# CENSORSHIP AGENT: IDENTIFYING AND DETERMINING OFFENSIVE WORDS

By

HARLINA BT MAT ALI

Dissertation submitted to the
Business Information System Programme
Universiti Teknologi PETRONAS
in partial fulfillment of
the requirements for the
Bachelor of Technology (Hons)
(Business Information System)

DECEMBER 2005

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

t
105.875
.S7
H254
2005

internet -- Security measures
IT/IS -- Thesis

## CERTIFICATION OF APPROVAL

**Censorship Agent: Identifying and Determining Offensive Words**

by

HARLINA BT MAT ALI

Dissertation submitted to the
Business Information System Programme
Universiti Teknologi PETRONAS
in partial fulfillment of
the requirements for the
Bachelor of Technology (Hons)
(Business Information System)

Approved:

[illegible]

MISS AMY FOONG OI MEAN
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
December 2005

## CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

[illegible]

HARLINA BT MAT ALI

## ABSTRACT

The advancement in computing enables anyone to become information producer, resulting in rapidly growing information in the Internet. One concern that arises from this phenomenon is the easy access to offensive, vulgar and/or obscene page by anyone with access to the Internet. The solution for this concern is filtering software. Current existing filtering software required human intervention in determining the harmfulness of page content. The fear of this trend brings out the desire to protect a community, especially children from the harmful content available. This paper represents a prototype of application that performs the task of identifying and determining the harmfulness content of a document without human intervention. The prototype is designed to extract the content of the document, stem the words into its root, and compare each word to the list of harmful words.

## ACKNOWLEDGEMENT

First and foremost, the author would like to thank God, for His blessing that helps her to complete this Final Year Project. Not forgotten to the IT/IS FYP Coordinator and Information System Department of Universiti Teknologi Petronas for the chance and guideline given to complete the project.

Special thanks go to the author's supervisor, Miss Amy Foong Oi Mean for suggesting an interesting topic and choosing the author as her FYP supervisee. Miss Amy was always there to listen and give advice to the author. She helps the author to complete the writing of this dissertation and the challenging research that lies behind it.

In addition, the author would like to extend her thank you to her friends that help her during the period of completion. They helped a lot in giving ideas and opinions regarding to this project.

The appreciation also goes to the evaluator that helps in conducting the testing for the prototype. Finally, the author would like to thank all lecturers and her colleagues who are involved either directly or indirectly in completing the FYP. All ideas, advices and compassions to the author are greatly appreciated.

## TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

# CHAPTER 1
# INTRODUCTION

## 1.1 Background Study

In this century, we are living in the information age where information plays an important role in our daily lives. This change can also be called as the 'Information Revolution'. The society's economics and cultural forces are now governed by the production and dissemination of vast amount of information. "Nearly everything we do, think and feel relies on the information we receive" [1]. The advances achieved in computing enable information to be digitized and obtain their unique characteristics of bits, namely convergence, compression, increased speed of dissemination and intelligence in the network [2] that have come together and form a network called Internet and the World Wide Web. These characteristics also enable anyone to become information producer, allowing them to produce and distribute information that can be available to the world wide audience. Internet has also become the most essential tools for most organizations [17].

The Science Magazine has estimated that the size of the web is roughly about 320 million pages, with the web growing by several hundred percent per year [2]. According to the Scientific American Article produced in March 1997, "the average life span of a web page is 75 days". The massive size of information available on the Internet produces unwanted byproduct which is the overabundance of information that can also be called "information overload". Searching information in the Internet becomes a challenge as it often produces an overwhelming numbers of links, in which many points to entirely irrelevant sites. As the web grows and become the attention of the public in 1994, the "Americans have been obsessed with the scourge of easily accessed on-line pornography, violence and hate speech" [2]. Most companies realize that the Internet can be a double-edged sword, as it is one of essential tools to succeed in today's world and yet it possesses potential distraction for employees [17].

The fear of this trend brings out the desire to protect a community, especially children from the harmful content available. As Internet becomes a part of daily life, the need for technology solutions to help in managing Web access in education and enterprise becomes more acute [17]. As law on harmful Internet content has been passed, the software industry developed technological solutions, namely the content blocking filtering software. The software perfectly enforces their rules, blocking prohibited sites from being viewed by Internet user. Four most popular software filters currently available are Net Nanny, Solid Oak Software's CYBERsitter, The Learning Company's Cyber Patrol and SpyGlass Inc.'s SurfWatch.

## 1.2 Problem Statement

Due to the desire to protect a community from the harmful information on the Internet, software filter is invented. The existing filter software (Net Nanny, CYBERsitter, Cyber Patrol, and SurfWatch) employed the same procedure. They employed the use of artificial intelligence web spider to flag potential inappropriate content to be reviewed, categorized and added to the blocked list by the company employees. The list of URL's added to the company BLOCK LIST will be block and not accessible to their respective clients. As each software call for human intervention, a huge amount of resource (mainly labor and money) is needed in order to keep up with the increasing number of web page in the Internet.

In addition, some filtering product yanks offending words from web pages without providing a clue to the reader that the text has been altered [18]. The altered text that results from filtering might change the meaning and intent of a sentence dramatically. For instance, because "homosexual" was listed as offensive in the offensive list, the sentence "The Catholic church is opposed to all homosexual marriages" appears to the user as "The Catholic church is opposed to all marriages."

The effort to create a perfectly working filter is not yet achieved. There are vast amount of reports in the Internet that pertains to the failure of filters to block the most repulsive content while most of them successfully block the non-sexual, non-violent content. There is no software available that is able to detect the potential harmful document without human intervention.

## 1.3 Objectives and Scope of Study

The main purpose of this study is to produce a prototype of software filter to identify obscene wordings in a document in order to determine if document is harmful. The prototype will identify the existence of obscene word in the document or page without the intervention of human. This lead to the second objective, which is to find a way to reduce resources needed in order to filter unwanted information. In general, this study focuses on identifying harmful content of document to find possible solution that could be implemented to improve the identification process and reduce human intervention.

# CHAPTER 2
# LITERATURE REVIEW

## 2.1 Introduction to Software Agents

Software Agent is the fastest growing area of Information Technology. Some define agent as a persistence software that entirely dedicated to a specific purpose, while another define it as a computer program that stimulate human relationship by doing something that another person can do for you [3]. According to M.R.Patra, & H.Mohanty, there are three basic characteristic expected from an agent, which are responsiveness, pro-activeness and social ability [6]. However, agent could not recognize all possible events in its environment unless it is built in the knowledge base. This knowledge base is the information repository for an agent as it determines how the agent perceives and response toward its environment.

Table 2.1 Basic Characteristics of an Agent

| | |
|---|---|
| Responsiveness | Ability to perceive and respond appropriately to changes in its environment that is triggered by the arrival of messages. |
| Pro-activeness | Ability to take action on behalf of the user by taking a goal-directives behavior |
| Social ability | Ability to take part in activity that involve communication with other agents such as cooperation, collaboration, negotiation, etc. |

According to Concise Oxford Dictionary (Tenth Edition), censor is an official who examines material that is to be published and suppresses parts considered offensive or a threat to security. The urge to censor is one of the oldest and most basic urges that grow from the desire to protect the community from harmful ideas. Harmful ideas are ideas that differ from the conventional community norm. Contrary to popular beliefs, the first work of censorship focuses on censoring literature that is deemed dangerous to religious or political authority rather than censoring those immoral and repulsive writing because the main force behind the censorship idea is the Church [2]. When the Church monopoly in pre-printing press comes to an end, they formalized a list of banned book, which include the immoral and obscene works.

## 2.2 Related Work

### 2.2.1 Censorware

There are four most popular filters which are Net Nanny, Solid Oak Software's CYBERsitter, The Learning Company's Cyber Patrol and SpyGlass Inc.'s SurfWatch. All developed around 1995, which correspond to "The Great Cyber Porn Panic".

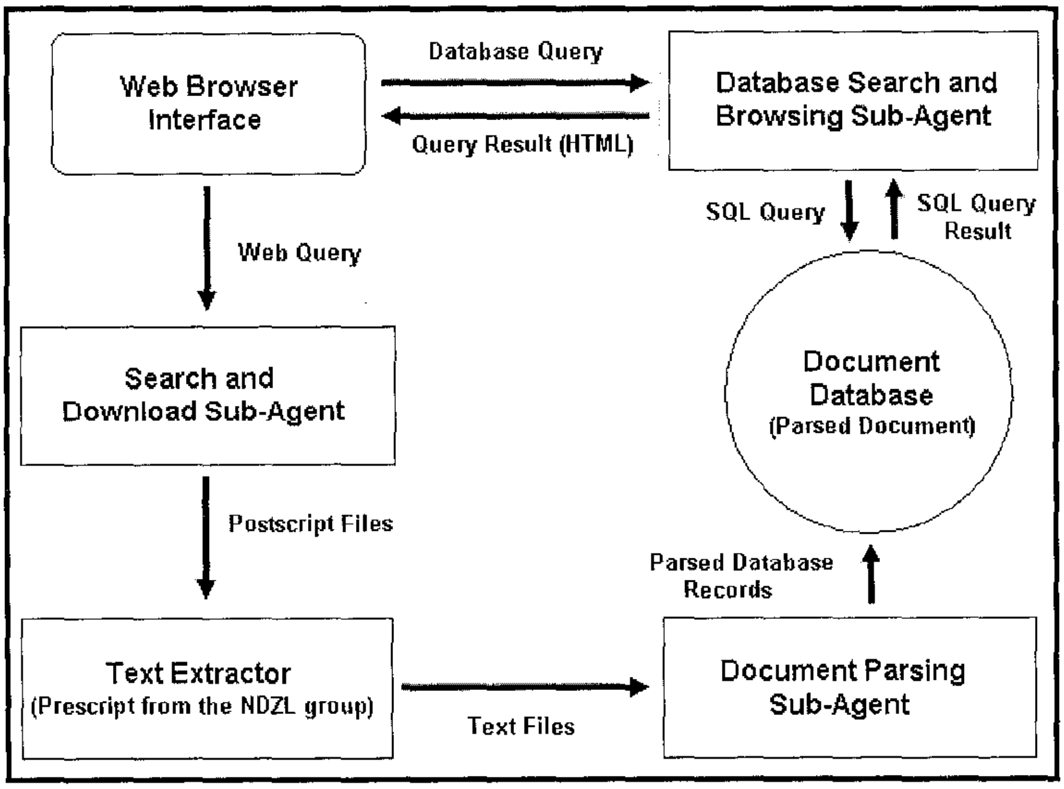
- *Net Nanny claims that it will protect the children and free speech on the internet while ensuring on-line safety for their users.*
- *CYBERsitter claims to provide the safest way to explore cyberspace and guaranteed to block over 95 percent of all objectionable content on the Internet.*
- *Cyber Patrol offers the best way to manage children's computer use and safety on the Internet.*
- *Surf Watch states that they empower people with the information, technology and tools needed to harness positive potential of Internet.*

All filters mentioned employed the same mechanism which is categorizing, listing, and word filtering and access or distribution control [2]. These companies follow the same procedure of employing a web spider to flag potential harmful content that will then be reviewed, categorized and added to the company's block sites lists by the company employees.

An artificial intelligence web spider will visit sites and create a report, 25 characters before and 25 characters after each occurrence of keywords used in the search. Once this is done, researcher will review this report to decide the harmfulness of the site. When necessary, researcher need to visit and review before being added to the block list.

### 2.2.2 Autonomous Web Agent

CiteSeer is an autonomous agent for automatic retrieval and identification of interesting publications. It is a Web based information agent that assists the user in the process of performing a scientific literature research [7]. With a given keyword, it uses the Web search engines and heuristics to locate and download papers. CiteSeer was developed to reduce the potential of duplication on previously performed work, and to keep up with current research especially in the quickly advancing fields. CiteSeer consists of three main components, namely a subagent to automatically locate and acquire research publication, a document parser with database creator and a database browser interface which support searching by keyword and browsing by citation links [7]. Figure 2.1 shows the architecture of CiteSeer.



Web Browser Interface
Database Query
Query Result (HTML)
Database Search and Browsing Sub-Agent
SQL Query
SQL Query Result
Web Query
Document Database (Parsed Document)
Search and Download Sub-Agent
Postscript Files
Parsed Database Records
Text Extractor (Prescript from the NDZL group)
Text Files
Document Parsing Sub-Agent

Figure 2.1 CiteSeer Agent Architecture

During the document parsing, document is downloaded in order to enable the process of extracting semantic features to be performed. Features of desired document is extracted and placed into SQL database. First step in document parsing is the extraction of raw text from the Postscript file [7]. Heuristics are used to identify the header, abstract, introduction, citation and word frequency in the document. Word frequencies are recorded for all stop words. The recorded words are then stemmed using Porter's algorithm. The word frequency of each citation is also recorded in order to apply the stop word removal and stemming.

One common semantic feature used by CiteSeer to gauge document topic similarity is the word vector. The Term Frequency Inverse Document Frequency (TFIDF) scheme is implemented to measure value for each word stem, in which the vector of all the word stem values represented a document.

### 2.2.3 Adaptive Information Retrieval

According to Tomas Olsson [4] an adaptive information retrieval system is based on the query and relevance feedback from the user. System retrieved document based on queries from user and wait for feedback from user to indicate if retrieved document match the wanted document. In the process of relevance feedback, user will identify relevant document from the list of retrieved document to enable the system to create new query based on the sample document [5]. Based on this concept, the new query created based on relevant documents will return document that will also be similar to the desired document.

### 2.2.4 Adaptive Information Filtering

There are two approaches for filtering information, namely cognitive filtering and social filtering. *Cognitive filtering* analyses the content of a document, compares it to the user model and wait for relevance feedback from user. This feedback is then used to change the user model. In this approach, document will be recommended to a new user if the content of the document is similar to previously encountered documents [4].

According to Tomas Olsson, "Social filtering is based solely on what different users are recommending". This concept relies on the opinion of other users that have the same preferences [4]. For example, when one likes document on Ferrari (sport car), they will probably like other document that is liked by other user who loves sport car.

Somehow, the concepts of information filtering and information retrieval are often difficult to differentiate, but both addressed the same issue of getting wanted information in which information retrieval is when one tries to find all relevant document from a collection while information filtering is when one tries to remove all irrelevant document from a collection [4].

### 2.2.5 Stemming Algorithm

According to the Search Engine Dictionary.com, "stemming is the use of linguistic analysis to get the root forms of the search terms to documents in its database" [10]. For example, once user enters the query, search engine reduces the words to its root and return document containing the root word. Stemming is used to remove prefix and suffixes from a word in order to obtain the root word thus help to improve retrieval effectiveness while reducing the size of indexing files [11]. The taxonomy of stemming algorithm is as shown in Figure 2.2.

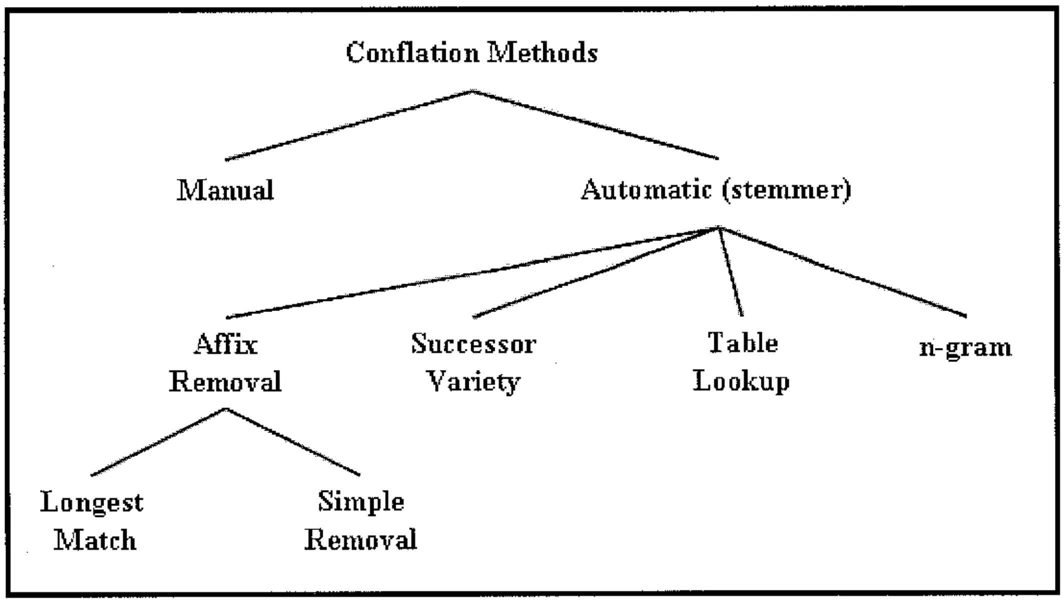


Figure 2.2 Stemming Algorithm Taxonomy

There are a number of different stemming algorithms currently, which are Paice/Husk Stemming Algorithm, Porter Stemming Algorithm, Lovins Stemming Algorithm, Dawson Stemming Algorithm and Krovetz Stemming Algorithm.

Porter algorithm is a conflation Stemmer developed by Martin Porter at University of Cambridge in 1980. This algorithm is a process for removing the commoner morphological and inflexion endings from words in English [12]. Porter Algorithm strip suffix based on the idea that the suffixes in the English language are mostly made of combination of smaller and simpler suffixes [13]. Figure 2.3 below shows the steps in Porter Stemmer Algorithm.

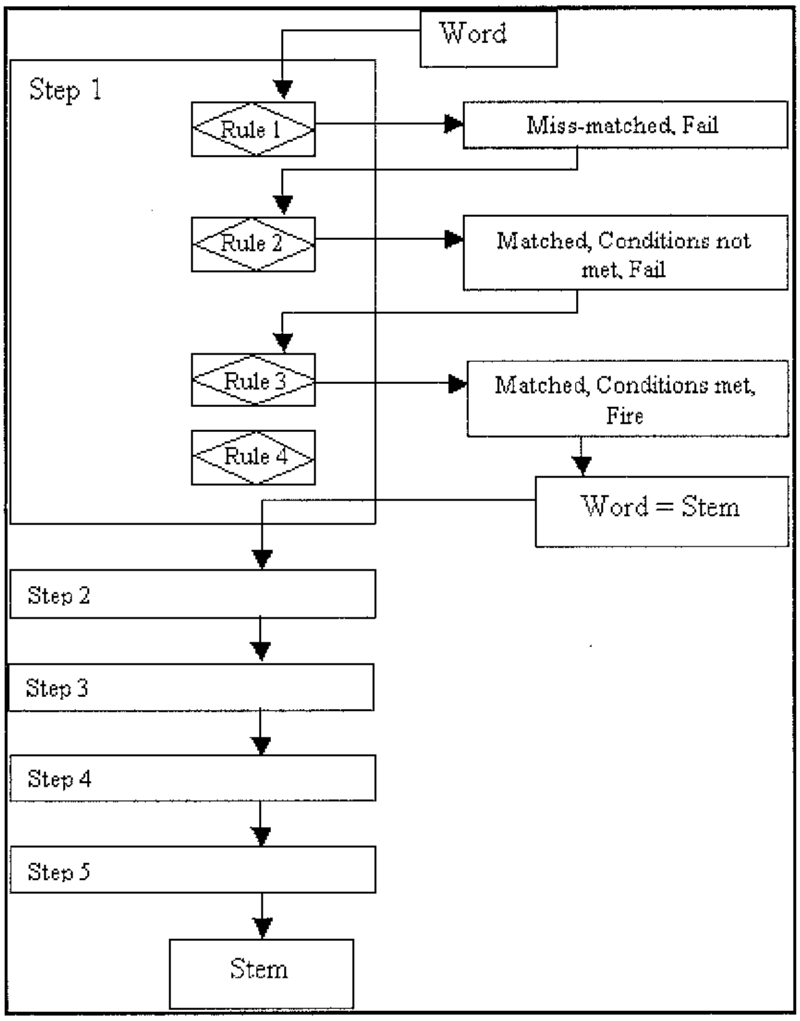


Figure 2.3 Porter Stemmer Algorithm

Porter Stemmer Algorithm has five steps. In each step, when a suffix rule matches to the word, conditions attached to the rules are tested on the resulting stem that has it suffix removed. Once a rule passes its condition and is accepted, the rule is applied to word and the word's suffix is removed. Otherwise, if any of the rules in step 1 is not accepted, word is tested with the other rules in the subsequent step [13].

This Porter Algorithm has been widely used, quoted, and adapted over the past 20 years [12].

Table 2.2 Porter Stemmer Algorithm Step 1

| STEPS | RULES | INPUT | OUTPUT |
|---|---|---|---|
| STEP 1(1) | SSES -> SS | Caresses | Caress |
| | IES -> I | Ponies | Poni |
| | | Ties | Ti |
| | SS ->SS | Caress | Caress |
| | S -> | Cats | cat |
| STEP 1(2) | EED -> EE | Feed | Feed |
| | ED -> | Agreed | Agree |
| | | Plastered | Plaster |
| | ING -> | Motoring | Motor |
| | | Sing | Sing |
| STEP 1(3) | AT -> ATE | Conflated | Conflate |
| | BL -> BLE | Troubled | Trouble |
| | IZ -> IZE | Sized | Size |
| STEP 1(4) | Y -> I | Happy | Happi |
| | | Sky | Sky |

Step 1 deal with plurals and past participles. In a set of rules written above in Table 2.2, only one is obeyed. The rule that is obeyed will be the one with the longest matching suffixes for a given word [14]. When stem ends with the letter "s", the stem is analyze against step 1(1) rule, when stem ends with "EED", "ED" or "ING", stem will be analyze against step 1(2) rule. When step 1 (1) and (2) is successful, stem is analyze against step 1(3) rule in which the letter "E" is put back on –AT, -BL and –IZ, so that the suffixes –ATE, -BLE, -IZE can be recognize later [14]. In step 1(4), when the stem contains a vowel and ends with letter "Y", "Y" will be replace with "I".

This subsequent step is much more straightforward. The following rules will be applied to get the root words.

Table 2.3 Porter Stemmer Algorithm Steps 2, 3 and 4

| STEPS | RULES | INPUT | OUTPUT |
|---|---|---|---|
| STEP 2 | ATIONAL -> ATE | Relational | Relate |
| | TIONAL -> TION | Conditional | Condition |
| | ENCI -> ENCE | Valenci | Valence |
| | ANCI -> ANCE | Hesitanci | Hesitance |
| | IZER -> IZE | Digitizer | Digitize |
| | ABLI -> ABLE | Conformabli | Conformable |
| | ALLI -> AL | Radicalli | Radical |
| | ENTLI -> ENT | Differentli | Different |
| | ELI -> E | Vileli | Vile |
| | OUSLI -> OUS | Analogousli | Analogous |
| | IZATION -> IZE | Vietnamization | Vietnamize |
| | ATION -> ATE | Predication | predicate |
| | ATOR -> ATE | Operator | operate |
| | ALISM -> AL | Feudalism | Feudal |
| | IVENESS -> IVE | Decisiveness | Decisive |
| | FULNESS -> FUL | Hopefulness | Hopeful |
| | OUSNESS -> OUS | Callousness | Callous |
| | ALITI -> AL | Formaliti | Formal |
| | IVITI -> IVE | Sensitiviti | Sensitive |
| | BILITI -> BLE | Sensibility | Sensible |
| STEP 3 | ICATE -> IC | Triplicate | Triplic |
| | ATIVE -> | Formative | Form |
| | ALIZE -> AL | Formalize | Formal |

| | | | |
|---|---|---|---|
| | ICITI -> IC | Electriciti | Electric |
| | ICAL -> IC | Electrical | Electric |
| | FUL -> | Hopeful | Hope |
| | NESS -> | Goodness | Good |
| STEP 4 | AL -> | Revival | Reviv |
| | ANCE -> | Allowance | Allow |
| | ENCE -> | Inference | Infer |
| | ER -> | Airliner | Airlin |
| | IC -> | Gyroscopic | Gyroscop |
| | ABLE -> | Adjustable | Adjust |
| | IBLE -> | Defensible | Defens |
| | ANT -> | Irritant | Irrit |
| | EMENT -> | Replacement | Replac |
| | MENT -> | Adjustment | Adjust |
| | ENT -> | Dependent | Depend |
| | ION -> | Adoption | Adopt |
| | OU -> | Homologou | Homolog |
| | ISM -> | Communism | Commun |
| | ATE -> | Activate | activ |
| | ITI -> | Angulariti | Angular |
| | OUS -> | Homologous | Homolog |
| | IVE -> | Effective | Effect |
| | IZE -> | Bowdlerize | bowdler |

Step 2 and 3 are applied when the measure of stem analyzed is more than 0, while step 4 is applied when the measure of stem being analyze is more than 1. The suffixes are all removed when step 4 is completed. Step 5 is a step to tidy up each stem.

Table 2.4 Porter Stemmer Algorithm Step 5

| STEPS | RULES | INPUT | OUTPUT |
|---|---|---|---|
| STEP 5 | E -> | Probate | Probat |
| | | Rate | Rate |
| | | Cease | Ceas |

Although the algorithm is widely used, there are still several drawbacks. One drawback that affects the retrieval performance of an Information Retrieval system is the over-stemming errors. In over-stemming, stemmers operating on natural words unavoidably make mistakes as natural languages are not completely regular constructs [15]. For instance, words which are distinct may be wrongly conflated to give similar stems.

On the other hand, there is the under-stemming error in which words that ought to be merged together may remain distinct after stemming takes place [15]. This error however does not affect the retrieval performance of an Information Retrieval. An example of this error is the word characterizes and characteristic. After stemming occur on both words, characterizes is stemmed into character and characteristic is stemmed into characterist.

# CHAPTER 3
# METHODOLOGY

Methodology used as the guide and framework throughout the development of this project is the System Development Life Cycle (SDLC), which consists of Planning, Analysis, Design, and Testing.

## 3.1 Planning Phase

The first phase for the project development is the planning phase in which the project's problem statement, objectives, scope of study, tools to be used and schedule are established. To accomplish this, research has been conducted to identify problem faced in the real world regarding to the Internet. Intensive discussion with the supervisor is carried out to identify scope feasible for further research and improvement. The activities conducted during the planning phase can be referred to Appendix I.

## 3.2 Analysis / Research Phase

In the analysis phase, intensive research is conducted to gather information related to the scope of study. Vast amount of journals, articles and report is used as reference to learn on previous work conducted by other researchers. Information regarding existing filtering software is gathered in order to understand the process and problems in identifying and filtering. Information gathered is then analyze to find solution that could improve the performance of current technology.

## 3.3 Design Phase

Once the analysis phase ends, the design of selected solution from the research is establish. The workflow of the project is determined to guide the implementation later in the implementation phase. Figure 3.1 shows the proposed architecture of the censorship agent that consists of three main components which are document preprocessing, list processing and document processing.

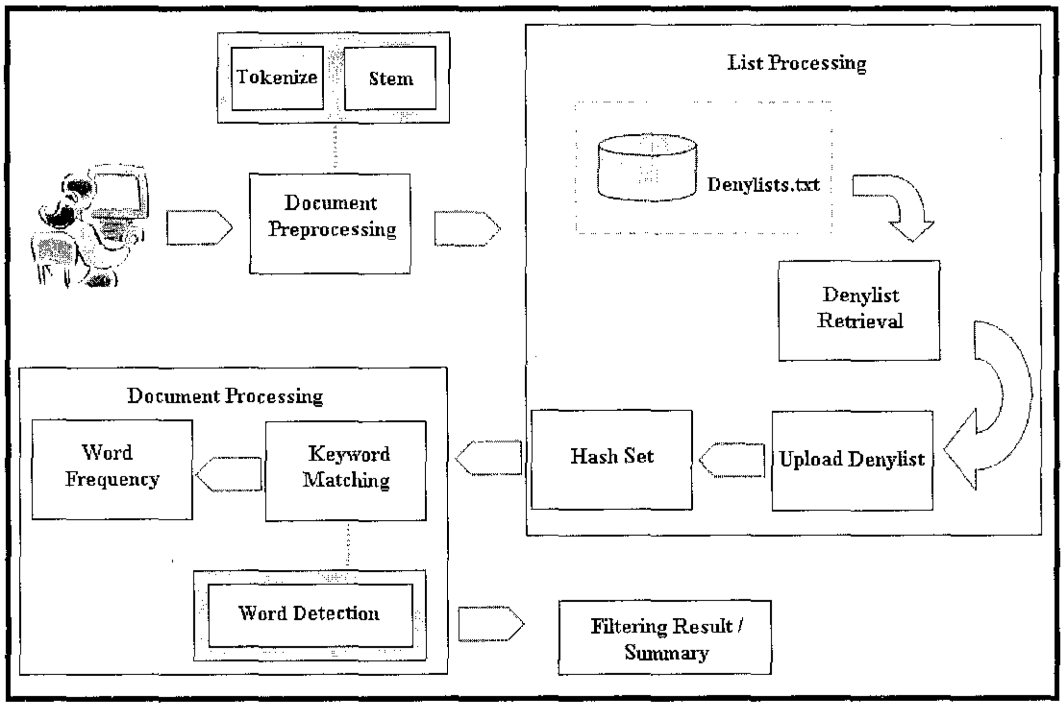


Figure 3.1 Architecture of Censorship Agent

Given a specified directory, *DocFilter* will browse through the specified directory for input. All text documents in the folder are read by *DocFilter*, offensive word in the text document is identified and the status is determined.

### 3.3.1 Document Preprocessing

The first step done by the application is document preprocessing, where the *DocFilter* read the text document contained in the specified directory and extract each string using a method called tokenizing. String tokenizing involved the used of existing function in Java to extract the substring from the string into individual word. This word is called token. This process is required in order to enable the system to analyze each word contains in the document word by word rather that painstakingly analyzing character by character [8].

Since Java is case sensitive, token is then change to lowercase to reduce the size of the denylist database. By doing this, the size of denylist database can be reduce as there is no need to specify each offensive words in different ways in which it can be written. For example, there is no need to replicate the same word stored in the database such as "Stem", "stem" or "STEM".

After each character is change to lowercase, the process of stemming each word take place. Stemming is the use of linguistic to get the root of a word. Existing Porter stemmer algorithm is used in *DocFilter*. During stemming, each token will be stem to its root by removing the Prefixes and Suffixes.

The first step in stemming is removing the Prefixes and Suffixes. According Robert Harris [9], there are two types of prefixes, the root and number prefixes. General Root prefixes are word like "mega", "mis" and "multi", while the number prefixes are word like "kilo", "giga" and "micro". According to the Concise Oxford Dictionary Tenth Edition, suffix is a morpheme added at the end if a word to form derivative. Example of word containing suffixes are "homeless", "brotherhood" and "hopeful". When words containing prefix and suffix are stemmed, the result will be as the following. Table 3.1 below shows some example of prefixes and suffixes in a word.

Table 3.1 Stemming result

| Mismatch | Match |
|---|---|
| Multimillionaire | Millionaire |
| Microeconomic | Economic |
| Homeless | Home |
| Brotherhood | Brother |
| Hopeful | Hope |

Once the prefixes and suffixes are completely removed, the resulting stem is used for further processing to detect the offensive word occurrence and determine the text document status.

### 3.3.2 List Processing

There is a list of offensive words stored in a text file named "denylist.txt", which acts as the database for DocFilter. Once all the strings have been tokenized, the list processing phase takes place. The offensive words contained in the denylist database are retrieved and uploaded into a hash set in order to enable DocFilter to perform keyword matching.

Hash set is a collection that contains unique elements, stored in a hash table and is actually a HashMap instance. It is typically made up of an array where items are accessed by integer index [21]. Features of a hash set are more efficient since access time in an array is bounded by a constant regardless of the number of items in the container. This class offers a constant time performance for the basic operation such as add, remove, contain and size, assuming the hash function distributes the elements properly among the buckets [22].

### 3.3.3 Document processing

In document processing, two functions take place, which are word detection and word frequency calculation.

In word detection, *DocFilter* used the keyword matching method in order to detect and filter the occurrence of offensive word in the text document. Keyword matching is a flexible filtering technique. Each token in the text document is analyzed by comparing the token with the list of offensive word in the hash table to find all the unacceptable words listed in the denylist database.

If the token analyzed matches with the word specified in the offensive list, the word is recorded and the number of offensive word occurrence is calculated. In addition, each offensive word that is detected in the document is replaced with a tag "<censored>" to inform user that the document is being filtered by *DocFilter*.

Once offensive word is detected, DocFilter calculate the frequency of offensive word occurrence. To calculate the percentage of total word occurrence of the document, the number of sentence in the document is recorded. Using the number of offensive word occurrence calculated in word detection, the percentage of offensive word occurrence for the whole text document is calculated with the following formula [16], as done by the Textalyser page:

$$\text{Percentage of offensive word occurrence} = \frac{\text{Number of offensive word occurrence}}{\text{Total number of word in the document}}$$

Textalyser is an online text analysis tool, which detailed the statistics of your text. The site also calculated each text input frequency occurrence.

Based on this percentage of offensive word occurrence calculation, the status of text document can be determined according to the level of blocking chosen by the user. Figure 3.2 shows the snap shot of DocFilter four blocking levels provided.

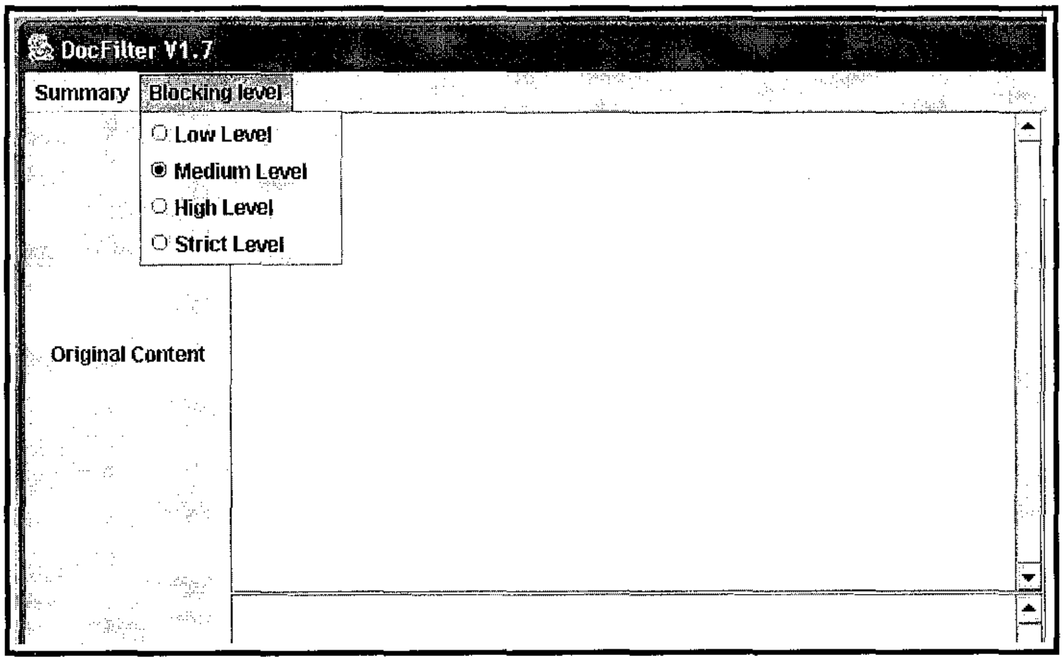


Figure 3.2 DocFilter four blocking levels

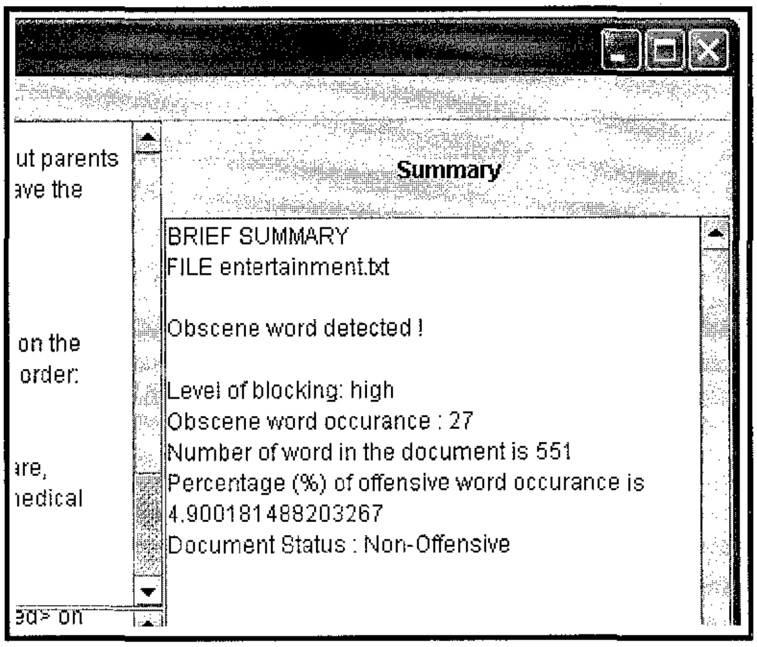
DocFilter provide four different blocking levels, namely “Low Level”, “Medium Level”, “High Level” and “Strict Level”. The percentage of offensiveness of each blocking level is as the following:

- Low blocking level – Document is offensive when the percentage of occurrence is more than 15%
- Medium blocking level – Document is offensive when the percentage of occurrence is more than 10%

- High blocking level – Document is offensive when the percentage of occurrence is more than 5%
- Strict blocking level – Document is offensive when the text document contains any offensive word.

Using this technique, the filter can be customized to allow some flexibility to support different level of access of different types of users.

Once offensive words are detected and the status of text document is determined, *DocFilter* provide user with the summary of the filtering done. Figure 3.3 shows the summary section of *DocFilter*.



```
BRIEF SUMMARY
FILE entertainment.txt

Obscene word detected !

Level of blocking: high
Obscene word occurance : 27
Number of word in the document is 551
Percentage (%) of offensive word occurance is
4.900181488203267
Document Status : Non-Offensive
```

Figure 3.3 *DocFilter* summary sections

## 3.4 Development Phase

Based on the architecture of DocFilter designed in the design phase, DocFilter is developed using SUN ONE Studio 4 (Forte for Java).

## 3.5 Testing and Delivery Phase

Series of testing will be conducted to test the functionality of the produce product, namely stub testing, and program testing. This testing will be conducted through the entire of the development process. For testing purpose, a prototype interface is created in order to see the output of each method coded.

### 3.5.1 Stub Testing

Stub testing is done throughout the application development life cycle. In stub testing, testing is done on each individual event of the application. There are a number of individual event that needs to be tested.

Table 3.2 Stub Testing

| | |
|---|---|
| Scanning function | Scan specified folder and upload document content in the folder |
| Stemming function | Stem each token to get the root word. |
| Keyword matching function | Match each token with the deny list in hash set to detect offensive word |
| Word occurrence calculation | Calculate the percentage of offensive word occurrence |

### 3.5.2 Program Testing

Once stub testing has been done on all four functions, all function is tested as an integrated unit. This testing is done after all functions listed in Figure 1.6 are integrated as one application. The application is tested as a whole to ensure that it works well as one.

## 3.6 Development Tools

For the development and construction of the project, a few sets of hardware and software are used.

### 3.6.1 Managerial / Documentation Tools

- *Microsoft Word*

  Used in the preparation of log books, project's documentation and final dissertation of the study.

- *Microsoft PowerPoint*

  Used in the presentation presented to the internal and external examiners.

### 3.6.2 Development and Construction Tools

- *SUN ONE Studio 4 (Forte for Java)*

  The main development for the project and construction is using the Forte SUN ONE Studio 4.

- *Microsoft Notepad version 5.1*

  Stored the list of stop words and wordings that is categorized as offensive.

- *Aglets version 2.0.2*

  Platform to run the agent.

- *Development and Construction Hardware*

  Hardware that is used for the development and construction of the project is a personal computer. Specification of the personal computer is as the following:

  System:

  Microsoft Windows XP
  Professional
  Version 2002
  Service Pack 2

  Computer:

  Intel®
  Celeron® CPU 1.70GHz
  1.70 GHz, 480 MB of RAM

# CHAPTER 4
# RESULT AND DISCUSSION

There are three modules developed for the application, namely document preprocessing, list processing and document processing. Each module was developed with an objective in mind. The first module is document preprocessing which involves tokenizing word into individual tokens and stemming each token into the roots. List processing module involves loading the list of offensive word from the text file (denylists.txt) into a hash set, while document processing module involve keyword matching and word occurrence calculation. Document preprocessing and list processing work in the background which is not transparent to the user. In the meantime, the document processing module will produce output that is readable to the user. Each of this module functions as expected.

## 4.1 Findings

Most of the offensive words tested using this application is correctly filtered, but there are still some inaccuracies in the process of filtering. There is a need to specify a number of word repeatedly in order for the application to correctly filter all offensive word as the application still faces problem in stemming each word. For instance both "erotica" and "erotically" need to be added into the database in order to enable the application to filter correctly. Table 4.1 shows the outcome of filtering result.

Table 4.1 Filtering outcome

| **Denylists.txt** | **Input** | **Result** |
|---|---|---|
| anal | anal | <censored> |
| | anally | anally |
| anarchy | anarchy | <censored> |
| ass | ass | <censored> |
| asshole | asshole | <censored> |
| beastiality | beastiality | <censored> |

| | | |
|---|---|---|
| bestiality | bestiality | <censored> |
| blowjob | blowjob | <censored> |
| | blowjobs | <censored> |
| | blow job | blow job |
| bomb | bomb | <censored> |
| | bombs | <censored> |
| bondage | bondage | <censored> |
| | bondages | <censored> |
| boob | boob | <censored> |
| | boobs | <censored> |
| | booby | booby |
| buttfuck | buttfuck | <censored> |
| | buttfucking | <censored> |
| | buttfucker | buttfucker |
| clit | clit | <censored> |
| | clits | <censored> |
| cock | cock | <censored> |
| | cocks | <censored> |
| coitus | coitus | coitu |
| copulate | copulate | <censored> |
| | copulation | <censored> |
| | copulatory | copulatory |
| cunnilingus | cunnilingus | cunnilingu |
| cunt | cunt | <censored> |
| dick | dick | <censored> |
| | dicks | <censored> |
| dildo | dildo | <censored> |
| | dildos | <censored> |
| | dildoes | dildoe |
| drug | drug | <censored> |
| | drugs | <censored> |
| | drugged | <censored> |
| | drugging | <censored> |
| ejaculate | ejaculate | <censored> |
| | ejaculation | <censored> |
| | ejaculator | <censored> |
| | ejaculatory | ejaculatory |
| erection | erection | <censored> |
| | erect | erect |
| | erectable | erectable |
| erotic | erotic | <censored> |
| | erotics | <censored> |
| | erotically | erotically |
| erotica | erotica | <censored> |
| fuck | fuck | <censored> |

| | | |
|---|---|---|
| | fucks | <censored> |
| | fucker | <censored> |
| | fucking | <censored> |
| | fucked | <censored> |
| horny | horny | <censored> |
| | hornier | hornier |
| | horniest | horniest |
| | horniness | horni |
| masturbate | masturbate | <censored> |
| | masturbation | <censored> |
| | masturbator | <censored> |
| nude | nude | <censored> |
| | nudes | <censored> |
| | nudity | nudity |

As seen on Table 4.1, some of the offensive word is not filtered. In table 4.1, a total number of 65 words are tested using *DocFilter*. From 65 of the words tested, 16 words are not filtered by *DocFilter*. In order to determine the percentage of filtering accuracy done by DocFilter, the number of words that is not filtered is divided by total of words tested. Based on this formula, the accuracy of DocFilter in filtering a document can be derived. The result of testing done for DocFilter filtering accuracy is 75% as shown in Figure 4.1.

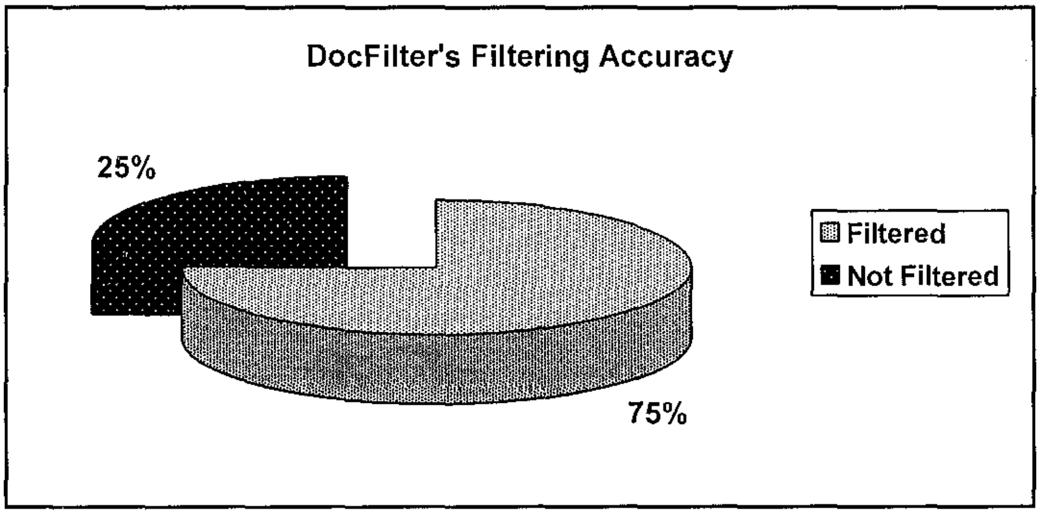


Figure 4.1 Filtering Accuracy

The inaccuracy in *DocFilter* arises due to the inaccuracy in the Porter Stemmer Algorithm. Some words are over-stemmed while some are under-stemmed. For instance the word coitus is over-stemmed to coitu. Due to over-stemmed and under-stemmed, some offensive words that occur in text document analyzed cannot be detected.

When a word is stemmed inaccurately, the application is unable to match the word analyzed with the list of offensive word in the hash set. Figure 4.2 shows the problem faced by DocFilter. There are three problems in the DocFilter's filtering which are over-stemming, under-stemming and others. Others in Figure 4.2 refer to words that are made up of multi-word or word that is not stemmed correctly.

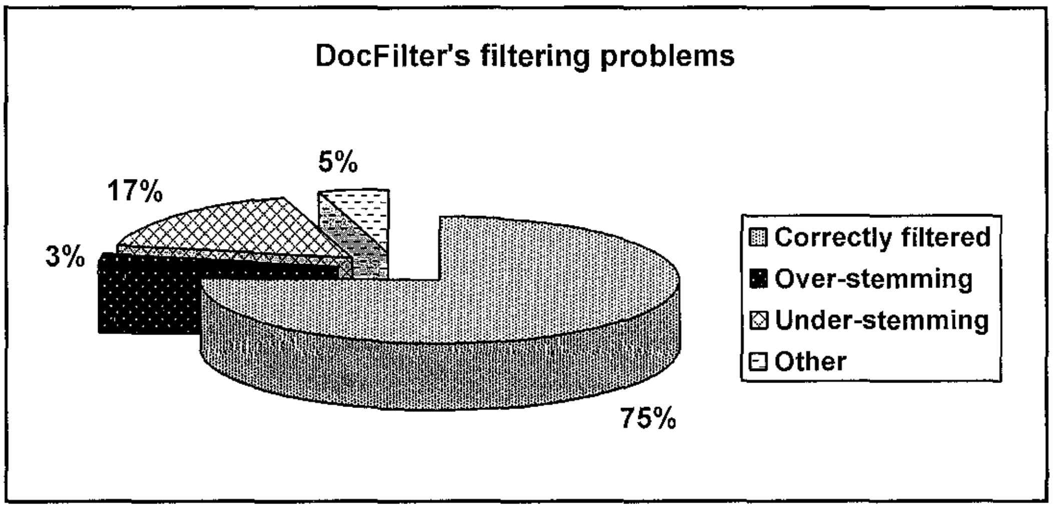


Figure 4.2 Filtering problems

From 65 words tested, a total of 11 words are under-stemmed that made up 17%, 2 words are over-stemmed that made up 3% and others are 5%.

## 4.2 User Evaluation

There are a total of six different types of document tested using DocFilter. These documents are business news, sport news, entertainment news, product description, song lyrics and personal blog.

A total of 25 users consist of 16 parents and 9 English linguists are involved in the evaluation process, each evaluator is given six different types of text documents. Step involved in this evaluation is as the following:

- First each user will read the original document.

- Next each user will highlight the occurrence of offensive word encounter.
- The same document is then processed using *DocFilter* to extract all offensive word in the document.
- Marks is given to each word that is highlighted both in the original document and extracted by *DocFilter*.
- For each report, average for the highlighted word by human evaluators and offensive word extracted by *DocFilter* is calculated.
- Then, total average for all documents is derived by comparing the scores of word highlighted by human evaluators against scores of *DocFilter*'s word occurrence.
- Finally, the text document status determined by DocFilter is compared against text document status determined by human evaluators.

The evaluation of offensive word of both offensive words highlighted by evaluators and offensive words detected by *DocFilter* is shown in Table 4.2.

Table 4.2 Offensive words evaluation

| **Document Type** | **Total number of words** | **Offensive word occurrence (based on denylist.txt)** | **Result from human evaluator (E)** | **Result from DocFilter (D)** | **Document status** |
|---|---|---|---|---|---|
| Song lyrics | 264 | 24 | 27.68 | 21 | Offensive |
| Sport news article | 795 | 0 | 0 | 0 | Not offensive |
| Business news article | 226 | 0 | 0 | 0 | Not offensive |
| Entertainment news article | 551 | 27 | 27.2 | 22 | Not offensive |
| Product description | 192 | 0 | 0 | 0 | Not offensive |
| Personal blog | 363 | 16 | 19.52 | 12 | Offensive |

From the evaluation conducted, the result can be summarized as the following:

1. There is no offensive word detected on 3 text document, sport news article, business news article and product description.
2. The average scores of offensive words detected by human evaluators differs

from the number of offensive words detected by *DocFilter* for the other 3 text document which are song lyrics, entertainment news articles and personal blog. Based on the list of offensive word stored in the denylist database, the actual occurrence in the song lyrics are 24, entertainment news articles contains 27 and personal blog contains 16 offensive words. The average scores by human evaluators are 26.76 for the song lyrics, 27.2 for the entertainment news and 19.52 for the personal blog. In the meantime, DocFilter manages to detect 21 offensive words for the song lyrics, 22 offensive words for entertainment news article and 12 offensive words for personal blog.

3. Based on the default blocking level (High Level), both song lyrics and personal blog is identified as offensive by DocFilter while the entertainment news article is not considers as offensive. The result produces is similar for both human evaluators and *DocFilter*.

Table 4.3 shows the scores comparison made between human evaluators and *DocFilter*. Using the following formula, the average score for DocFilter accuracy in determining the text document status is Result from *DocFilter* **(D)** divided by Result from human evaluator **(E)**.

Table 4.3 Scores comparison between human evaluators and *DocFilter*

| **Document type** | **Result from human evaluator / English expert (E)** | **Result from *DocFilter* (D)** | **(D / E) x 100%** |
|---|---|---|---|
| Song lyrics | 27.68 | 21 | 75.86 |
| Sport news article | 0 | 0 | 0 |
| Business news article | 0 | 0 | 0 |
| Entertainment news article | 27.2 | 22 | 80.88 |
| Product description | 0 | 0 | 0 |
| Personal blog | 19.52 | 12 | 61.47 |

Based on "High" blocking level, the performance of *DocFilter* in determining the status of text document is 75.86% for song lyrics, 80.88% for entertainment news article and 61.47% for the personal blog. Based on these evaluations, the total average scores for DocFilter's filtering accuracy is 86.36%.

After the document is processed, any offensive word encounter is then replaced with "<censored>". This tag is used so that the reader will be aware that the document has been filtered. The significance of this approach is that it will preserve the meaning and intent of a sentence. Refer to appendix III for further understanding.

# CHAPTER 5
# CONCLUSION AND RECOMMENDATION

From the study of Internet and its impact, one concern is discovered about Internet's content. As the amount of information is growing rapidly, it is difficult for filtering companies to keep up their task in identifying, filtering and blocking those harmful pages. This project aims to find possible method or solution that could identify offensive word in text document and determine the text document status without human intervention. This could lead to reduction of resource required to track harmful pages.

The proposed architecture of DocFilter has been carefully design to provide result efficiently and accurately. Overall results on DocFilter evaluation are satisfactory. Based on the user evaluation conducted, DocFilter have produced a fairly good result as it has succeeded in identifying offensive words in the text document up to 86.36% for 6 different text documents (song lyrics, sport news articles, business news article, entertainment news article, product description and personal blog).

DocFilter has also managed to determine the status of text document being filtered based on the level of blocking chosen by the user.

There are number of recommendation suggested for the application. Recommendation is as the following:

- Improvement to stemming algorithm

  The accuracy of filtering can be achieved if the stemming algorithm can be improved to stem each word into their root word accurately. Currently, DocFilter faces some inaccuracy in the filtering process due to the problems of under-stemmed and over-stemmed.

- Automate list of offensive word

An automate list of offensive word is a good approach to be apply to DocFilter. It will enable the user to add and delete word from the offensive word database as they see fit since there is a possibilities that new offensive words will appear.

- Recognition for medical document

Some medical documents and articles that does contain some of the offensive word listed in the deny list database such as the word "breast" although the document is not offensive. A function should be embedded in D*ocFilter* so that it can recognize these documents and thus, the document will not be filter in similar way.

# REFERENCES

[1] Web Skills and Evaluation. The Importance of information. Retrieved August, 18, 2005, from the World Wide Web: <http://edtech.tennessee.edu/~set8/intro2.html>

[2] Christopher D. Hunter (1999). *Filtering the Future?: Software Filters, Porn, PICS and the Internet Content Conundrum. Unpublished thesis.* University of Pennsylvania: Faculty of the Annenberg School.

[3] Nick Jennings., & Michael Wooldridge (1996, January). Software Agent. *IEE REVIEW*, 17-20.

[4] *Department of Computer and System Sciences.* (1998) Information Filtering with Collaborative Interface Agent. (Report).Stockholm, Sweden.

[5] Liren Chen., & Katia Sycara. (1997). *WebMate : A Personal Agent for Browsing and Searching.* 12

[6] M. R.Patra., & H.Mohanty (2001). *A Formal Framework to Build Software Agents. IEEE.* 119-126.

[7] Bollacker, K., Lawrence, S. & Giles L. (1998). CiteSeer: An autonomous Web Agent for Automatic Retrieval and Identification of Interesting Publications. In the *2nd International ACM Conference on Autonomous Agent.*

[8] *String Tokenizing and File Handling.* Retrieved September 3 2005 from the World Wide Web: <http://www.csc.liv.ac.uk/~frans/COMP101/AdditionalStuff/tokenizing1.html>

[9] Word Roots and Prefixes. Retrieved August 7 2005 from the World Wide Web: <http://www.virtualsalt.com/roots.htm>

[10] Search Engine Dictionary.com. Retrieved August October 11 2005 from the World Wide Web: <http://www.searchenginedictionary.com/terms-stemming.shtml>

[11] Stemming Algorithms. Retrieved on 11 August 2005 from the World Wide Web: <http://www.mis.nsysu.edu.tw/~syhwang/Courses/IR/StemmingAlgorithms.ppt>

[12] The Porter Stemming Algorithm. Retrieved August 15 from the World Wide Web: <http://www.tartarus.org/~martin/PorterStemmer>

[13] The Lancaster Stemming Algorithm. Retrieved August 15 from the World Wide Web: <http://www.comp.lancs.ac.uk/computing/research/stemming/general/>

[14] An algorithm for suffix stripping. Retrieved August 20 from the World Wide Web: <http://tartarus.org/~martin/PorterStemmer/def.txt>

[15] *Faculty of Computer Sciences.* Further Enhancement to the Porter's Stemming Algorithm. (Report). Beirut, Lebanon.

[16] Textalyser. Retrieved August 27 from the World Wide Web 6: <http://textalyser.net/index.php?lang=en#analysis>

[17] Packet Dynamics Ltd. studying Bloxx Filtering Technologies Version 2. (1999-2005). *Bloxx No Nonsense*, 1-6.

[18] Censorware: How well does Internet filtering software protect student. Retrieved September 2 from the World Wide Web: <http://www.electronic-school.com/0198f1.html>

[19] H.M & P.J Deital (2002). *Java: How to Program 4th Edition.* New Jersey: Prentice Hall.

[20] Y. Daniel Liang (2003). *Rapid Java Application Development using Sun ONE Studio 4.* New Jersey: Prentice Hall.

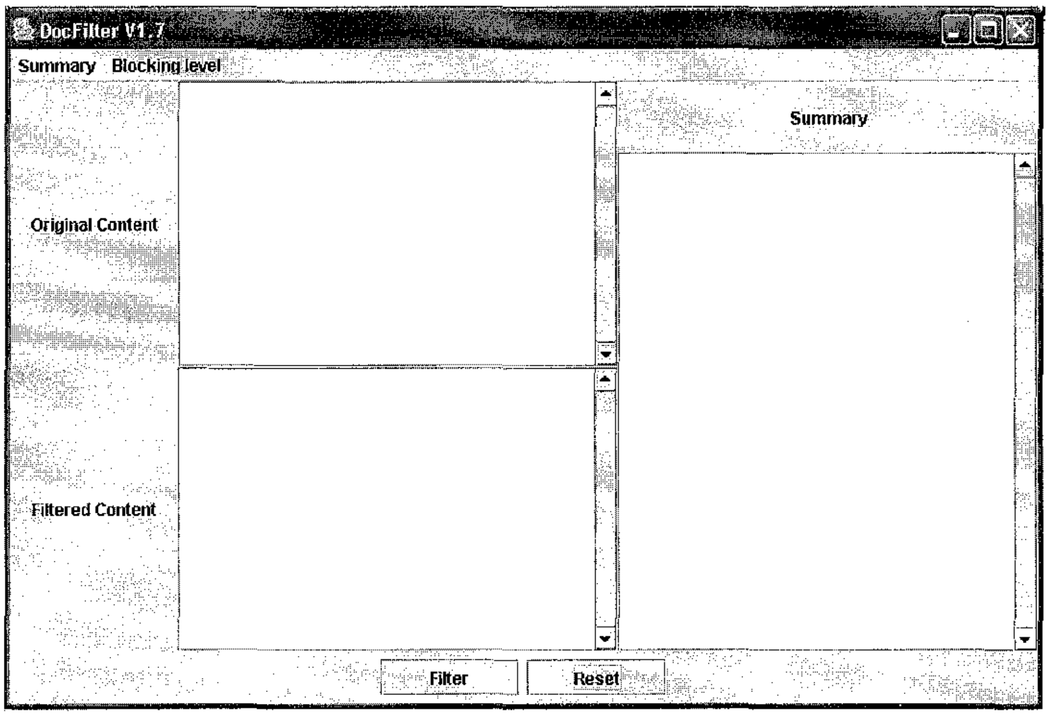
[21] Sets. Retrieved October 10, 2005 from the World Wide Web: <http://www.gobosoft.com/eiffel/gobo/structure/set.html>

[22] Class Hashset. Retrieved October 10, 2005 from the World Wide Web: <http://java.sun.com/j2se/1.5.0/docs/api/java/util/HashSet.html>

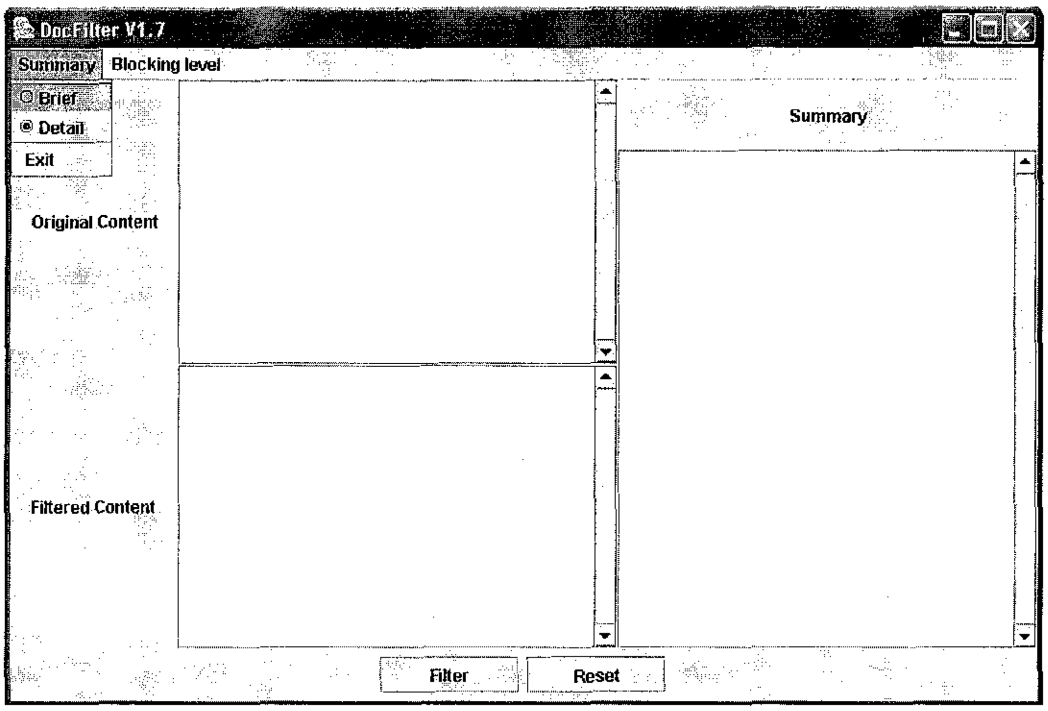
# APPENDIX I – TIMELINE

| Activities | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Planning** | | | | | | | | | | | | | | | | |
| Research Title for Initial Proposal | ■ | | | | | | | | | | | | | | | |
| Identify problem statement | ■ | | | | | | | | | | | | | | | |
| FYP Briefing | | ■ | | | | | | | | | | | | | | |
| Identify project objectives and project scope | | ■ | | | | | | | | | | | | | | |
| **Analysis** | | | | | | | | | | | | | | | | |
| Research (Journals and articles) | | | ■ | ■ | ■ | | | | | | | | | | | |
| Research on other related work | | | ■ | ■ | ■ | | | | | | | | | | | |
| Research and feasible study on development tool | | | | ■ | ■ | | | | | | | | | | | |
| Analyze research materials | | | | | ■ | ■ | | | | | | | | | | |
| Literature Review | | | | | ■ | ■ | | | | | | | | | | |
| **Design** | | | | | | | | | | | | | | | | |
| System Architecture | | | | ■ | ■ | | | | | | | | | | | |
| Learning Programming Language | | | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | | | |
| **Development** | | | | | | | | | | | | | | | | |
| Implement design into coding | | | | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | | |
| **Testing** | | | | | | | | | | | | | | | | |
| Stub testing | | | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | | | |
| Program testing | | | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | | |
| Revision of System | | | | | | | | | | | | | | | ■ | ■ |
| Preparation on Final Report/ Dissertation | | | | | | | | | | | ■ | ■ | ■ | ■ | ■ | ■ |

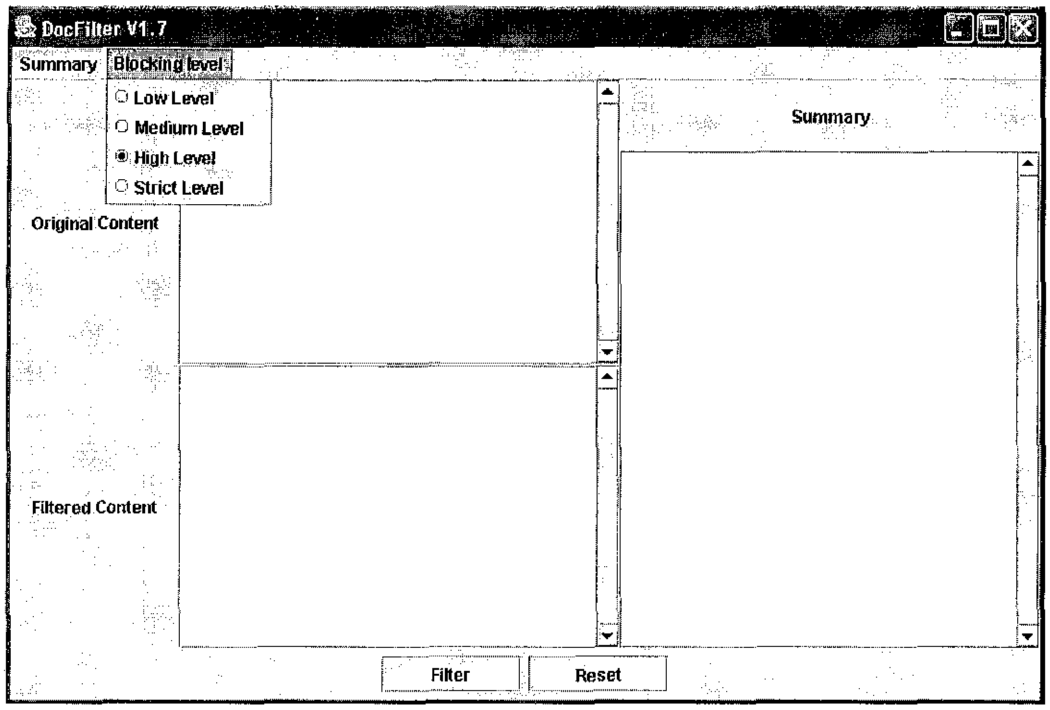
## APPENDIX II – DOCFILTER SNAPSHOT



DocFilter interface

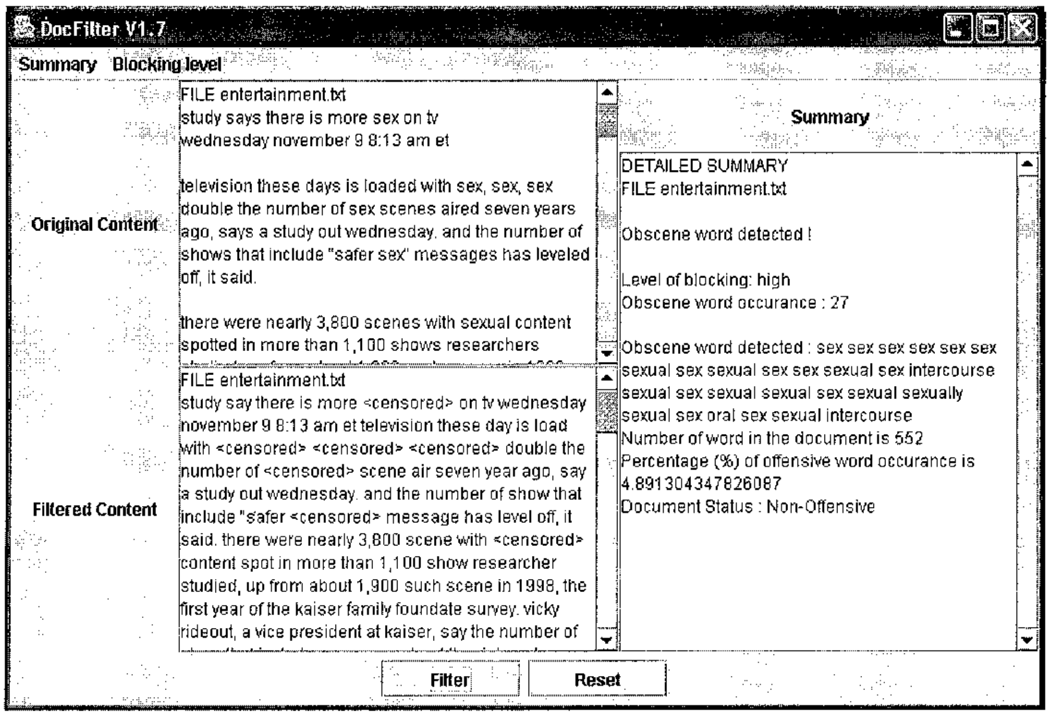


DocFilter's summary options. The two options are Brief summary and Detailed summary.



DocFilter's blocking level options. Four different blocking level provided are low level, medium level, high level and strict level.

# APPENDIX III – DOCFILTER OUTPUT SNAPSHOT



```
FILE entertainment.txt
study says there is more sex on tv
wednesday november 9 8:13 am et

television these days is loaded with sex, sex, sex
double the number of sex scenes aired seven years
ago, says a study out wednesday. and the number of
shows that include "safer sex" messages has leveled
off, it said.

there were nearly 3,800 scenes with sexual content
spotted in more than 1,100 shows researchers
```

```
FILE entertainment.txt
study say there is more <censored> on tv wednesday
november 9 8:13 am et television these day is load
with <censored> <censored> <censored> double the
number of <censored> scene air seven year ago, say
a study out wednesday. and the number of show that
include "safer <censored> message has level off, it
said. there were nearly 3,800 scene with <censored>
content spot in more than 1,100 show researcher
studied, up from about 1,900 such scene in 1998, the
first year of the kaiser family foundate survey. vicky
rideout, a vice president at kaiser, say the number of
```

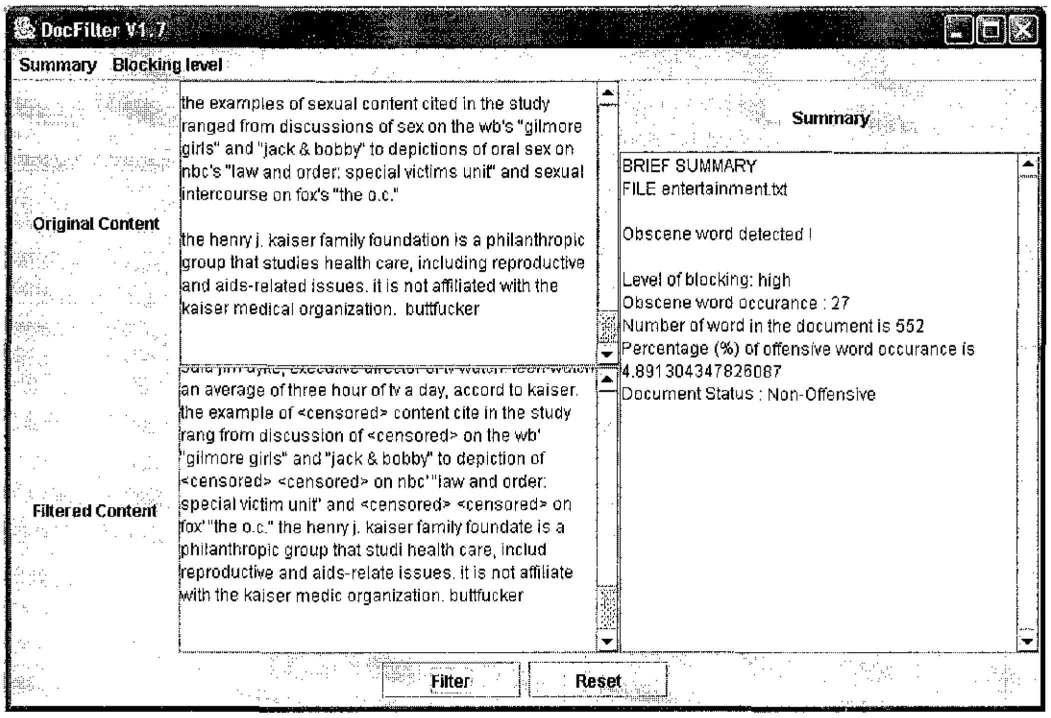
```
DETAILED SUMMARY
FILE entertainment.txt

Obscene word detected !

Level of blocking: high
Obscene word occurance : 27

Obscene word detected : sex sex sex sex sex sex
sexual sex sexual sex sex sexual sex intercourse
sexual sex sexual sexual sex sexual sexually
sexual sex oral sex sexual intercourse
Number of word in the document is 552
Percentage (%) of offensive word occurance is
4.891304347826087
Document Status : Non-Offensive
```

DocFilter's result with detailed summary.



the examples of sexual content cited in the study ranged from discussions of sex on the wb's "gilmore girls" and "jack & bobby" to depictions of oral sex on nbc's "law and order: special victims unit" and sexual intercourse on fox's "the o.c."

the henry j. kaiser family foundation is a philanthropic group that studies health care, including reproductive and aids-related issues. it is not affiliated with the kaiser medical organization. buttfucker

an average of three hour of tv a day, accord to kaiser. the example of <censored> content cite in the study rang from discussion of <censored> on the wb' "gilmore girls" and "jack & bobby" to depiction of <censored> <censored> on nbc' "law and order: special victim unit" and <censored> <censored> on fox' "the o.c." the henry j. kaiser family foundate is a philanthropic group that studi health care, includ reproductive and aids-relate issues. it is not affiliate with the kaiser medic organization. buttfucker

```
BRIEF SUMMARY
FILE entertainment.txt

Obscene word detected !

Level of blocking: high
Obscene word occurance : 27
Number of word in the document is 552
Percentage (%) of offensive word occurance is
4.891304347826087
Document Status : Non-Offensive
```

DocFilter's result with brief summary.