.io.BufferedReader in = new BufferedReader(new
                java.io.InputStreamReader(myInputStream));
          //  System.out.println("Id Name Job2");
            List<String> allWords=new PLSAUtilities().extractWords(in);
          //  System.out.println("Id Name Job3");
            myInputStream.close();
```

```
                System.out.println("File saved"+"sss==="+allWords+""+count++);
                 Document doc1=new Document(allWords,"Document"+count);
                 allDocuments.add(doc1);
            }
         System.out.println(allDocuments.size()+"don: - "+count );
      } catch (FileNotFoundException e) {
         System.out.println("FileNotFoundException: - " + e);
      } catch (SQLException e) {
         System.out.println("SQLException: - " + e);
      }
      catch (IOException e) {
         System.out.println("SQLException: - " + e);
      }finally {
         try {
            connection.close();
            statement.close();
         } catch (SQLException e) {
            System.out.println("SQLException Finally: - " + e);
         }
      }
      return allDocuments;

   }
```

A.2.2 Generating Document Term matrix

```
 // Hello.java
public Boolean buildTermDocumentMatrix(int ntopics, double accuracyMeasurement)
       {
        //load all the documents and terms from crawled data
         List<Document> docs1=PLSAUtilities.readWebDocumentsFromDB();
        //assign the document list to the java 2 dimensioanl array as below
           data=new Plsa(5).train(docs1,5);
            nDocs = data[0].length;
            nWords = data[1].length;
            nTopics = ntopics;
            InitializeDataStructure();
            return EM(accuracyMeasurement);
       }
```

Appendix B

Errors and Logging

B.1 SQL Queries used

B.1.1 Crawling Querries

Getting all websites downloaded grouped by seedUrl

Table B.1: PErfomance of PLSA

Getting all websites downloaded	select * from test.crawledWebSites order by startDownloadTime DESC;
Getting all websites grouped by source Url	**SELECT count(*),seedUrl FROM test.crawledWebSites group by seedUrl;**
Getting all the minimum times to download	select MIN(numberOfWords),MIN(wordsProcessedFile), MIN(downloadTime),MIN(timeTaken),MIN(originalFilesize), MIN(processedFileSize),count(*)as Docs,seedUrl from test.crawledWebSites GROUP by seedUrl;
Get Maximum Times	select MAX(numberOfWords),MAX(downloadTime),MAX(timeTaken), MAX(originalFilesize),MAX(processedFileSize),count(*) as Docs,seedUrl from test.crawledWebSites GROUP by seedUrl;
Get Average Times	select AVG(numberOfWords),AVG(downloadTime),AVG(timeTaken), AVG(originalFilesize)/AVG(downloadTime)as downloadRateBytes-perMs,AVG(originalFilesize),AVG(processedFileSize),count(*) as Docs,seedUrl from test.crawledWebSites GROUP by seedUrl;

B.2 Definitions of Performance Metrics

```
PLSA =(2*Recall*Precision)/(Recall + Precision)
Precision
Recall=TN/(TN + FP)
False Positive (FP) Rate
True Positive (TP) Rate
Root Mean Squared Error
```

```
    Mean Absolute Error=1/N
    kappa statistic
    Incorrectly Classified Instances (Rate)=(FP + FN)/N
    Correctly Classified Instances (Rate)   =(TP + TN)/N
```

B.3 Crawling Errors

B.3.1 Network Errors on Crawling

```
 java.net.UnknownHostException: www.espnfc.com
is page content greater than 0..http://www.espnfc.com..

Sleeping for 3 seconds...
   at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:223)
   at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:431)
   at java.net.Socket.connect(Socket.java:527)
   at sun.net.NetworkClient.doConnect(NetworkClient.java:158)
   at sun.net.www.http.HttpClient.openServer(HttpClient.java:424)
   at sun.net.www.http.HttpClient.openServer(HttpClient.java:538)
   at sun.net.www.http.HttpClient.<init>(HttpClient.java:214)
   at sun.net.www.http.HttpClient.New(HttpClient.java:300)
   at sun.net.www.http.HttpClient.New(HttpClient.java:319)
   at sun.net.www.protocol.http.HttpURLConnection.getNewHttpClient(HttpURLConnection.java:987)
   at sun.net.www.protocol.http.HttpURLConnection.plainConnect(HttpURLConnection.java:923)
   at sun.net.www.protocol.http.HttpURLConnection.connect(HttpURLConnection.java:841)
   at zw.co.WebCrawling.PLSAWebCrawler.fetchPageContent(PLSAWebCrawler.java:229)
   at zw.co.WebCrawling.PLSAWebCrawler.crawl(PLSAWebCrawler.java:158)
   at zw.co.WebCrawling.PLSAWebCrawler.main(PLSAWebCrawler.java:63)
No valid URL could be found.
```

B.3.2 Download Timeouts

```
 Fetching page content for URL.http://www.bbc.co.uk/a-z/..

java.net.SocketTimeoutException: Read timed out
is page content greater than 0..http://www.bbc.co.uk/a-z/..

Sleeping for 3 seconds...
```

```
   at java.net.SocketInputStream.socketRead0(Native Method)
   at java.net.SocketInputStream.read(SocketInputStream.java:129)
   at java.io.BufferedInputStream.fill(BufferedInputStream.java:218)
   at java.io.BufferedInputStream.read1(BufferedInputStream.java:258)
   at java.io.BufferedInputStream.read(BufferedInputStream.java:317)
   at sun.net.www.http.HttpClient.parseHTTPHeader(HttpClient.java:709)
   at sun.net.www.http.HttpClient.parseHTTP(HttpClient.java:652)
   at sun.net.www.protocol.http.HttpURLConnection.getInputStream(HttpURLConnection.java:1218)
   at java.net.HttpURLConnection.getResponseCode(HttpURLConnection.java:379)
   at zw.co.WebCrawling.PLSAWebCrawler.fetchPageContent(PLSAWebCrawler.java:232)
   at zw.co.WebCrawling.PLSAWebCrawler.crawl(PLSAWebCrawler.java:158)
   at zw.co.WebCrawling.PLSAWebCrawler.main(PLSAWebCrawler.java:63)

Crawling in loop...http://www.bbc.co.uk/sport/0/football/



java.net.ConnectException
MESSAGE: Connection refused
```

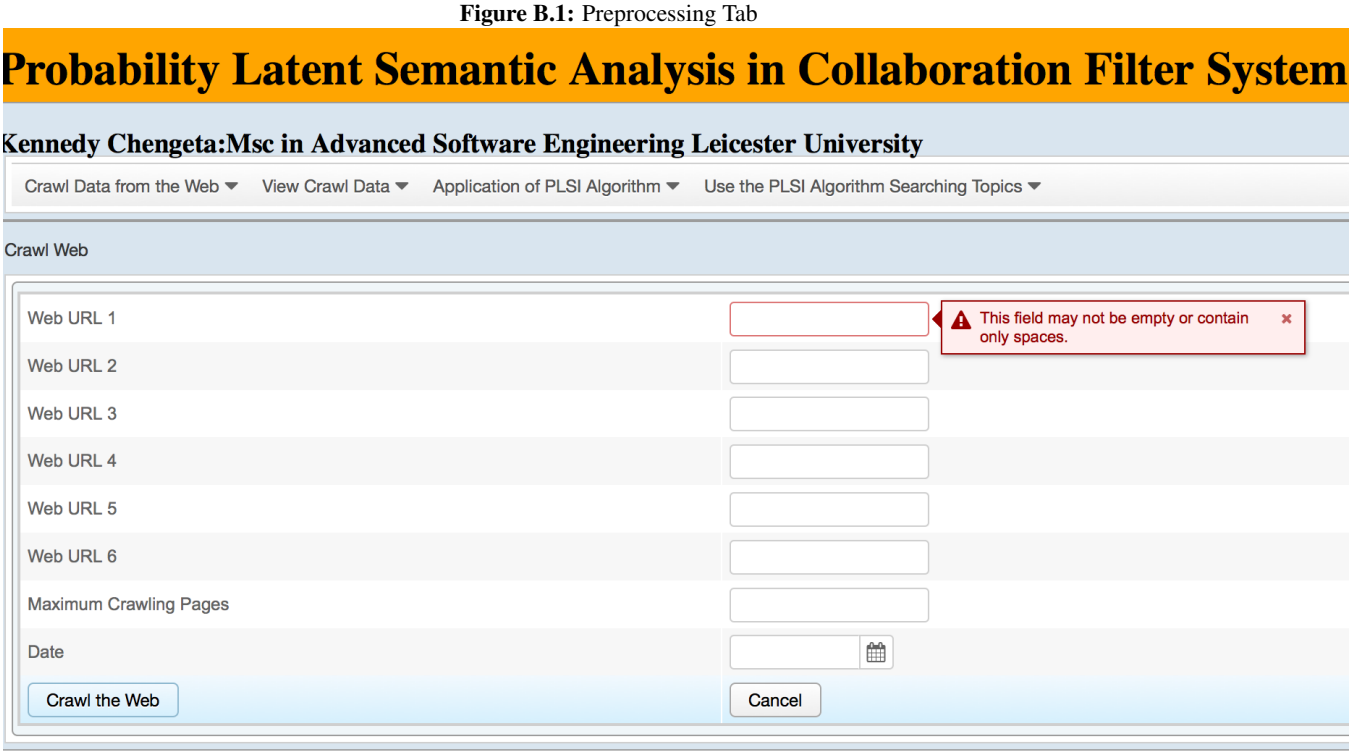
B.3.3 database jdbc connection errors

```
 STACKTRACE:

java.net.ConnectException: Connection refused
   at java.net.PlainSocketImpl.socketConnect(Native Method)
   at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:339)
   at java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:200)
   at java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:182)
   at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:392)
   at java.net.Socket.connect(Socket.java:579)
   at java.net.Socket.connect(Socket.java:528)
   at java.net.Socket.<init>(Socket.java:425)
   at java.net.Socket.<init>(Socket.java:241)
   at com.mysql.jdbc.StandardSocketFactory.connect(StandardSocketFactory.java:256)
   at com.mysql.jdbc.MysqlIO.<init>(MysqlIO.java:271)
   at com.mysql.jdbc.Connection.createNewIO(Connection.java:2744)
   at com.mysql.jdbc.Connection.<init>(Connection.java:1553)
   at com.mysql.jdbc.NonRegisteringDriver.connect(NonRegisteringDriver.java:285)
   at java.sql.DriverManager.getConnection(DriverManager.java:571)
   at java.sql.DriverManager.getConnection(DriverManager.java:215)
```

```
   at zw.co.plsi.PLSAUtilities.getConnection(PLSAUtilities.java:241)
   at zw.co.plsi.PLSAUtilities.readWebDocumentsFromDB(PLSAUtilities.java:75)
   at zw.co.plsi.PLSAUtilities.main(PLSAUtilities.java:58)
```

B.4 Model View Controller Errors-Application Errors

Figure B.1: Preprocessing Tab



B.5 PLSA processing errors

B.5.1 Out of memory

```
 8025 data_d_w:  0:
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
   at zw.co.plsi.WebDocumentsPLSA.init(WebDocumentsPLSA.java:235)
   at zw.co.plsi.WebDocumentsPLSA.EM(WebDocumentsPLSA.java:133)
   at zw.co.plsi.WebDocumentsPLSA.buildTermDocumentMatrix(WebDocumentsPLSA.java:83)
   at zw.co.plsi.WebDocumentsPLSA.main(WebDocumentsPLSA.java:483)
```

B.6 Database Creation SQL Scripts

Create a database named ”Crawler” and create a table called ”Record” like the following:

```
CREATE TABLE IF NOT EXISTS ‘Record‘ (
  ‘RecordID‘ INT(11) NOT NULL AUTO_INCREMENT,
  ‘URL‘ text NOT NULL,
  PRIMARY KEY (‘RecordID‘)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;
```

B.6.1 Starting MySQL Database on MacOS operating System

```
 kennedys-MacBook-Pro-2:log4j kennedy$ sudo /usr/local/mysql/support-files/mysql.server
Password:
Usage: mysql.server  {start|stop|restart|reload|force-reload|status}  [ MySQL server options ]
kennedys-MacBook-Pro-2:log4j kennedy$ sudo /usr/local/mysql/support-files/mysql.server start
Starting MySQL
 SUCCESS!
kennedys-MacBook-Pro-2:log4j kennedy$
```

B.7 Summary

As always, provide a summary at the end.

Table C.1: Crawling Batch Run Table

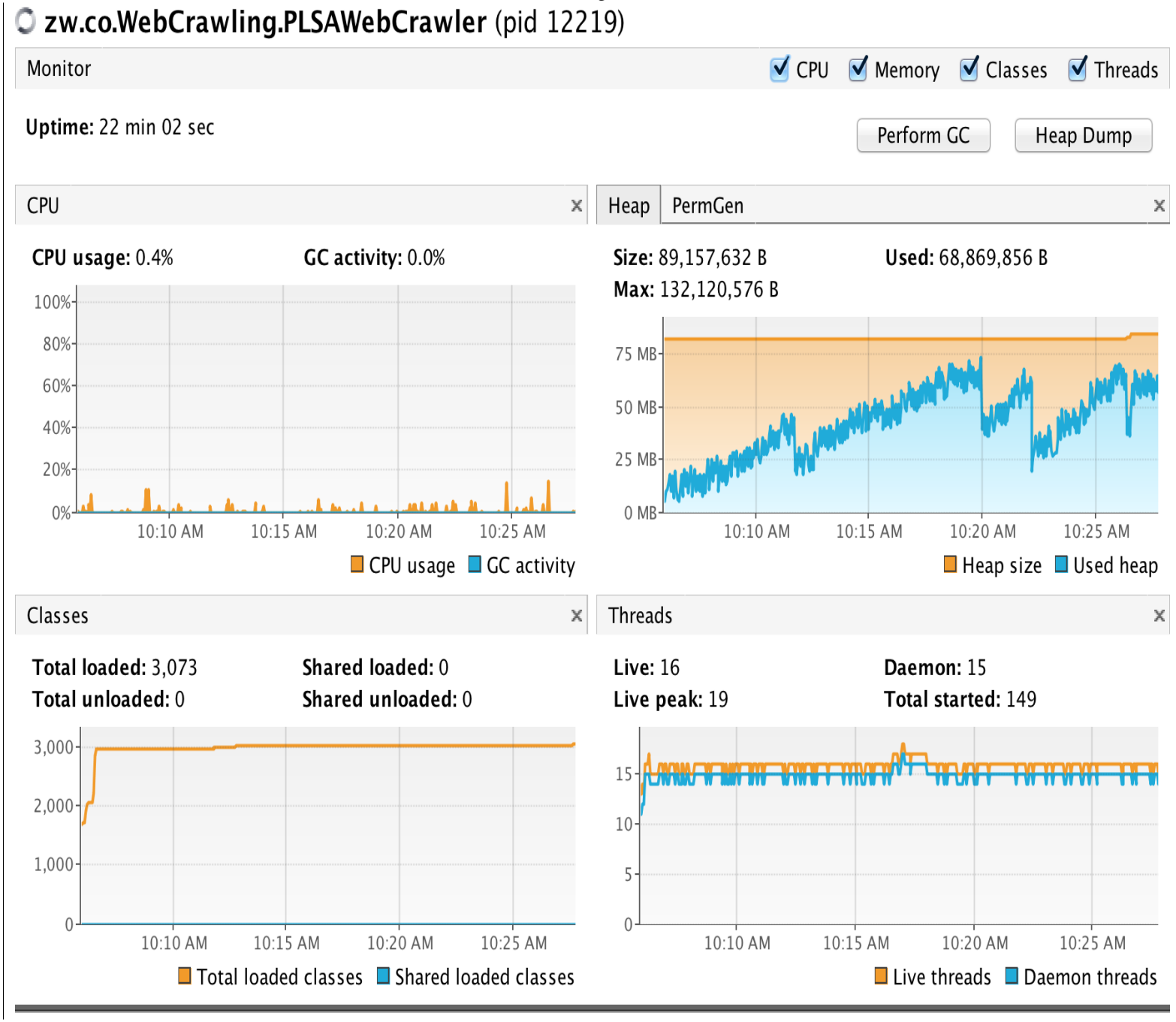


Table C.2: Crawling Batch Run Table

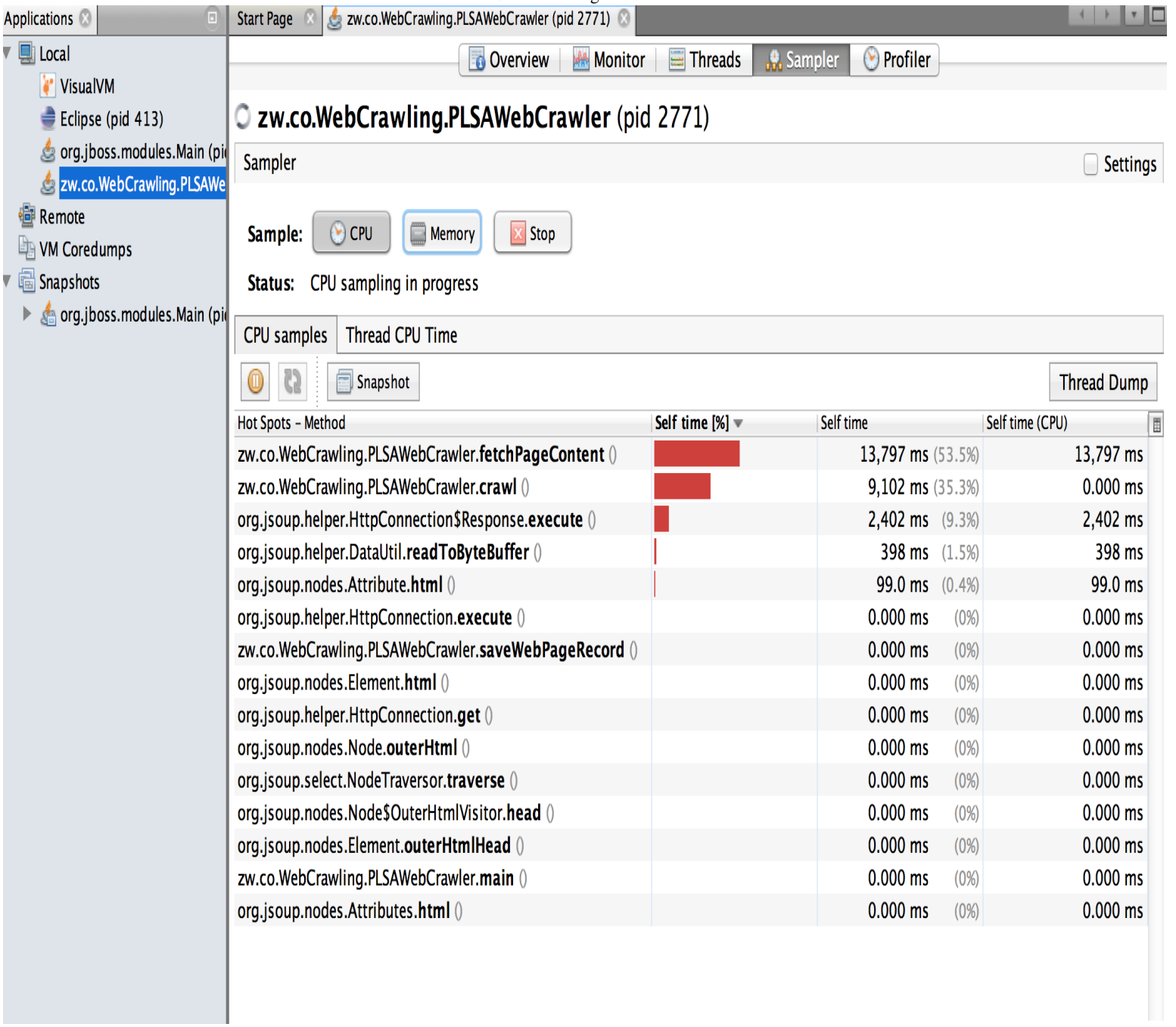
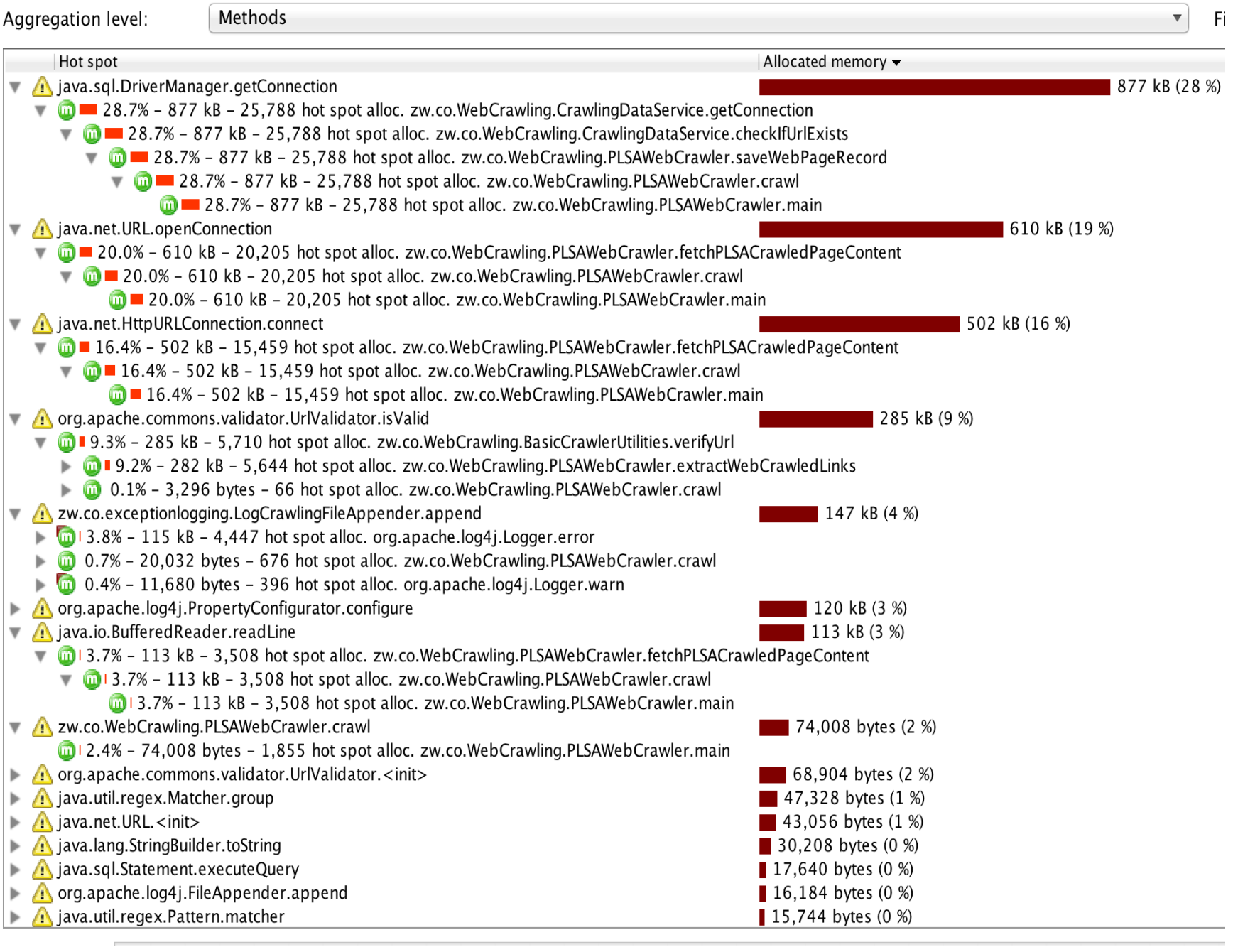


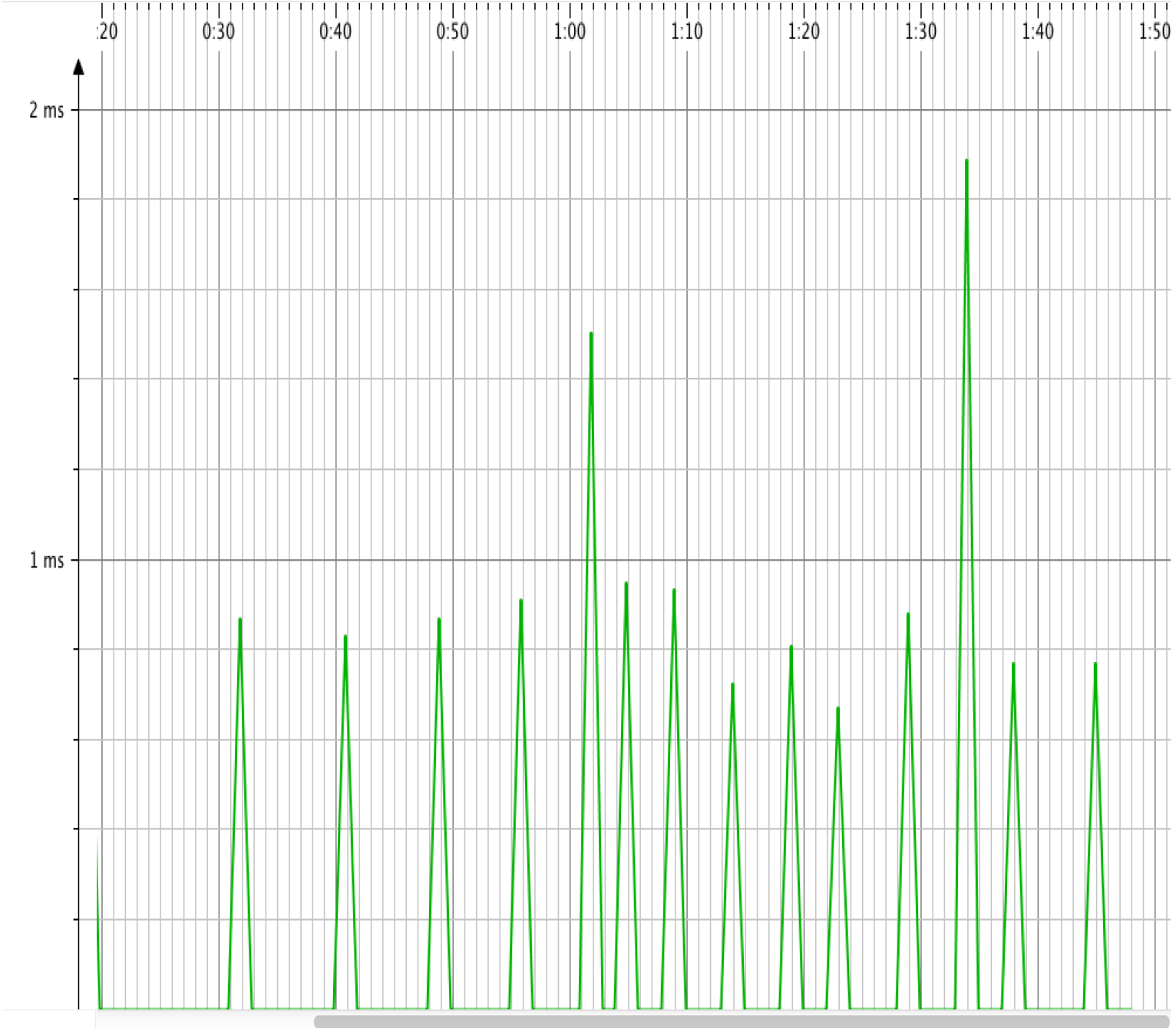
Table C.3: Crawling Batch Run Table



Time Line | Connections | Hot Spots | Telemetries | Events | Tracker

JDBC
JDBC connections and execution of statements

Available probe telemetries: Average Statement Execution Time



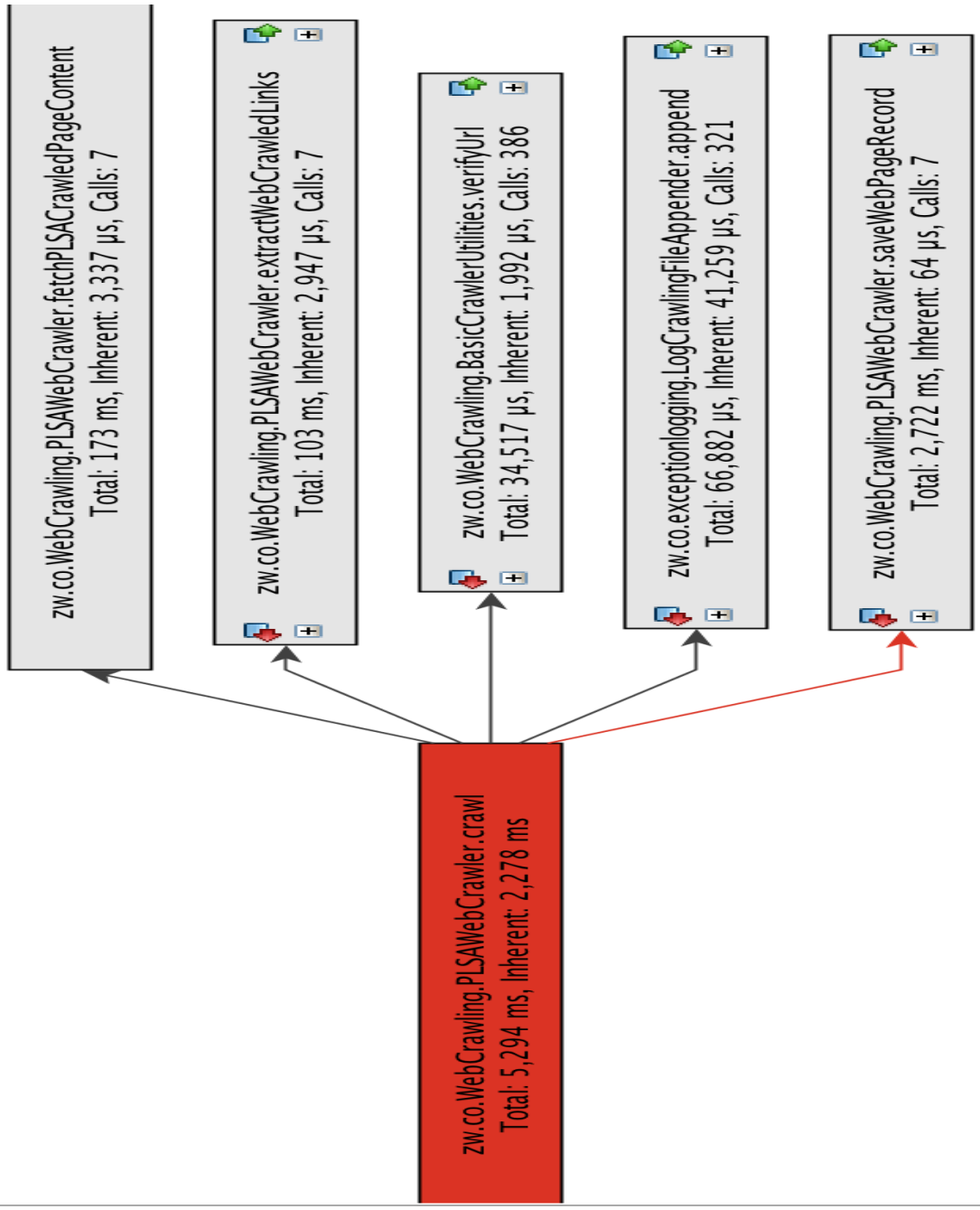

zw.co.WebCrawling.PLSAWebCrawler.crawl
Total: 5,294 ms, Inherent: 2,278 ms
zw.co.WebCrawling.PLSAWebCrawler.fetchPLSACrawledPageContent
Total: 173 ms, Inherent: 3,337 µs, Calls: 7
zw.co.WebCrawling.PLSAWebCrawler.extractWebCrawledLinks
Total: 103 ms, Inherent: 2,947 µs, Calls: 7
zw.co.WebCrawling.BasicCrawlerUtilities.verifyUrl
Total: 34,517 µs, Inherent: 1,992 µs, Calls: 386
zw.co.exceptionlogging.LogCrawlingFileAppender.append
Total: 66,882 µs, Inherent: 41,259 µs, Calls: 321
zw.co.WebCrawling.PLSAWebCrawler.saveWebPageRecord
Total: 2,722 ms, Inherent: 64 µs, Calls: 7

Table 6.5: Drawing Batch Run Table

Show sockets: Both open and closed | Filter by All text fields | Reset Filters

ID	Socket Address	Start Time	End Time	Type	Read Count	Read Duration	Read Throughput	Write Count	Write Duration	Write Throughput
59	localhost/127.0.0.1:3306	16:30.478 [Jan 17, ...	16:30.4...	Socket	13	15075 s	23,395 bytes	7	8117 s	5,758 bytes
61	localhost/127.0.0.1:3306	16:35.148 [Jan 17, ...	16:35.1...	Socket	13	15154 s	25,175 bytes	7	8159 s	286 bytes
63	localhost/127.0.0.1:3306	16:37.338 [Jan 17, ...	16:37.3...	Socket	13	15184 s	23,395 bytes	7	8176 s	5,451 bytes
64	localhost/127.0.0.1:3306	16:42.058 [Jan 17, ...	16:42.0...	Socket	13	15267 s	25,175 bytes	7	8221 s	273 bytes
66	localhost/127.0.0.1:3306	16:46.708 [Jan 17, ...	16:46.7...	Socket	13	15334 s	23,395 bytes	7	8257 s	8,586 bytes
70	localhost/127.0.0.1:3306	17:08.678 [Jan 17, ...	17:08.6...	Socket	13	15652 s	25,175 bytes	7	8428 s	273 bytes
72	localhost/127.0.0.1:3306	17:12.068 [Jan 17, ...	17:12.0...	Socket	13	15702 s	23,395 bytes	7	8455 s	7,198 bytes
74	localhost/127.0.0.1:3306	17:17.948 [Jan 17, ...	17:17.9...	Socket	13	15810 s	25,175 bytes	7	8513 s	263 bytes
76	localhost/127.0.0.1:3306	17:21.678 [Jan 17, ...	17:21.6...	Socket	13	15865 s	23,395 bytes	7	8543 s	8,354 bytes
78	localhost/127.0.0.1:3306	17:26.528 [Jan 17, ...	17:26.5...	Socket	13	15949 s	25,175 bytes	7	8588 s	268 bytes
80	localhost/127.0.0.1:3306	17:29.938 [Jan 17, ...	17:29.9...	Socket	13	16000 s	23,395 bytes	7	8615 s	7,327 bytes
82	localhost/127.0.0.1:3306	17:34.678 [Jan 17, ...	17:34.6...	Socket	13	16081 s	25,175 bytes	7	8659 s	286 bytes
84	localhost/127.0.0.1:3306	17:38.068 [Jan 17, ...	17:38.0...	Socket	13	16130 s	23,395 bytes	7	8685 s	7,194 bytes
3	www.espnscrum.com/105.225.1.147:80	14:37.948 [Jan 17, ...	14:40.3...	Socket	39	39899 s	62,105 bytes	2	2040 s	477 bytes
5	www.espnscrum.com/105.225.1.147:80	14:40.518 [Jan 17, ...	14:46.3...	Socket	10	10247 s	11,748 bytes	1	1023 s	261 bytes
7	www.espnscrum.com/105.225.1.147:80	14:47.178 [Jan 17, ...	14:48.5...	Socket	39	40308 s	60,764 bytes	2	2063 s	429 bytes
9	www.espnscrum.com/105.225.1.147:80	14:48.768 [Jan 17, ...	14:55.0...	Socket	41	42661 s	72,810 bytes	3	3113 s	666 bytes
12	www.espnscrum.com/105.225.1.147:80	14:55.358 [Jan 17, ...	15:00.7...	Socket	6	6253 s	12,250 bytes	1	1041 s	237 bytes
14	www.espnscrum.com/105.225.1.147:80	15:00.958 [Jan 17, ...	15:02.6...	Socket	38	39929 s	60,377 bytes	2	2097 s	429 bytes
16	www.espnscrum.com/105.225.1.147:80	15:02.828 [Jan 17, ...	15:08.0...	Socket	8	8412 s	12,147 bytes	1	1051 s	237 bytes
35	www.espnscrum.com/105.225.1.147:80	15:39.818 [Jan 17, ...	15:45.0...	Socket	11	12073 s	12,372 bytes	1	1097 s	237 bytes
37	www.espnscrum.com/105.225.1.147:80	15:45.168 [Jan 17, ...	15:50.2...	Socket	34	37707 s	60,918 bytes	3	3317 s	645 bytes
50	www.espnscrum.com/105.225.1.147:80	16:11.758 [Jan 17, ...	16:17.3...	Socket	9	10224 s	12,359 bytes	1	1135 s	237 bytes
39	www.espnscrum.com/105.225.1.147:80	15:50.608 [Jan 17, ...	15:56.0...	Socket	9	9991 s	12,247 bytes	1	1109 s	237 bytes
41	www.espnscrum.com/105.225.1.147:80	15:56.248 [Jan 17, ...	15:57.4...	Socket	44	49190 s	60,832 bytes	2	2232 s	429 bytes
43	www.espnscrum.com/105.225.1.147:80	15:57.738 [Jan 17, ...	16:02.9...	Socket	8	8949 s	12,343 bytes	1	1118 s	237 bytes
45	www.espnscrum.com/105.225.1.147:80	16:03.088 [Jan 17, ...	16:04.6...	Socket	41	46198 s	60,710 bytes	2	2249 s	429 bytes
47	www.espnscrum.com/105.225.1.147:80	16:04.828 [Jan 17, ...	16:11.4...	Socket	50	56705 s	73,081 bytes	3	3394 s	666 bytes
52	www.espnscrum.com/105.225.1.147:80	16:17.448 [Jan 17, ...	16:19.0...	Socket	41	46929 s	60,817 bytes	2	2284 s	429 bytes
54	www.espnscrum.com/105.225.1.147:80	16:19.398 [Jan 17, ...	16:24.8...	Socket	5	5726 s	12,294 bytes	1	1145 s	237 bytes
56	www.espnscrum.com/105.225.1.147:80	16:24.978 [Jan 17, ...	16:28.0...	Socket	36	41645 s	60,607 bytes	2	2306 s	429 bytes
58	www.espnscrum.com/105.225.1.147:80	16:28.268 [Jan 17, ...	16:33.4...	Socket	9	10417 s	12,275 bytes	1	1157 s	237 bytes
60	www.espnscrum.com/105.225.1.147:80	16:33.688 [Jan 17, ...	16:35.1...	Socket	38	44283 s	60,341 bytes	2	2327 s	429 bytes
62	www.espnscrum.com/105.225.1.147:80	16:35.258 [Jan 17, ...	16:42.0...	Socket	59	69221 s	90,594 bytes	3	3510 s	640 bytes
73	www.espnscrum.com/105.225.1.147:80	17:15.478 [Jan 17, ...	17:17.9...	Socket	46	55920 s	70,917 bytes	2	2424 s	383 bytes
75	www.espnscrum.com/105.225.1.147:80	17:18.118 [Jan 17, ...	17:23.6...	Socket	13	15821 s	14,053 bytes	1	1216 s	214 bytes
77	www.espnscrum.com/105.225.1.147:80	17:24.868 [Jan 17, ...	17:26.4...	Socket	45	55197 s	70,012 bytes	2	2449 s	393 bytes
Total:					1,595	1791688 s	2,638 kB	367	411409 s	158 kB

Appendix C

Perfomance

C.1 Crawling Perfomance

C.1.1 CPU and Memory

C.1.2 Response Times

C.1.3 Socket Connections

C.1.4 Database connections

Figure C.1: Sockets Tab

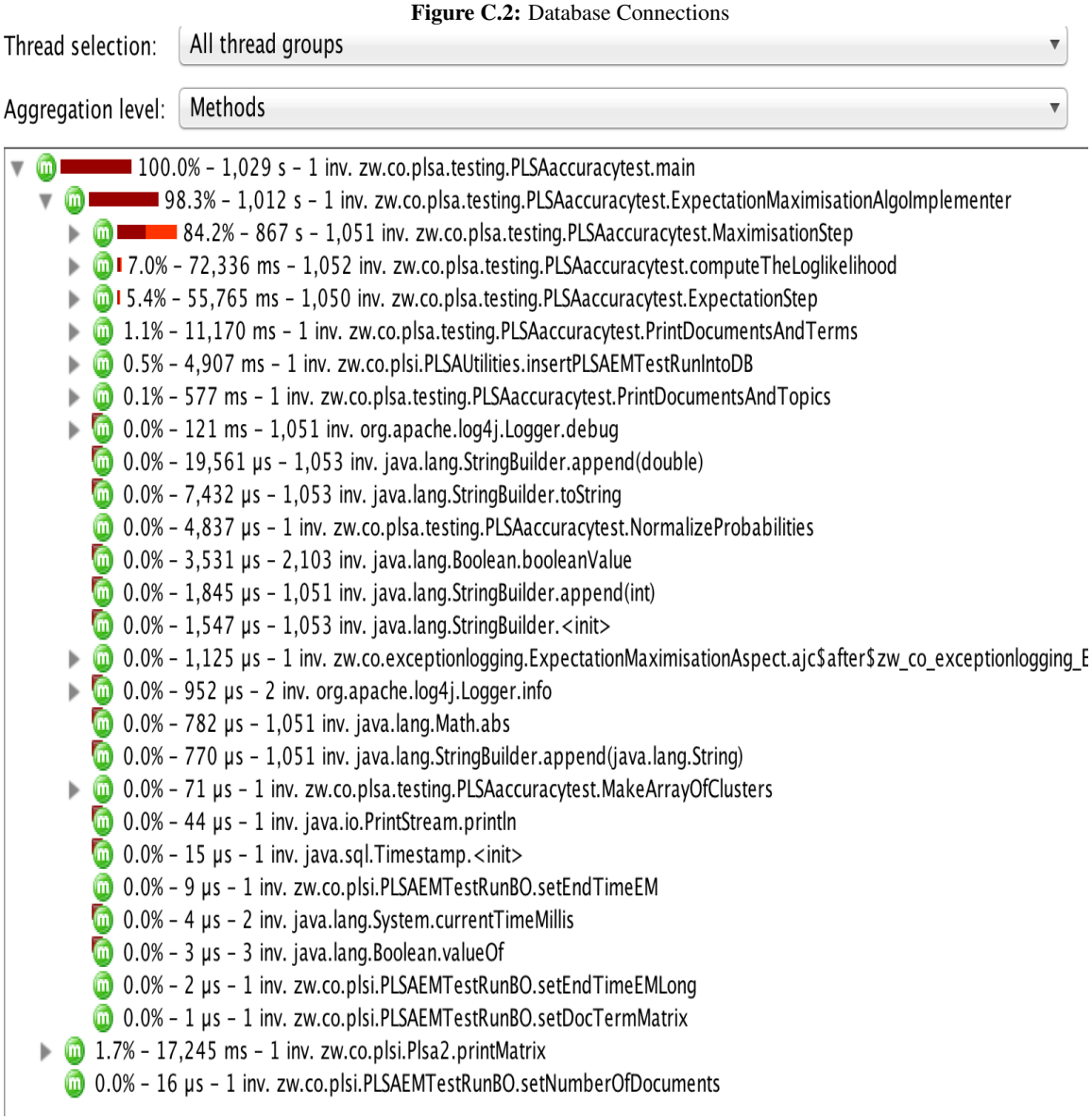
Show events: All types Filter by All text fields

Start Time	Event Type	Duration	Connection ID	Description	Thread
0:04.167 [Jan 17, 2...	Statement execution	1319 µs	23	SHOW VARIABLES	main [m
0:04.171 [Jan 17, 2...	Statement execution	677 µs	23	SHOW COLLATION	main [m
0:04.173 [Jan 17, 2...	Statement execution	535 µs	23	SELECT * FROM test.crawledWebSites where strUrl='...	main [m
0:04.174 [Jan 17, 2...	Connection closed	0 µs	23	jdbc:mysql://localhost:3306/test	main [m
0:08.694 [Jan 17, 2...	Statement execution	991 µs	24	SHOW VARIABLES	main [m
0:08.696 [Jan 17, 2...	Statement execution	661 µs	24	SHOW COLLATION	main [m
0:08.699 [Jan 17, 2...	Statement execution	620 µs	24	SELECT * FROM test.crawledWebSites where strUrl='...	main [m
0:08.699 [Jan 17, 2...	Connection closed	0 µs	24	jdbc:mysql://localhost:3306/test	main [m
0:13.488 [Jan 17, 2...	Statement execution	939 µs	25	SHOW VARIABLES	main [m
0:13.490 [Jan 17, 2...	Statement execution	576 µs	25	SHOW COLLATION	main [m
0:13.492 [Jan 17, 2...	Statement execution	529 µs	25	SELECT * FROM test.crawledWebSites where strUrl='...	main [m
0:13.493 [Jan 17, 2...	Connection closed	0 µs	25	jdbc:mysql://localhost:3306/test	main [m
0:18.068 [Jan 17, 2...	Statement execution	989 µs	26	SHOW VARIABLES	main [m
0:18.071 [Jan 17, 2...	Statement execution	692 µs	26	SHOW COLLATION	main [m
0:18.074 [Jan 17, 2...	Statement execution	804 µs	26	SELECT * FROM test.crawledWebSites where strUrl='...	main [m
0:18.075 [Jan 17, 2...	Connection closed	0 µs	26	jdbc:mysql://localhost:3306/test	main [m
0:22.432 [Jan 17, 2...	Statement execution	1082 µs	27	SHOW VARIABLES	main [m
0:22.435 [Jan 17, 2...	Statement execution	691 µs	27	SHOW COLLATION	main [m
0:22.437 [Jan 17, 2...	Statement execution	543 µs	27	SELECT * FROM test.crawledWebSites where strUrl='...	main [m
0:22.438 [Jan 17, 2...	Connection closed	0 µs	27	jdbc:mysql://localhost:3306/test	main [m

C.2 PLSA Perfomance

C.2.1 Database connections

Figure C.2: Database Connections



C.2.2 Log tracing using Log4J and AspectJ

Table C.7: Gather PLSA Data using AspectJ and Log4J

Gather PLSA Data using AspectJ and Log4J		
Join	**Join**	**Execution**
before(),after()	ExpectationMaximisationAlgoImplementer()	execution(* zw.co.WebCrawling.PLSAWebCrawler. ExpectationMaximisationAlgoImplementer(..)).
before(),after()	expectationStep()	execution(* zw.co.plsa.testing.PLSAaccuracytest.ExpectationStep(..)).
before(),after()	maximisationStep()	execution(* zw.co.plsa.testing.PLSAaccuracytest.MaximisationStep(..)).
before(),after()	computeTheLoglikelihood()	execution(* zw.co.WebCrawling.PLSAWebCrawler. computeTheLoglikelihood(..)).
before(),after()	initProbabilities()	execution(* zw.co.WebCrawling. PLSAWebCrawler.initProbabilities(..)).
before(),after()	generatePLSAMatrixTable()	execution(* zw.co.plsi.Plsa2.generatePLSAMatrixTable(..)).

Table C.8: Perfomance of PLSA

15:20:32,130 WARN zw.co.exceptionlogging.AspectJAspect - computeTheLoglikelihood has ended: 130 milliseconds total Log Likelihood Computation Time=70129
15:20:32,131 WARN zw.co.exceptionlogging.AspectJAspect - ExpectationStep has started:
15:20:32,234 WARN zw.co.exceptionlogging.AspectJAspect - ExpectationStep has ended: 103 millisecondstotal Expectation Time=52097E Step Number= 435
15:20:32,235 WARN zw.co.exceptionlogging.AspectJAspect - MaximisationStep has started:
15:20:33,825 WARN zw.co.exceptionlogging.AspectJAspect - MaximisationStep has ended: 1590 millisecondstotal Maximization Time=1049149
15:20:33,826 WARN zw.co.exceptionlogging.AspectJAspect - computeTheLoglikelihood has started:
15:20:33,963 WARN zw.co.exceptionlogging.AspectJAspect - computeTheLoglikelihood has ended: 137 milliseconds total Log Likelihood Computation Time=70266
15:20:33,964 WARN zw.co.exceptionlogging.AspectJAspect - ExpectationStep has started:
15:20:34,061 WARN zw.co.exceptionlogging.AspectJAspect - ExpectationStep has ended: 96 millisecondstotal Expectation Time=52193E Step Number= 436
15:20:34,062 WARN zw.co.exceptionlogging.AspectJAspect - MaximisationStep has started:
15:20:35,604 WARN zw.co.exceptionlogging.AspectJAspect - MaximisationStep has ended: 1541 millisecondstotal Maximization Time=1050690
15:20:35,605 WARN zw.co.exceptionlogging.AspectJAspect - computeTheLoglikelihood has started:
15:20:35,732 WARN zw.co.exceptionlogging.AspectJAspect - computeTheLoglikelihood has ended: 127 milliseconds total Log Likelihood Computation Time=70393
15:20:35,741 WARN zw.co.exceptionlogging.AspectJAspect - computeTheLoglikelihood has started:
15:20:35,870 WARN zw.co.exceptionlogging.AspectJAspect - computeTheLoglikelihood has ended: 128 milliseconds total Log Likelihood Computation Time=70521
15:22:02,320 WARN zw.co.exceptionlogging.AspectJAspect - ExpectationMaximisation has ended: 1261866 millisecondstotalDocument Generation Time=45428totalInitializationTime=62

C.2.3

C.2.4 CPU Consumption during execution

Figure C.3: CPU and Heap Size(450 Documents)

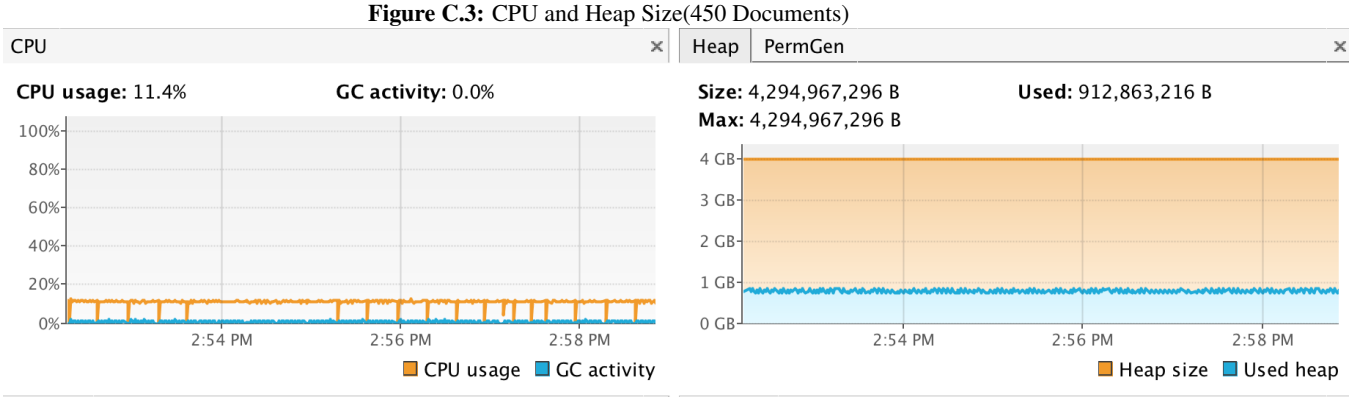
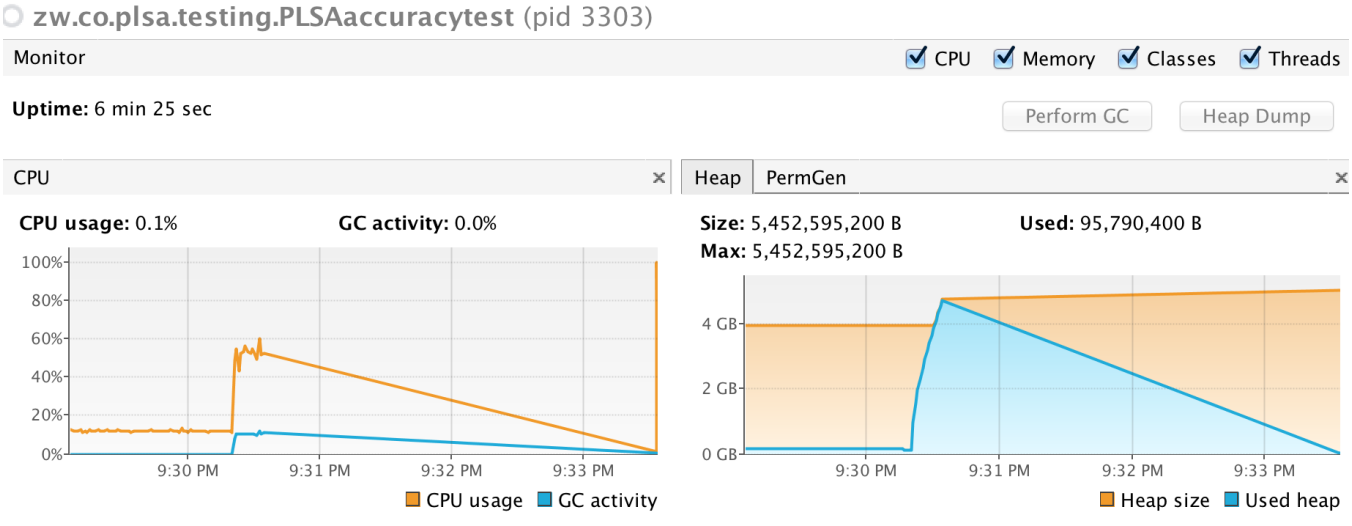
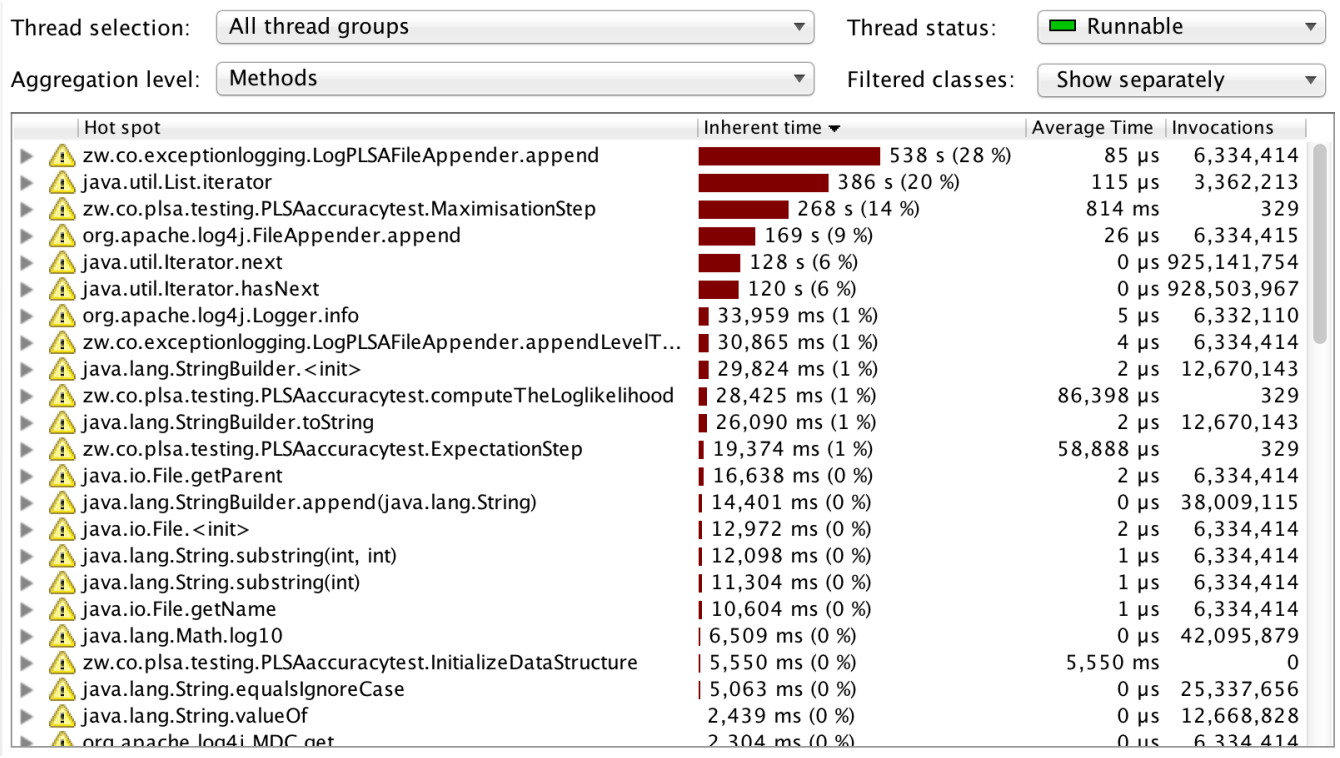


Figure C.4: CPU and Heap Size(450 Documents)



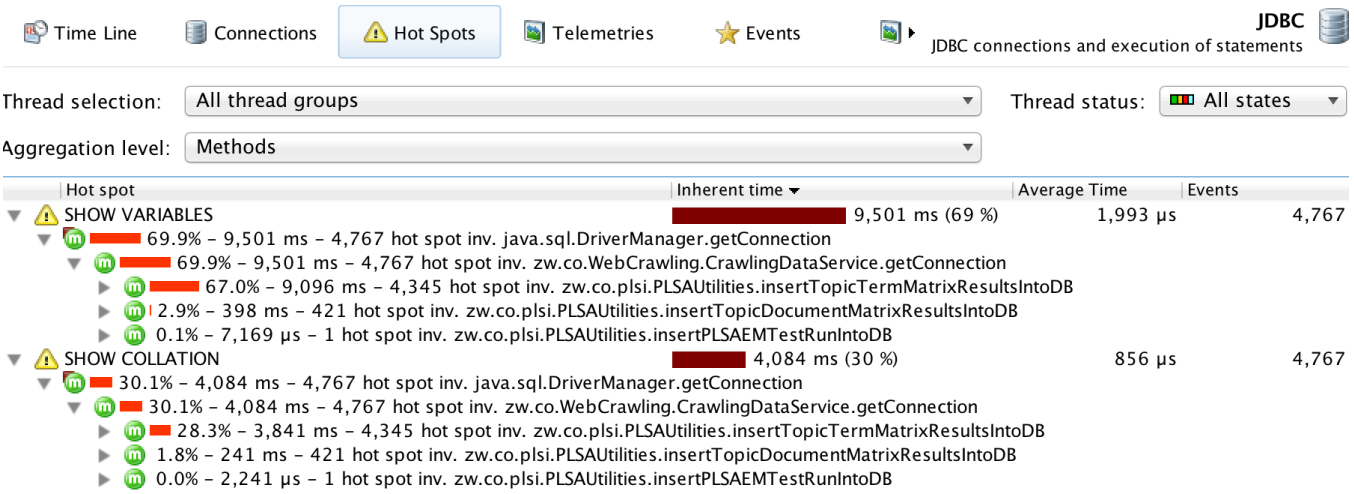
C.2.5 Java threads during processing

Figure C.5: Sockets Tab



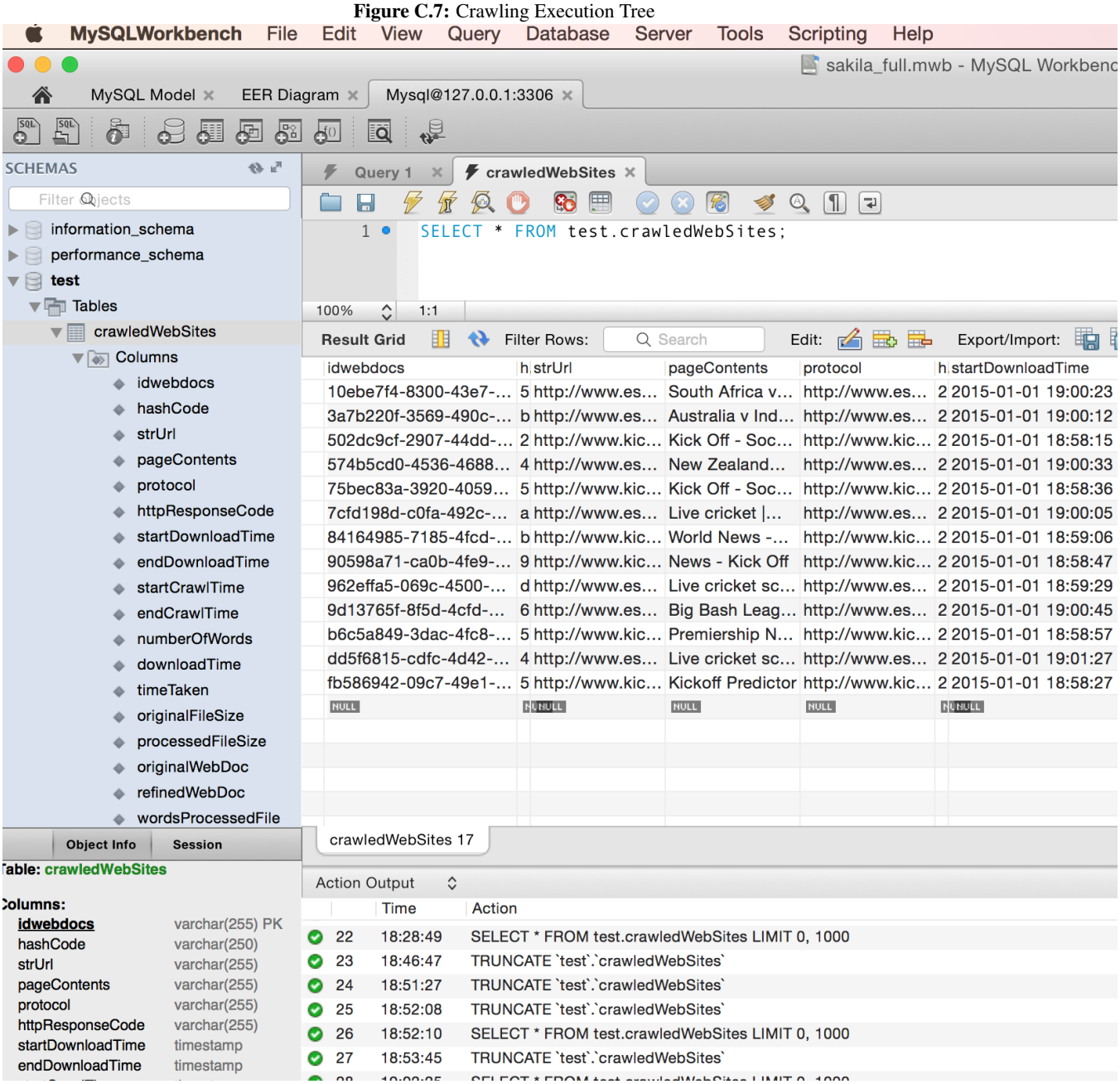
C.2.6 Inserting data into MySQL Database

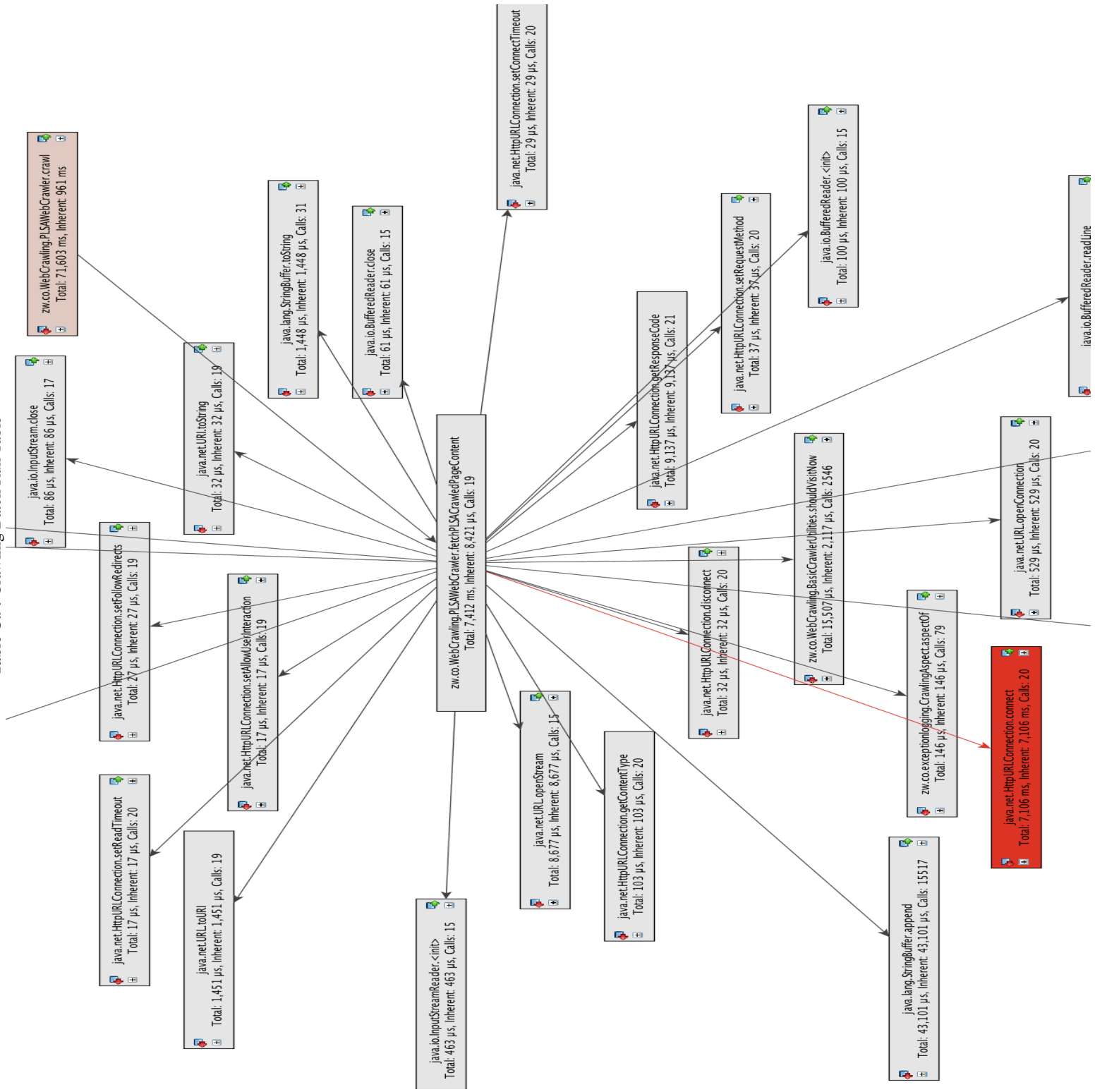
Figure C.6: Sockets Tab



C.3 Crawling Results

Figure C.7: Crawling Execution Tree




zw.co.WebCrawling.PLSAWebCrawler.crawl
Total: 71,603 ms, Inherent: 961 ms
java.io.InputStream.close
Total: 86 µs, Inherent: 86 µs, Calls: 17
java.net.URL.toString
Total: 32 µs, Inherent: 32 µs, Calls: 19
java.lang.StringBuffer.toString
Total: 1,448 µs, Inherent: 1,448 µs, Calls: 31
java.io.BufferedReader.close
Total: 61 µs, Inherent: 61 µs, Calls: 15
java.net.HttpURLConnection.setConnectTimeout
Total: 29 µs, Inherent: 29 µs, Calls: 20
java.net.HttpURLConnection.setRequestMethod
Total: 37 µs, Inherent: 37 µs, Calls: 20
java.io.BufferedReader.<init>
Total: 100 µs, Inherent: 100 µs, Calls: 15
java.io.BufferedReader.readLine
java.net.HttpURLConnection.setFollowRedirects
Total: 27 µs, Inherent: 27 µs, Calls: 19
java.net.HttpURLConnection.setAllowUserInteraction
Total: 17 µs, Inherent: 17 µs, Calls: 19
zw.co.WebCrawling.PLSAWebCrawler.fetchPLSACrawledPageContent
Total: 7,412 ms, Inherent: 8,421 µs, Calls: 19
java.net.HttpURLConnection.getResponseCode
Total: 9,137 µs, Inherent: 9,137 µs, Calls: 21
java.net.HttpURLConnection.disconnect
Total: 32 µs, Inherent: 32 µs, Calls: 20
zw.co.WebCrawling.BasicCrawlerUtilities.shouldVisitNow
Total: 15,507 µs, Inherent: 2,117 µs, Calls: 2546
java.net.URL.openConnection
Total: 529 µs, Inherent: 529 µs, Calls: 20
java.net.HttpURLConnection.setReadTimeout
Total: 17 µs, Inherent: 17 µs, Calls: 20
java.net.URL.toURI
Total: 1,451 µs, Inherent: 1,451 µs, Calls: 19
java.io.InputStreamReader.<init>
Total: 463 µs, Inherent: 463 µs, Calls: 15
java.net.URL.openStream
Total: 8,677 µs, Inherent: 8,677 µs, Calls: 15
java.net.HttpURLConnection.getContentType
Total: 103 µs, Inherent: 103 µs, Calls: 20
zw.co.exceptionlogging.CrawlingAspect.aspectOf
Total: 146 µs, Inherent: 146 µs, Calls: 79
java.net.HttpURLConnection.connect
Total: 7,106 ms, Inherent: 7,106 ms, Calls: 20
java.lang.StringBuffer.append
Total: 43,101 µs, Inherent: 43,101 µs, Calls: 15517

C.3.1 Perfomance Results

Figure C.8: CPU Memory Tab

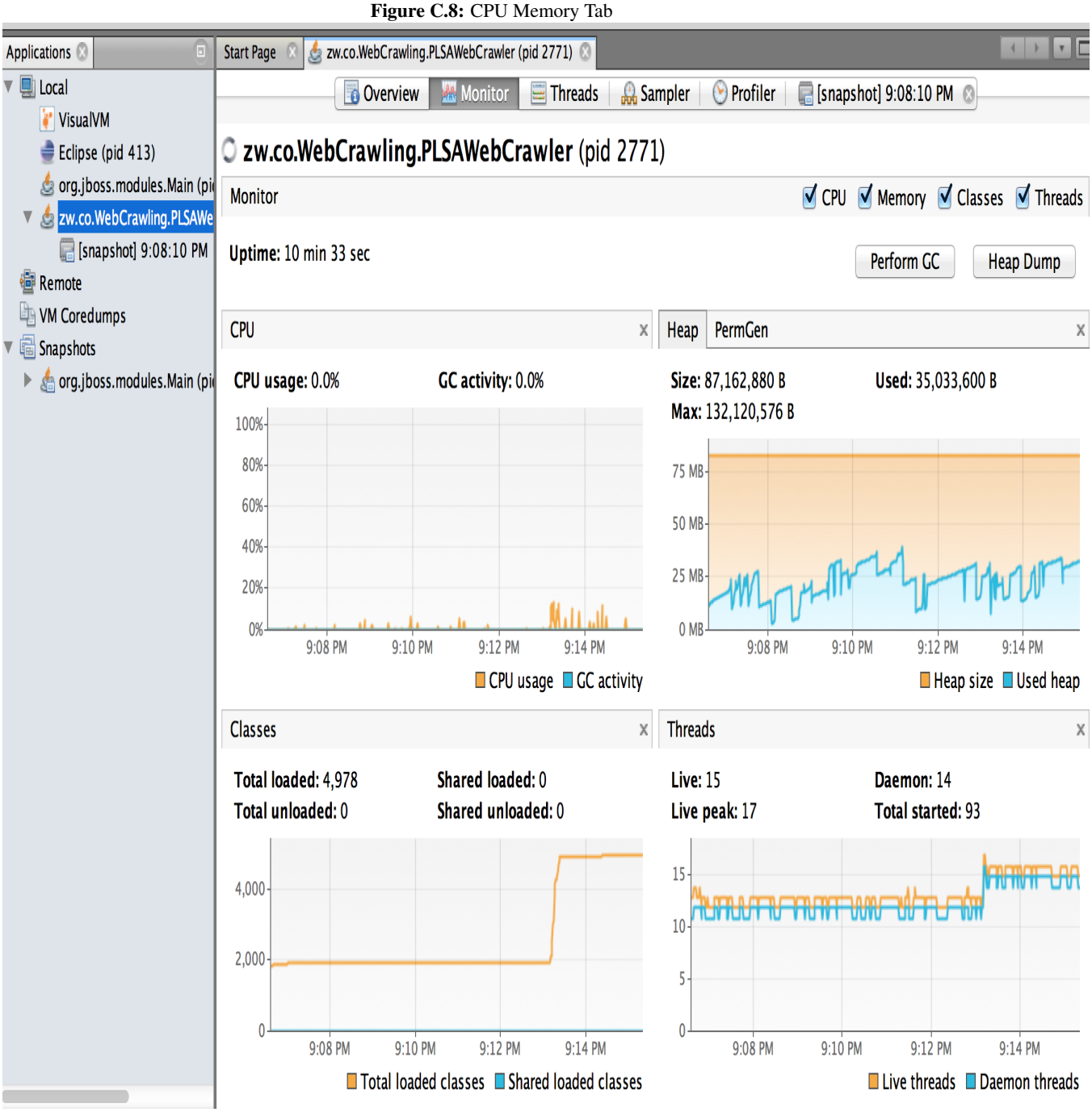


Figure C.9: CPU Memory Tab

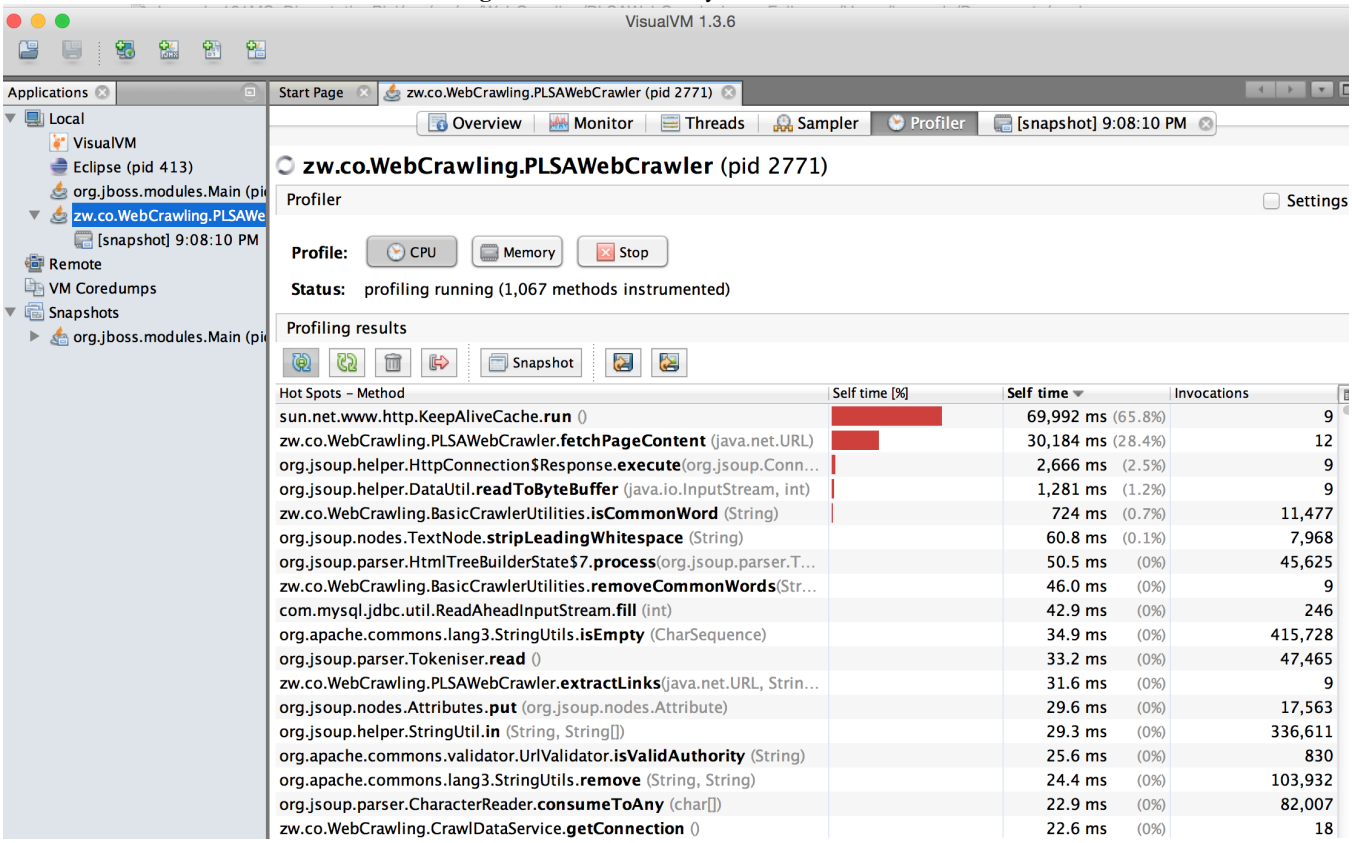
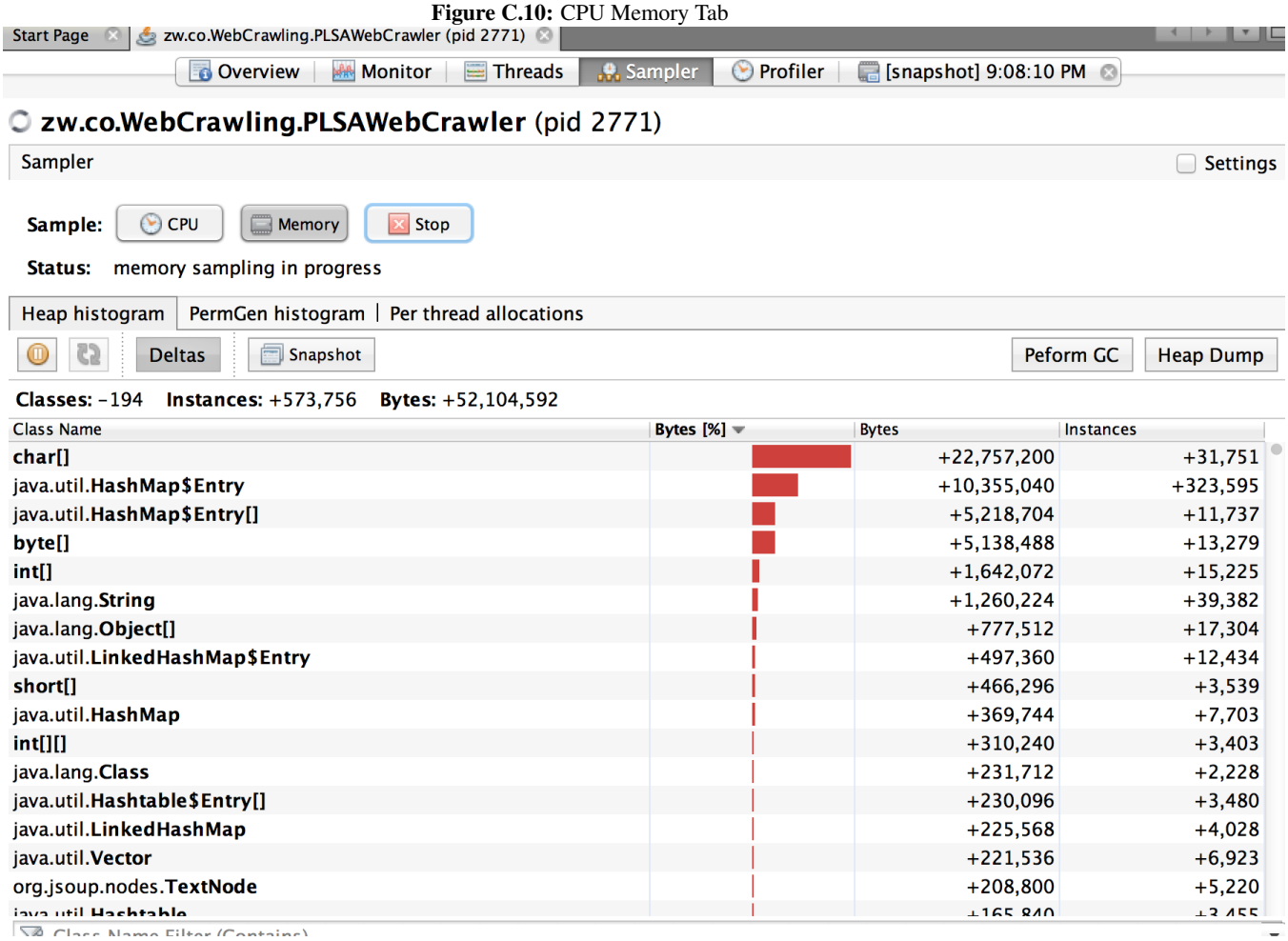


Figure C.10: CPU Memory Tab



Appendix D

Acronyms

PLSA	Probability Latent Semantic Analysis
LSA	LAtent Semantic Analysis
SQL	Standard Query Language
JDBC	java database connection
AJAX	asynchronous java and XML
JEE	Java Enterprise Edition
CPU	Central Processing Unit
EM	Expectation Maximisation
MLE	Maximum Likelihood Estimate
SQL	Standard Query Language

Index