ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ (MSc)
στα ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Εξαγωγή Χαρακτηριστικών Όρων και Κατασκευή Προφίλ Χρήστη σε Πραγματικό Χρόνο με χρήση Δυναμικής Συλλογής Κειμένων για Εξατομίκευση Περιεχομένου στον Παγκόσμιο Ιστό»

Στεφανάτος Κων/νος
Μ3090001

ΑΘΗΝΑ, ΦΕΒΡΟΥΑΡΙΟΣ 2011

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ (MSc)
στα ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Εξαγωγή Χαρακτηριστικών Όρων και Κατασκευή Προφίλ Χρήστη σε Πραγματικό Χρόνο με χρήση Δυναμικής Συλλογής Κειμένων για Εξατομίκευση Περιεχομένου στον Παγκόσμιο Ιστό»

Στεφανάτος Κων/νος

Μ3090001

Επιβλέπων Καθηγητής: Βαζιργιάννης Μιχαήλ
Εξωτερικός Κριτής: Καλαμπούκης Θεόδωρος

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΘΗΝΑ, ΦΕΒΡΟΥΑΡΙΟΣ 2011

Real Time Keyword Extraction & User Profiling using Dynamic Corpus for Web Personalisation

Εξαγωγή Χαρακτηριστικών Όρων και Κατασκευή Προφίλ Χρήστη σε Πραγματικό Χρόνο με χρήση Δυναμικής Συλλογής Κειμένων για Εξατομίκευση Περιεχομένου στον Παγκόσμιο Ιστό

Athens University of Economics and Business

Department of Informatics

MSc Information Systems

Author:

Constantinos Stefanatos

Supervisor:

Michalis Vazirgiannis, Professor

Committee:

Theodoros Calampoukis, Professor

February 7, 2011

Abstract

Web personalization is the process of customizing a Web site to the needs of specific users, taking advantage of the knowledge acquired from the analysis of the user's navigational behaviour (usage data) in correlation with other information collected in Web context, namely structure, content and user profile data. ISPs have full knowledge of the visited Web graph by all the subscribers' navigational patterns and all users' demographic and context related data; as such, they can add recommended advertisements in every web page the user visits in real time. This gives a potential for an ISP Advertising framework that is targeted. The proposed system for personalization on the web will be implemented in two parts, the first situated on the ISP's server and the other on the router provided to the subscriber; the latter is the subject of this thesis. The system offers the ability to provide real time personalization services to customers. The system will collect information concerning the navigation and preferences of users as they browse the Internet. By utilizing usage-data, data mining techniques and learning procedures, user profiles will be created as well as appropriate sets of recommendations that will be related to both their long and short term interests. By extracting keywords from the current web page and scoring them according to a local dynamic term corpus, a web page vector representing the current web page is constructed. In similar fashion, a user profile vector is maintained and updated as the user browses more web pages. Both these vectors are transmitted to the ISP's server in order to find the most relevant advertisements. The recommended advertisements will be introduced in real time into the web page content as traffic is passing through telecommunication devices (such as a router) situated close to the user.

Ελληνικά

Η παροχή εξατομικευμένου περιεχομένου στο Διαδίκτυο είναι η διαδικασία τροποποίησης μιας ιστοσελίδας σύμφωνα με τις ανάγκες των χρηστών, λαμβάνοντας υπόψη τη γνώση που αποκομίζεται από την ανάλυση της συμπεριφοράς πλοήγησης ενός χρήστη (δεδομένα χρήσης) σε συνδυασμό με άλλες πληροφορίες που συλλέγονται από το περιεχόμενο ιστού. Πρωτίστως αναφερόμαστε στη δομή, το περιεχόμενο και τα δεδομένα που αφορούν το προφίλ του χρήστη. Οι πάροχοι περιεχομένου έχουν πλήρη γνώση του γράφου επισκεψιμότητας για όλους τους συνδρομητές τους, καθώς και όλα τα δημογραφικά στοιχεία που αφορούν τους χρήστες αυτούς. Επομένως, μπορούν να προσθέσουν εξατομικευμένες διαφημίσεις σε κάθε ιστοσελίδα που επισκέπτεται ο εκάστοτε χρήστης, σε πραγματικό χρόνο. Κάτι τέτοι δίνει τη δυνατότητα δημιουργίας ενός πλαισίου παροχής στοχευμένων διαφημίσεων στο επίπεδο του παρόχου. Το σύστημα που προτείνουμε σε αυτή τη διπλωματική και αφορά την παροχή εξατομικευμένων διαφημίσεων στο Διαδίκτυο, θα υλοποιηθεί σε δύο τμήματα, όπου το πρώτο βρίσκεται στον εξυπηρέτη του παρόχου και το δεύτερο στο δρομολογητή του χρήστη. Το δεύτερο τμήμα είναι και το αντικείμενο της διπλωματικής αυτής. Το σύστημα προσφέρει τη δυνατότητα παροχής εξατομικευμένων υπηρεσιών σε πραγματικό χρόνο στους συνδρομητές. Το σύστημα θα συλλέγει πληροφορίες σχετικά με την πλοήγηση και τις προτιμήσεις των χρηστών καθώς αυτοί περιηγούνται στο Διαδίκτυο. Χρησιμοποιώντας δεδομένα χρήσης, τεχνικές εξόρυξης δεδομένων και διαδικασίες μάθησης, θα δημιουργούνται προφίλ χρηστών καθώς και κατάλληλα σύνολα προτάσεων που θα σχετίζονται με τα ενδιαφέροντά τους σε μακρόχρονη αλλά και σε βραχύχρονη βάση. Μέσω της εξαγωγής λέξεων κλειδιών από την εκάστοτε ιστοσελίδα και τη βαθμολόγηση αυτών βάσει μιας τοπικά διατηρούμενης δυναμικής συλλογής όρων, κατασκευάζεται μία διανυσματική αναπαράσταση τηες ιστοσελίδας αυτής. Με παρόμοιο τρόπο, κατασκευάζεται και ενημερώνεται το προφίλ χρήστη, καθώς ο εκάστοτε χρήστης περιηγείται σε περισσότερες ιστοσελίδες. Τα δύο αυτά διανύσματα μεταδίδονται στον εξυπηρέτη του παρόχου ώστε να βρεθούν οι καταλληλότερες διαφημίσεις. Οι διαφημίσεις αυτές θα εισάγονται σε πραγματικό χρόνο στο περιεχόμενο

της ιστοσελίδας, καθώς η κίνηση του χρήστη διέρχεται από τηλεπικοινωνιακές συσκευές που βρίσκονται εγγύτερα στο χρήστη (όπως ο δρομολογητής που χρησιμοποιεί).

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
CSS	Cascading Style Sheets
CTR	Click Through Rate
DF	Document Frequency
DM	Data Mining
DPI	Deep Packet Inspection
DR	Dimensionality Reduction
eCAP	Embedded ICAP
FE	Feature Extraction
FS	Feature Selection
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HTTP Secure
IAB	Interactive Advertising Bureau
ICAP	Internet Content Adaptation Protocol

IDF	Inverse Document Frequency
IP	Internet Protocol
IR	Information Retrieval
ISP	Internet Service Provider
JIT	Just In Time
JSON	JavaScript Object Notation
ML	Machine Learning
PCA	Principal Component Analysis
SOAP	Simple Object Access Protocol
TF	Term Frequency
TF-IDF	Term Frequency - Inverse Document Frequency
TM	Text Mining
URL	Uniform Resource Locator
WCM	Web Content Mining
WM	Web Mining
WSM	Web Structure Mining
WUM	Web Usage Mining
WWW	World Wide Web
XML	eXtensible Markup Language

Chapter 1

Introduction

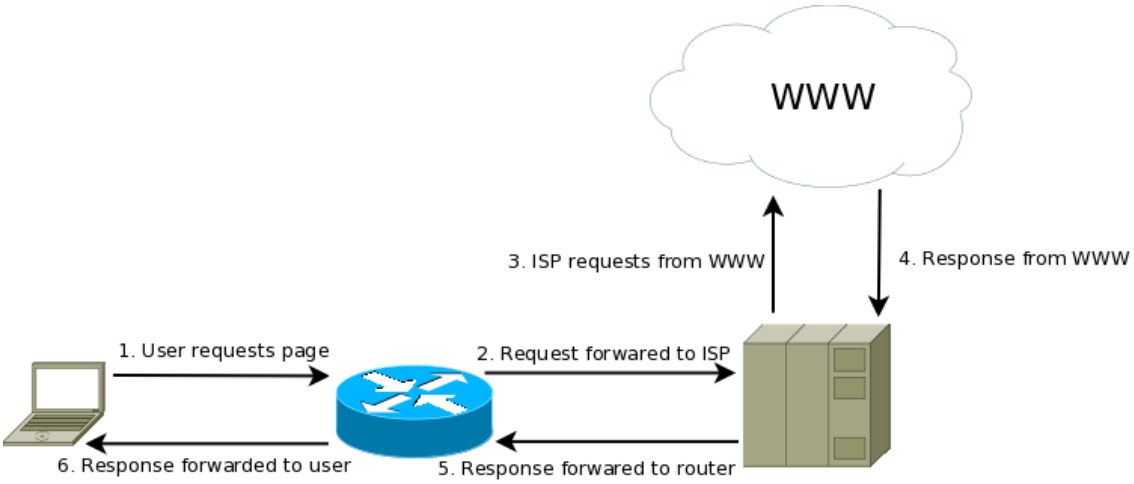
Nowadays, on-line advertising poses the backbone on which a large part of the Internet's ecosystem is based on, concerning the way sites and companies sustain themselves financially. Many web site owners and operators, alongside all the major search engines (Google, Yahoo!, Bing) act as advertising distributors, serving mostly textual advertisements. The two major advertisement ditribution channels employed today are sponsored and contextual advertising. The profits originating from on-line advertising are especially great and are enjoyed by all the participants.

Due to the sheer volume of content available through the Internet, and to the vast majority of users with access to that content, a means to reduce the Information Overload effect is of great importance, so as to facilitate each user's ability to find the information they desire, as easily and efficiently as possible. In respective terms, there is a need to have advertisements that are as targeted as possible, in order to maximise each of the interested parties' profit, as well as the enjoyment of the users.

The approach that has been found and offers all of the above, is that of content personalisation, while concerning advertisements, it is referred to as personalised advertising. As a term, personalisation has existed since the mid nineties. It can be defined as the procedure with which a user is supplied with the information they require, the moment they require it. It is usually accomplished by gathering data

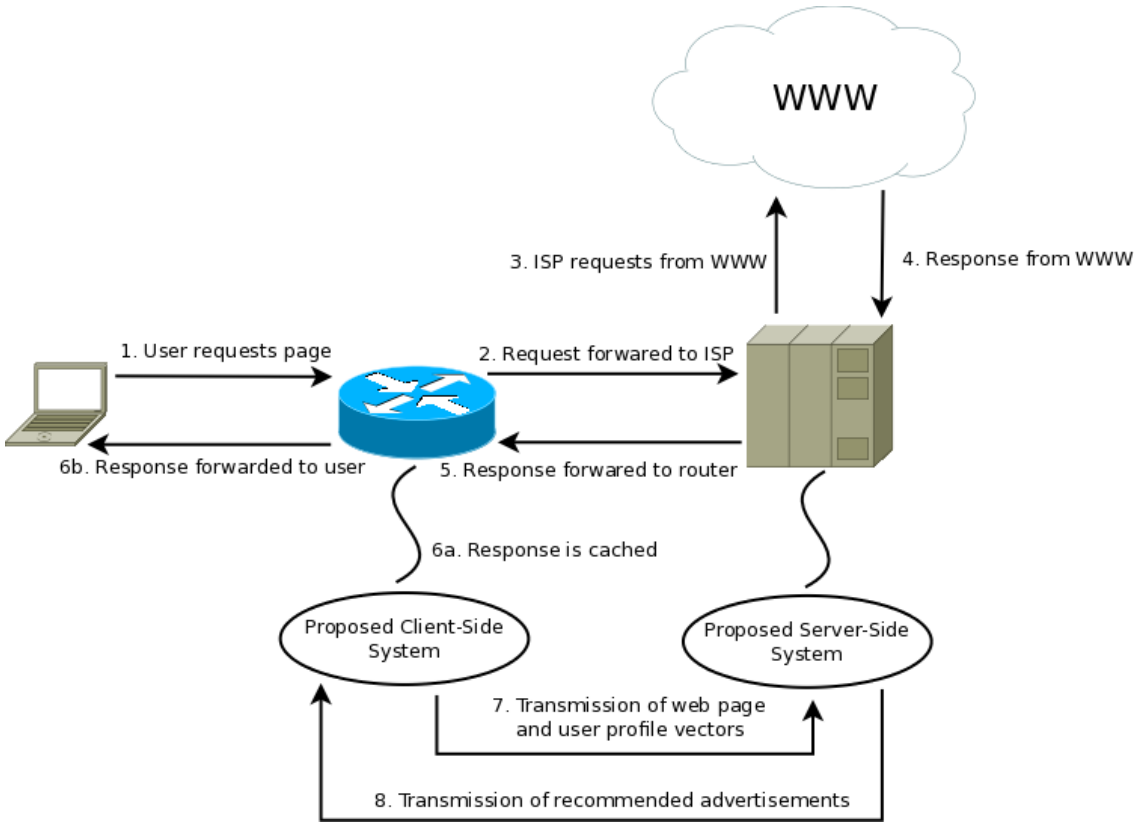
concerning the user, in the form of a profile representing the user. By employing the user profile, a system can perform a search and retrieve that content that is more appropriate for the specific user, and thus provide them with the most appropriate content. A similar approach is followed when dealing with personalised advertising.

Figure 1.1: Regular request - response procedure



In this thesis, our goal is to implement a system that serves personalised advertisements, operating at the Internet Service Provider (ISP)'s level, in real time. It is divided in two segments, one operating on each user's router, acting as their entry point to the ISP's server, and the other operating at one or more of the ISP's servers which are the gateways to each user's actual connection to the rest of the Internet. Specifically, on the router segment, which is the focus of this thesis, a user profile is created and maintained by being constantly updated, while at the same time, the most significant keywords are extracted from every web page the user visits, thus portraying their current focus of interest. By employing these two pieces of information, the router transmits them to the server segment, which is tasked with matching the most relevant advertisements with them and returning those advertisements back to the router; finally, these advertisements are displayed on the

Figure 1.2: Altered request - response procedure



web page the user is viewing, in an appropriate section. The main requirements of the aforementioned system are speed and precision, since everything will be operating in real time.

In order to better illustrate how the system will be affecting the regular flow of traffic normally employed when a user makes a request to browse to a specific web page, two figures are going to be shown: the first (figure 1.1) illustrates how a request is usually handled in a normal situation, while the second (figure 1.2) shows the different steps that the proposed system introduces.

Thesis Organisation

The structure of the current thesis is organised in separate chapters, each dealing with a specific subject. We are now going to present each of the chapters and give a short description of what each chapter describes.

In the second chapter, the necessary theoretical background is given, so as to provide the reader with a solid understanding of the concepts mentioned in this text, while at the same time presenting some of the difficulties that had to be dealt with, leading to the justification of what was chosen to circumvent them.

In the third chapter, the related work is presented, with an analysis of the problems that they had to deal with and the reasons each of them failed to serve their purpose.

In the fourth chapter, a thorough view of the selected system architecture is provided, by outlining its structure, analysing the way it has been implemented and finally describing the problems that had to be dealt with. We also present the advantages the proposed system has in comparison to the commercial solutions presented in the third chapter, along with some alternatives approaches that were considered during its design process.

In the fifth chapter the evaluation method is described, along with the results obtained by it.

Finally, a presentation of this work's conclusions is given in the sixth and final chapter, with some suggestions for future extensions and possible improvements.

Chapter 2

Theoretical Background

Information Retrieval (IR) is mainly tasked with searching and retrieving the appropriate information requested. The search and retrieval can be performed on data that are present in a textual form, in databases, or even in the World Wide Web (WWW). It is a part of computer science that has bloomed, especially in recent years, mostly due to the Internet's wide uptake and the fact that it serves as an information delivery platform, available on a twenty four hour basis, with a global reach. The available volume of data accessible through the Internet has grown rapidly, especially during the past decade, since the ease with which an individual can publish whatever kind of information they want to, is phenomenal. As such, the need for the proper IR tools has arisen, that should be available at all times to all users, through the form of search engines.

Since the volume search engines need to process increases at an exponential rate, as time goes by, the process of categorisation and proper selection of results that correspond to the user's request, needs to be automated to the largest possible extent. This has led to the development and employment of a part of Artificial Intelligence (AI) called Machine Learning (ML), for the purposes of IR. ML offers automated means with which a system, after completing a training period using proper data and methods, can perform categorisation of the data it is given.

The combination of IR and ML has led to the development of Data Mining (DM) techniques; DM is another scientific area, tasked with pattern recognition of differ-

ent kinds of data and employed in many different operational environments, with the most prominent being that of searching large data sets. There is a separate part of DM which focuses on textual data, called Text Mining (TM), which is responsible of selecting and retrieving information from text sources. It includes various text preparation methods, such as structuring, parsing, determining various patterns and finally evaluating and interpreting the results. In order for the results to be deemed appropriate, the measures of Precision and Recall are employed, which are founded by the scientific area of IR. As a measure, Precision can be described by the following formula:

$$\text{Precision} = \frac{|\text{relevant documents} \bigcap \text{retrieved documents}|}{|\text{retrieved documents}|}$$

On the other hand, Recall can be represented by the following formula:

$$\text{Recall} = \frac{|\text{relevant documents} \bigcap \text{retrieved documents}|}{|\text{relevant documents}|}$$

There is another specialisation of IR based on TM that is called Web Mining (WM), which can be separated in three different categories of use:

1. Web Usage Mining (WUM): tasked with the examination of the information contained in web server logs, it provides a way for user profiles to be constructed.

2. Web Content Mining (WCM): tasked with finding relevant information contained in all forms of data that are distributed through the Internet, such as texts, images, sounds, etc.

3. Web Structure Mining (WSM): this technique tries to extract conclusions using the structure of the Internet, by taking advantage of the structure of the documents distributed (HyperText Markup Language (HTML), eXtensible Markup Language (XML), et al) and analysing the interconnections between the hyper-links between different web pages.

The advantages of WM are many and it is worth noting that as a technique, WM has a certain commercial interest, since by employing such methods, the serving

of personalised content and advertisements is made possible; in that way, companies can benefit from providing their clients with the most relevant content to their needs, thus increasing the companies' profit and at the same time the clients' satisfaction. The application of WM techniques can also help in recognising patterns of use of the Internet, since the users can be classified based on the content they access and their general movement across the Internet, thus making possible the detection and subsequent prevention of various threats (abusive behaviour, spam, bot-nets, et al).

On the other hand, the disadvantages that can be attributed to WM, are based mostly on an ethical aspect, and not so much on a technological one. Specifically, it can be used in such a way that a user can be regarded as being under a form of surveillance, which if kept unknown to them, can be considered as violating their privacy. On a similar approach, companies that are employing WM techniques can sometimes gather user related data which they can use for a purpose quite different to what they claim to, thus harming the end users. As an example we have the selling of user profiles to advertising companies without their knowledge, and subsequently, their consent.

2.1 Dimensionality Reduction

In order to complete our reference to DM in general, along its more specific derivatives that are going to be used in this thesis, there is need to present in some detail a procedure that is necessary in every successful DM attempt. Due to the large volume of data that need to be processed each time, there is a need for a method that allows us to focus on the appropriate data for each occasion, depending on the task at hand. On a similar basis, due to the fact that data are usually multidimensional, we need to have a means to concentrate on these dimensions that possess the most relevant information for our purposes. In order to accomplish the aforementioned targets, various procedures and methods of Dimensionality Reduction (DR) are followed. DR as a term, originates from the scientific area of Statistics and serves the purpose of reducing all possible variables of a problem, in order to ease the

problem's analysis and solution. By applying such methods, one is able to form the most complete possible depiction of a problem while at the same time doing so in the most computationally efficient way. DR can be divided in two categories, those of Feature Extraction and Feature Selection.

2.1.1 Feature Extraction

Feature Extraction (FE) aims to transform the processed data from their initial multidimensional form to one with reduced dimensions, in order to accentuate the most important parameters of each problem. The methods that are applied in order to accomplish FE can be divided in two broad categories, those of Linear and Non-Linear DR.

1. In the category of Linear reduction, the major method followed in most practices is called Principal Component Analysis (PCA). According to this method, a linear matching of possibly independent data of the initial problem space to a smaller number of independent variables in a smaller dimension space is performed. These components are matched to the smaller dimension space with some vectors, which are usually orthogonal in relation to each other, and should they be transformed again, have the capability to describe the majority of the variance of the initial problem's dimensions. This is possible since each of these vectors describes the largest part of the variance of the data it represents in the reduced dimension space.

2. In the Non-Linear category, the problem's dimensions are usually far too many to easily interpret, since the more dimensions a problem has, the more difficult it is to keep track of and describe. When dealing with such a case, some assumptions need to be made as to where the data that interest us are located, in order to focus on that subset of dimensions. Non-Linear methods are usually extensions of Linear methods. As an example, we shall name that of the Kernel based Component Analysis, which is based on the PCA method we mentioned above, but differentiates itself by employing a kernel,

whose responsibility is to transform the initially linear transformations of the Linear method to a dimension space with Non-Linear matching.

2.1.2 Feature Selection

Feature Selection (FS) follows a different approach to that of FE, since it attempts to locate a subset of the initial problem's variables and work with that subset in order to solve the problem. There are two major implementation approaches for FS:

1. Feature Ranking: in this approach, all the features of the problem are weighted by the employed weighting method and are subsequently compared against a threshold value. If the weight of a feature is larger than the threshold, then that feature is selected, while on the other hand, it is not.

2. Subset Selection: in this approach, various searches are performed on the total set of problem variables, in order to detect and select the optimal subset of those. The searching algorithms that can be applied vary from exhaustive search (which is usually less efficient the more variables it has to deal with) to complex genetic search algorithms. What is usually followed in practice though, for the sake of performance, is the selection of a satisfying subset of variables, instead of the optimal one.

In order to evaluate the features that are selected, independent of the approach followed, some metrics are usually applied, with some of the most prominent ones being:

1. Chi-Squared Distribution

2. Information Gain

3. Accuracy

4. Document Frequency

In this paragraph we are going to present some details of the aforementioned metrics [1]. The Chi-Squared Distribution is a common statistical analysis tool that measures the deviation from the expected distribution, by assuming that the features' frequency is independent of the class' value. This particular metric does not perform well in text applications, since in such cases, some features (words) tend to have very small frequencies. Information Gain measures the reduction of entropy that occurs due to a feature being present or not; that is, it measures the effect a feature's actual selection or not has via means of entropy. Accuracy, as a metric, evaluates the expected accuracy that a classification algorithm would have should it be based on one feature at a time. Finally, Document Frequency is a simplistic metric, since it is based solely upon the number of documents in a corpus that contain a given feature (word).

2.1.3 Text Feature Selection

In order to concentrate on the situations where the problem at hand concerns the selection of features from a text source, we need to keep in mind the fact that the number of texts we need to deal with is usually large and most of the time, these texts are characterised by a large and varying word count. In cases where the data we need to examine are located in a textual form, we need a way to reduce the dimensions of the problem in order to accentuate the features that are of more importance and select them instead of others. DR in text based problems is usually accomplished by following some standardised procedures, which are usually deemed necessary, before any further text processing is to take place. These procedures are the following:

1. Lower casing text characters
2. Stop word removal
3. Word stemming

In order to provide the reader with a more complete understanding of the purpose each of the aforementioned procedures serves, we shall present them in the

following sections.

2.1.3.1 Lower casing text characters

One of the most basic and common procedures usually followed when preparing a text document for FS, is that of converting the characters of the text to their lower case counterpart. Since the actual use of upper or lower case can vary in a great way, lower casing a text's characters simplifies the number of possible terms in the document, since words written in a different case could be interpreted as different words, despite of the fact that semantically, they are referring to the same term. For some examples, see table 2.1.

Table 2.1: Lower case example transformations

Original Case	**Lower Case**
Computer	computer
HTML	html
eCAP	ecap

2.1.3.2 Stop Word Removal

When attempting to index a text document during FS, some common words - often referred to as stop words - tend to appear that despite their high occurrence rate, provide no actual value to the document and should not be considered as features of that document (see table 2.2 for a sample selection of such words). Moreover, when a text document is large in size, processing terms such as these can have a great impact on the overall performance of FS, since the volume of words that need to be processed is increased, thus burdening the system or the person performing the FS. Due to the above, a common practice is to remove such words from a text document, before proceeding to index and select the most prominent features contained in that document. The most common approach in removing these stop words is to make use of a dictionary; this dictionary is basically a list of common

words which are searched for and if found, are removed from the text, before proceeding with any further processing. There is a special need to mention that each different language, as is expected, has a different set of such words, while at the same time, some words can be considered common or not in the same language, according to context: for example, the word ”computer” cannot be considered a common term in a novel written in the English language. On the other hand, when performing FS on a text that is written on the subject of computer science, such a term is expected to be present many times, and can subsequently be treated as common.

Table 2.2: Sample list of stop words

and	for	the
can	do	my
who	where	why

2.1.3.3 Word Stemming

In any text document, independent of the language it is written in, it is expected to find some words that are nothing more than different forms of the same word, or some words that originate from a common word. That common or same word is often referred to as the stem word, and in IR, it is quite common to treat various derivative words as semantically equal to their stem word. In order to accomplish better FS from a text source, a procedure known as word stemming is usually followed, which attempts to extract the stem of each word in the document and use that stem instead of the actual word itself, in any further processing attempt (see table 2.3 for examples[1]). In that way, there is a more evident chance that terms with a high occurrence will be selected as important features from that text. By using the stem instead of the actual word, there is another gain, in terms of computational costs when processing text documents for FS: the spatial (in memory

[1]The stemming algorithm used to produce these sample stems is the Porter algorithm

size) and temporal (in processing time) requirements to create the list of the most prominent features in a text are reduced. Since this procedure is often employed and offers many benefits, a few different algorithms have been developed that offer such a functionality and are available to use in a wide range of programming languages. The most prominent algorithms used today are:

1. Porter
2. Lovins
3. Paice/Husk
4. Lancaster

Table 2.3: Word Stemming examples

Original Word	**Resulting Stem**
computer	comput
computational	comput
weight	weight
weighting	weight

2.1.4 Feature Weighting

After completing all the aforementioned procedures necessary to prepare the text for FS, the next step is to set up the necessary means with which features will be selected. The most common procedure that does this is the construction of a term index that contains all the terms in the text and the subsequent weighting of each term, according to a selected method or criteria.

Concerning the models that are used in such a procedure, the most common ones are the following: those that make use of a document - term matrix, those that are based on the bag of words concept and finally those that represent each

term in the vector space [2, 3]. In the models that make use of a document - term matrix, each term is placed in a matrix, where each column represents a term and each line represents each distinct document that belongs in the indexed collection. Consequently, each cell contains the number of occurrences of each term in each document. In vector space models, each term is represented by a vector whose features (length, angle) are determined by the weight of the term in the text. Finally, in the bag of words model, which is the simplest, all the distinct terms present in the text are concentrated, in no particular order. These types of models are most frequently used in text filtering applications, for example for detecting and filtering spam messages in an email application.

As for the actual procedure of weighting the terms of each document, the procedures followed vary from the simplest approach of weighting each term by its frequency in a document, to the more complex one of assigning predefined weights to each term, determined by their importance or use in each document. In the simple approach where the term's frequency is used as the term's weight, this is usually normalised to the total number of words that exist in the document, instead of the raw number of occurrences each term has. The other approach, where special weights are assigned to each term, is normally used when we are searching for specific terms in a document or in cases where the documents we are working with are structured and tend to have discreet elements (such as titles, headings, et al).

Under some circumstances, the terms of a document are not inspected independently, but instead are compared collectively against a collection of documents, called a corpus. In such cases, the approach of assigning a weight to a term has no significant meaning, since searching in a corpus returns different results each time. To circumvent this problem, term weighting in corpus based problems is usually performed via the Term Frequency - Inverse Document Frequency (TF-IDF) approach [4]. This approach examines all of the documents that belong in the corpus and returns a value that tends to be greater for terms that are frequent in a specific text but rare in the collection as a whole. On the contrary, this value tends to be lesser for terms that occur frequently in the collection, thus determined

to be quite common and not very distinctive of each document. What the TF-IDF approach does, in simple words, is that it can clearly indicate whether a term can be regarded as distinctive and representative of the document it is found in.

2.1.5 Feature Selection from Web Pages

Web pages represent a structured text document, since they tend to be written in HTML, which follows specific rules concerning its syntax and uses a set of predefined tags that act as specific attributes to the text they enclose. Keeping their structured form in mind, FS from web pages can be based on the FS methods that we presented earlier concerning text sources, with the addition of some extensions [5] that take advantage of their structured nature.

Thanks to the structured nature of web pages, it is quite easy to extract certain features from them, which means that we have to treat them differently concerning the weighting scheme we are going to use against them. This poses the most significant difference FS from web pages has when compared to FS from simple text documents; terms that are contained in tags of certain interest (such as page title tags, headings, links, etc) need to be treated differently that plain text, since due to the tag they belong to, they tend to have special meaning.

The most common approach used when performing FS from web pages, is to look for and assign a different weight to the following tags [6]:

- Title of the page (<title>)
- Headings (<h[1-6]>)
- Hyper link text (<a>)
- Keywords describing the page (<meta>)

Most of the approaches in literature (as an example, see [7]) search for these elements and the text included in them, and assign different weights to that text when compared to the rest of the text of the web page – that is the text that is contained in the <body>tag when dealing with HTML pages – which is usually treated as

plain text during FS. It is worth noting that there is another preparatory procedure, besides the ones we mentioned before (lower casing text, stop word removal, word stemming), that is specific to web pages and needs to be performed before the FS process can proceed: that is the removal of all the markup contained in the text [6], which means stripping all the HTML tags from the web page document, along with all the content that is not eligible for FS: we are referring to all the ”hidden” text that can be contained in a web page, such as JavaScript code, normally contained in the <script>tag, and code that refers to the web page's appearance (Cascading Style Sheets (CSS) style code), normally included in the <style>tag.

2.2 On-Line Advertising

During the past years, due to the advent of high speed wireless networks and of portable devices with Internet connectivity, we are gradually being lead to the ubiquitous computing age. As a natural consequence, the Internet has managed to become an irreplaceable part of everyday life in a global scale, offering everything, regardless of time or location.

Due to its large number of users, the Internet acts as an excellent field for business activities, since it has been possible for some time now to purchase a wide range of products and services on-line. And as is expected, where there is business activity, there is always need for advertising. The Internet can act as an excellent means of advertising, since an on-line advertisement is available globally, as soon as it is published, with no delays and with the potential for instant attraction of clients. It is worth noting another important factor that makes on-line advertising such an attractive choice. That is the optimisation of the price the advertised party is called to pay in order to be advertised: in contrast to what is usually required in traditional advertising (print, television, radio) where each advertisement's cost is calculated by a variety of factors (such as the section of the page the advertisement is going to be placed in a newspaper, the time of day the advertisement is going to be ”aired” on the radio or television, how many times it is going to be ”aired” each day, et al), the cost of an on-line advertisement is usually determined by one

of the following methods [8]:

1. Pay per click
2. Pay per impression
3. Pay per action

In the first case, the advertised party is billed every time an end user clicks on their advertisement. In the second, they are billed every time the advertising distributor selects their advertisement for display in a web page while in the last one, they are billed only if the advertisement actually led to any kind of transaction with an end user. In conclusion, on-line advertisements present a very attractive means of advertising, especially for advertised parties, due to their flexible cost requirements.

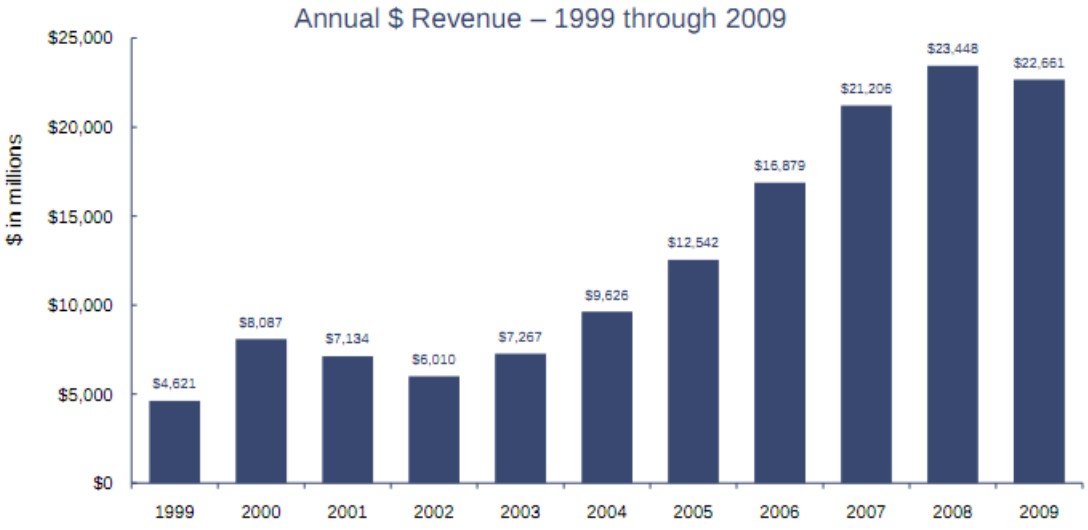
In order for the reader to have a better perception of the investments conducted in the on-line advertising business sector, some financial data are going to be presented. According to the official report of the Interactive Advertising Bureau (IAB) concerning the first semester of 2010, an approximate total of 12 billion dollars in revenue has been totalled to originate from on-line advertising, just in the United States, showing an increase of approximately 1.2 billion when compared to the first semester of 2009 [9]. As an indication, we need to mention that the annual revenue for 2009 [10] was equal to 22.7 billion dollars. The increasing trend in revenue originating from on-line advertising during the decade of 1999-2009 is clearly visible in figure 2.1.

It is clear that the revenue in on-line advertising is great, both for the advertising and the advertised parties. It is also worth noting that on-line advertising provides financial support and sustenance to a large part of the Internet's ecosystem, since a large number of web pages cover their operating expenses (hosting, equipment, et al) by displaying advertisements.

Online advertising comes in a variety of forms:

- Simple text
- Static images

Figure 2.1: Historical annual revenue trends in on-line advertising



- Multimedia (video, sound)
- Interactive (Flash based)

For the purposes of this thesis, we are going to focus on the simplest form, that of simple textual advertisements.

The predominant approaches concerning the advertisement distribution channels are the following [11]:

1. Sponsored Search: advertisements are provided in search engine result pages, depending on the query submitted by the user. The majority of current search engines, including all the major ones (Google, Yahoo!, Microsoft) support such advertisements, while it is worth noting that they usually play a twofold role: besides the obvious role of being a search engine, they usually act as an advertisement distributor.

2. Contextual Advertising: advertisements are placed in the content of any web page. In this approach, there is usually an intermediary, called the advertisement network, responsible for the optimal selection of the most relevant

advertisements with the twofold purpose of increasing revenue (usually split between the advertised party and the advertisement network) and improving the experience of the end user (and potential customer). It is worth noting that most search engines nowadays act as advertisement networks; taking into consideration the aforementioned facts, it is easy to deduce that search engines are the major players in the online advertising business.

In the following sections, more details concerning the aforementioned approaches are provided.

2.2.1 Sponsored Search

Up until now, the majority of the profits originating from online advertising has been attributed to sponsored search, that is advertisements displayed in search engine result pages [10]. The advertisements displayed in sponsored search, as has been mentioned already, are selected based on the search query submitted by the user. The advertised parties submit their advertisements to the search engine operator/owner, and the search engine analyses each search query it is requested to process, and selects those advertisements that consist a better match with the search query, from the pool of available advertisements. In most advertisement networks, the price each advertiser is called to pay for each click an advertisement receives is usually determined by a bid process, where the advertised parties bid for a specific phrase, called bid phrase [8], which is used to represent the advertisement. The place each advertisement receives during the aforementioned selection and the actual display on each page is determined by its relevance with the search query along with the price paid by the advertised party to purchase the associated bid phrase. Each advertisement can be associated with an infinite number of bid phrases. Along with the search query terms, an advertisement is characterised by a title (which is usually displayed in bold letters) and a respective description, which is usually short (less than 120 characters) and is displayed under the title in the results page. Finally, each advertisement is associated with the target Uniform Resource Locator (URL) [12], where the user will be redirected to, should they click on the advertisement.

Sponsored search advertising has some underlying hazards for content publishers [12], since the display of an irrelevant advertisement can have an impact on the site's credibility, and in turn, the site's market share. This indicates the importance of investing in the research and evolution of advertisement recommendation systems, in order to minimise the probability of displaying irrelevant advertisements. Through these investments, publishers are basically investing in the maintenance of their credibility and in the end users' positive reaction to advertisements as a whole. The latter could very well lead to increased profits for both the publisher and the advertiser.

2.2.2 Contextual Advertising

Contextual advertising is based on the same concepts as sponsored search advertising [8]. The bid phrases we mentioned before are not used in contextual advertising, although they can provide a concise description of the targeted recipient of each advertisement, according to the advertiser. As such, they can be used during the advertisement selection process, where according to research they are an important feature.

Contextual advertising as an approach, presents better potential for success than sponsored search, since the majority of users online spend more time on web pages that have actual content than on search engine result pages [13]. Nevertheless, they present a much more sophisticated problem to solve, since while sponsored search advertisements are matched against a user's search query, contextual advertisements need to be matched against a web page's content, which has the form of complete text and is therefore much more complex to analyse, instead of the simple keywords used in the case of sponsored search. Due to the simplicity of the approach of sponsored search, a common approach in contextual advertising is to attempt to represent the web page as a set of key words [13], that can sum the web page's content in the best way possible, and subsequently use those key words during the matching process for the selection of advertisements. In short, there is an attempt to follow a similar approach to sponsored search [8, 12], where the search query is substituted by the web page's content. The most

important features to consider when selecting key words to describe a web page, are the frequency of a candidate word in the text and the frequency of that word in the total document collection (when dealing with corpus based approaches).

The majority of approaches used in contextual advertising [14] are based on finding common words or phrases in web pages and advertisements, or on a combination of contextual and syntactic factors. Advertisements and web pages are usually represented as vectors in a vector space [8]. Subsequently, most of these approaches attempt to calculate the similarity rate between these two vectors; in other words, they attempt to find the advertisements that have the shortest distance from the web page vector [11]. The most common metric used in such approaches is the metric of cosine similarity [12]. For performance and scalability reasons, inverted indexes and efficient similarity search algorithms are used. A drawback this approach has is that it is based in a priori knowledge and it does not take advantage of facts such as an advertisement's actual display or the number of clicks it receives.

The implicated parties in contextual advertising assume one of the following four discrete roles [8]:

1. The content publisher: the owner of the web page on which advertisements are to be displayed. The content publisher's main targets are to maximise the profit gained from advertisements and to provide a satisfying experience to the end user.

2. The advertiser: they are the ones who provide the advertisements. They are usually occupied with the organisation of various advertising campaigns, where they collect and promote advertisements that are related to a certain subject for a certain period of time (e.g. portable computers during the beginning of a new academic year). Their main target is to promote products or services.

3. The advertising network: it usually acts as an intermediary between the advertiser and the content publisher and is responsible with the proper selection and display of advertisements. The profit gained by advertisements

is split between the advertising network and the content publisher.

4. The end user: visits the content publisher's web page in order to gain access to that page's content.

The advertising network model tries to benefit each of the implicated parties, since the clicks an advertisement receives yield profits not only to the content publisher and the advertising network as we have already mentioned, but to the advertiser as well, since they are recipients of traffic toward their web pages. Contextual advertising systems process the content of a web page in order to extract/select the most representative features on that page and then search through the available advertisements in order to find those that closer matches to these features.

2.3 Personalised Advertisements

The big gamble for content providers on the Internet is to manage to transfer the most appropriate content to end users, based on those users' preferences, needs and behaviour. The creation of a personalised experience for each user is a great advantage to all those involved:

- to the provider, their content is accessed by people interested in it,
- to end users, where the benefits are summarised in their time being spent qualitatively, since they gain access to the content they care more for, instead of having to waste time to filter through a large number of web pages in order to find the ones that truly matter to them.[2]

Taking into consideration the increase in business and especially commercial activities that happen on-line during the last years, one of the most rapidly evolving sectors where content personalisation occurs for end users is that of advertising. This sector is of major research and commercial interest, since advertisements, if targeted correctly (shown to people according to those people's interests

[2]The latter is mentioned in literature as Information Overload, and is a natural consequence of the huge volume of information that is distributed through the Internet.

and preferences), have great potential of increasing the profits of advertised parties and as a consequence, of the advertisement distributors. The end users enjoy the benefits as well [15], since they are faced with advertisements that can have actual value to them, instead of treating them as some kind of disturbance that happens while on-line [16].

The major advertisement providers on a global scale are the companies that operate search engines (Google, Yahoo!, Microsoft, et al), since through their web pages, which nowadays act as entry points to the Internet (the majority of users on-line are driven to the desired web site through a search engine, except for those rare occasions when they are aware of the complete URL for that site), there is traffic of volume equal to millions every day.

In order for them to maximise the profit for themselves, and for their clients, they need ways to provide the most relevant and targeted advertisements possible to their users. Consequently, they need a means to provide personalised advertisements. In order to be able to track the preferences or the behaviour of a user, a user profile needs to be constructed and maintained.

2.3.1 User Profiling

The importance of user profiling for personalised advertising has been stressed already in the aforementioned section. A profile that serves such purposes usually includes features regarding the user; such features are related to the user's behaviour on-line, their preferences, their interests and their habits. Alternatively, a user profile can contain all of the above, for a specific web page, since it is quite common for a web page to maintain a profile for each of the users that visit it, in order to provide them with a personalised experience.

The approaches followed in order to create a user profile fall between two major categories: those of explicit and those of implicit feedback [17, 18, 19]. In the first category, the user is actively involved during the construction of their profile, while in the second, the profile is constructed automatically by the system, without any interference from the user. The main reason people prefer the latter, is the fact that explicit feedback methods are widely regarded as placing an extraneous

burden on the user, in order for them to provide the necessary information regarding their profile; users mostly prefer less time consuming methods [17, 18, 19]. Another important factor is that users tend to disregard any further interaction with a system, as soon as they gain access to the information they were originally using the system for. It is worth noting, that according to the comparison presented in [19], implicit feedback methods show a tendency to provide better profiles than explicit methods. One further categorisation in profiling approaches is that concerning the way the profile is maintained and updated: profiles fall into two broad categories in that respect, static and dynamic [19]. In the former category, the profile is not updated once constructed, while in the latter, it is updated in either a regular or more sparse schedule. The first is called a short term profile, while the second a long term profile. Taking into consideration all of the above, a short presentation of all the major techniques used in each of the above categories is going to be given, along with each technique's advantages and drawbacks. Specifically, we are going to refer to the following methods:

1. Forms
2. Cookies
3. Behaviour Analysis Tools
4. User Classification
5. Collaborative Filtering
6. User History
7. Relevance Feedback

In the sections that follow, a concise presentation of each method is to be given.

2.3.1.1 Forms

The simplest approach used when creating a user profile is to provide the user with a form [17, 18] that contains a set of questions, which are targeted in order to

obtain the proper answers from the user, so as to understand the user's preferences and/or interests. The advantages of this method are its simplicity and the user's involvement. On the other hand, the obvious drawbacks are the fact that a form is restricted in length and in choices, while at the same time the actual correctness of the user profile created is based solely upon the form's creator, who needs to successfully select the proper questions that convey the necessary information. It is obvious that this method belongs to the explicit feedback category, and it is of course regarded as a static profile method. Due to the latter, it places the extra burden of keeping the profile up to date, entirely on the users, who need to renew their choices every time their preferences are altered [17, 19].

2.3.1.2 Cookies

Cookies are some of the oldest technological means used to maintain information about a user, and as such are readily employed when creating a user profile by a multitude of sites [19]. Cookies are practically files created by a web site and stored on the end user's computer through the web browser in use [20]. It is possible for the web site to retrieve such cookies and process the information they contain when a user visits the cookie's originating site after it has been stored. The information cookies usually store regards a session of the user in that specific web site, such as the user's time of arrival, any choices they made, search terms, settings, preferences or forms they may have completed. The advantages of this specific technology are the easy implementation and their transparent mode of operation, since they do not require any user interaction (such as completion of a form) and all the work is done in the background – cookies are therefore classified in the implicit feedback category, while also being able to provide a dynamic profile. As drawbacks, we have the fact that cookies are restricted to specific web pages and can therefore be used to provide a personalised experience for a single website (per cookie). Due to the fact that cookies tend to operate in the background, there have been major concerns regarding user security and privacy when cookies are in use. As such, the user should be aware of when a site uses cookies and should be able to choose whether to allow or not the use of such cookies. On

the other hand, most modern web browsers tend to block some cookies, so taking advantage of cookies to create a user profile is not always possible.

2.3.1.3 Behaviour Analysis Tools

These tools are based on the user's behaviour when they browse the Internet, since they collect information such as the hyper links the user clicked on, how much time they spent on each web page and even the location the hyper links had on each page. Through implicit feedback, a graph representing the user's motion through the Internet is constructed; as a consequence, the user profile originates from the way the user selects web pages, the way those web pages connect with each other and how much time is spent on each of them. Such tools provide a dynamic profile.

2.3.1.4 User Classification

This method, which belongs to the implicit feedback category, classifies users to various groups and is the method that has become quite popular in on-line stores [17]. According to this method, an up to date list of products that are of interest to the user is maintained (products that are of interest to the user are products that the user either bought or searched for) and according to that list, a user is classified to a group of users that are interested for similar products. By such an approach, general user profiles are created, in the form of groups. Personalisation based on this method has a more massive character, but tends to be more efficient, since it is capable of providing a personalised experience in shorter time spans, while the user is able to change their classification quite dynamically, by just browsing for different products; this is a clear indication of the dynamic character of the profile created by this method.

2.3.1.5 Collaborative Filtering

Collaborative filtering methods tend to harness the power of the community in order to provide a more personalised experience and especially recommendations:

they are based on an explicit feedback method, where users are able to rate content (whether it is web pages, products, et al) and that content is recommended to other users based on the rates it has collected. The whole idea of collaborative filtering is based on the fact that users with same (or at least quite similar) interests and/or preferences in a specific period of time tend to retain the same degree of similarity in the future. Collaborative filtering is mainly a social spin of the user classification method, since it offers the same kind of massive personalisation, while at the same time actively involving users. This has the major advantage of the users actively influencing the content they see, with the minimum effort required on their part. These methods provide dynamic profiles and tend to used in many cases.

2.3.1.6 User History

A user's history, in other words the web pages they visited sorted chronologically, is an important tool in the effort of trying to recognise their preferences and desires, with the ultimate goal of creating a representative user profile. As a tool it is available only on specific locations, such as the Internet Service Provider (ISP) server to which the user connects in order to gain access to the Internet, or the user's local computer, through the web browser they use. Taking advantage of such a tool is usually an effective solution to construct a truly representative user profile and can be perceived as the most powerful of the implicit feedback methods, that provide the most dynamic form of profile. Due to the inherent limited access though, various approaches have been adopted [19]: they either force the users to configure their web browser in order to channel traffic through a proxy server owned and controlled by them, in order to maintain each user's history, or they provide some kind of user agent software which operates on the user's computer in tandem with their web browser of choice, thus maintaining a private representation of the browsing history. The drawback of such approaches is that user consent is required, something quite difficult to obtain, since the user is basically asked to consent on their being monitored.

2.3.1.7 Relevance Feedback

Relevance feedback methods [17, 19] have been employed in many occasions, since they give the users a direct control of the recommendations they receive from the system, due to their explicit feedback nature. This method provides a dynamic profile and operates in the following manner: users is shown recommendations based on their current profile, and they are provided with an option to rate these recommendations in a binary way (such as ”like”/”dislike” the recommendation). In that way, the system adapts to their changing needs and updates their maintained profile, in order to better represent their choices.

2.3.2 User Profile Representations

Besides the methods mentioned above concerning the construction of a user profile, it is worth noting that there are several different forms a user profile can be represented by. According to [19], these forms are the following:

1. Vector based keyword profiles
2. Semantic network profiles
3. Concept-based profiles

In the first case, the most important terms from the user profile are extracted in the form of keywords, are weighted and then matched to a vector representation, which forms the profile. Popular variations of this form are the use of a single keyword, the use of a single vector for each interest area and finally the use of multiple vectors per interest area.

In the second case, keywords are extracted yet again, but instead of being placed in a vector, they are placed in a connected network of terms (where a term or a set of terms are placed in a node), based on their meaning. By taking advantage of semantic links, a broader profile can be constructed than the one based solely on keywords extracted so far.

Finally, in the last case, a predefined taxonomy of concepts is used, where terms that are going to be included in the user profile are modelled against. This

representation is quite similar to the semantic network representation, differing in the fact that the nodes represent abstract topics the user is interested in, instead of terms or collections of terms. The form that is going to be used in our case is the first, the vector based keyword profile.

2.4 Keyword Extraction for Advertising

Contextual advertising systems, such as Google's AdSense and Yahoo!'s Contextual Match, attempt to extract all the relevant keywords from a web page and then display advertisements based on these keywords. The proper selection of keywords is extremely important from the advertiser's aspect, since an improvement of 10% can lead up to an equal increase in Click Through Rates (CTRs) in displayed advertisements, subsequently leading to a potential increase of profit [6]. The approaches adopted during keyword extraction for advertising purposes, are the following:

- treat words present in web pages as independent terms
- combine words in text to form phrases
- inspect the location of a word in the text
- count in HTML tags and the words that appear in them

Keyword extraction is directly related to areas such as information extraction, the recognition of named entities and phrase annotation, since all of them attempt to find the most important phrases in documents. The approach followed in such areas is to divide phrases in their individual terms and their independent handling as discreet terms instead of the phrase as a whole. Similarly, this is performed in keyword extraction.

The algorithms that perform keyword extraction can be divided in two broad categories [16]:

1. Corpus dependent approaches

2. Corpus independent approaches

The first category demands a large number of documents to be present in the collection and predefined keywords as training parameters, in order to successfully construct a prediction model. For example, the TF-IDF [4] metric is commonly used to weight a term that is part of a collection of documents. Other common choices in literature are Mutual Information [21] and the log-likelihood measure [22].

On the other hand, approaches that do not make use of a document collection, tend to extract keywords from a single document with no previous or relevant information. For example, statistical analysis of a document can lead to keywords being extracted from it, using metrics such as word correlation based on their co-occurrence in a specific context [23].

The approach that yields the best performance is the one that makes use of a corpus. Nonetheless, the prediction model used in corpus dependent approaches, tends to be limited to specific fields, since the documents in a collection are usually related to a single or limited concepts. This has a significant impact on the quality of the keywords that will be extracted when a document does not belong to similar concepts with the ones present in the collection. Due to this drawback, many approaches that are corpus independent are blooming. Many known applications that offer keyword extraction are GenEx, KEA and the latter's variations.

In an advertising system, the presence of a good probability prediction is very important, besides the precision of the keyword extraction, since having good predictions can lead to potential increase of profits [6]. The most common metric used during the evaluation of the accuracy of probabilities is the entropy metric, where lesser values are matched to better probability estimates, with a value of zero being the optimal. The importance of a good probability prediction is evident by the bid phrases the advertisers pay for in their advertisements: if they are aware of the probability estimates of each keyword, they can bid quite different amounts to acquire them, since they can have a clearer image on the benefits each keyword can have on their advertisements and the probabilities it has to lead to a successful sale.

Chapter 3

Related Work

In this chapter, we are going to present some work originating from the commercial sector, where various businesses have attempted to provide personalised advertising to customers. Specifically, a short presentation of four different companies is going to be provided, with the approach each followed along with the problems they faced. The companies presented here, can be grouped in two categories, the first being the ones that used proprietary software to serve advertisements to the users (Claria and DoubleClick), and the second being those that collaborated with ISPs in order to provide personalisation to their subscribers (Phorm and NebuAd). The second category is the one that is of more significance for this thesis, yet the first is provided as well, in terms of completeness. Following that presentation, is a short collection of the reasons why each of the mentioned commercial products failed to deliver. In the section that follows, an overview of the academic research around the BM25 scoring algorithm is presented. BM25 is the basis of the scoring algorithm used in our system. Following is a short presentation of three different approaches on keyword extraction, from the academic domain.

3.1 Commercial Advertising Solutions

3.1.1 Claria

Claria Corporation (formerly known as Gator Corporation) was a company that was in the advertising network business and distributed adware products (under the Gator brand - also known as Gain AdServer due to their Gain advertising network). It was established in 1998 and shut down in 2008, after ten years of operations. The company's Gator software was distributed along with many free of charge programs for Microsoft's Windows operating system, and since installed, it tracked the user's browsing habits and displayed advertisements based on the user profile it generated and maintained. The software was clearly marked as spyware, since its behaviour and mode of operation was often unwanted by the end user and caused disruptive effects. Gator was the company's flagship software, but a range of other programs were released; yet all of them had an advertisement centred philosophy, since they all reported data concerning the user to the company's Gain advertisement network and displayed the corresponding advertisements even when these programs were operating in the background. As a side effect, the Gator software has caused some financial unrest to many content publishers whose sites were supported financially by the display of advertisements: the software replaced banner advertisements in web pages by advertisements served by the company's Gain network, thus depriving the content publishers of any possible revenue through the advertisements initially displayed on their site. As a natural consequence, such practice was a cause for legal action from major content publishers [24] against the company. The Gator software follows an approach to personalised advertising that is both disruptive against competitors and content publishers and of course breaches the user's privacy.

3.1.2 DoubleClick

DoubleClick was founded in 1996 and it was one of the first companies that operated in the online advertising sector. It was acquired by Google in 2008. Dou-

bleClick offers technology products and services that are sold primarily to advertising agencies and media companies to allow clients to traffic, target, deliver, and report on their interactive advertising campaigns. The company's main product line is known as DART, which is designed for advertisers and publishers, as it intends to increase the purchasing efficiency of the former and to minimise the unsold inventory of the latter.

DoubleClick targets advertisements based on various criteria. Targeting can be accomplished using Internet Protocol (IP) addresses, business rules set by the client or by referring to information (regarding users) stored within cookies on their machines. Some of the types of information collected regarding users are:

- Web browser
- Operating System
- ISP
- Bandwidth
- Time of day

According to [15], DoubleClick operates as a facilitator between the advertiser and the end user. DoubleClick's means of tracking user activity was attaching user browsing histories to anonymous user identifications. While a user's activity could be tracked, the actual identity of the user remained unknown. However, the ability to further refine data profiles was made possible through a series of acquisitions of other companies and their proprietary client databases (such as Abacus Direct in November 1999 [25]). DoubleClick then announced that it intended to match their anonymous user data with specific user names, personal information, and e-mail addresses from Abacus' database. However, the company decided against such a move after the public outcry that ensued (concerns were raised and many lawsuits were filed [25]) and opted to refrain the aforementioned move, until proper legislation and industry-wide standards concerning privacy were adopted.

3.1.2.1 Privacy concerns

In the context of electronic marketing the potential harm results from the fact that the organization developing user profiles can accumulate potentially sensitive information about a user, based on their Internet activities. A company that is able to electronically monitor an individual's computer use could potentially gain intimate knowledge of that individual's situation as a result of the Internet sites they visited. The organization developing the profile may intend to use such information solely for the purpose of advertising, however it is not difficult to see the potential harm to the individual's practical interests if such information came into the hands of a third party.

Reduced to its simplest form, electronic monitoring, as is currently practised, amounts to unauthorized observance. Many individuals using the Internet have no knowledge of when their online activity is being monitored for the purposes of developing an advertising profile. During electronic monitoring, the observed information is usually collected, organized, and distilled before another individual views it. It is even possible that another individual will never view such information and that any advertising that is tailored to an Internet user will be done completely by electronic means. The possibility of harm is minimized even further once Internet advertising companies such as DoubleClick make additional efforts to ensure user profiles remain anonymous.

DoubleClick responded to the privacy concerns in a variety of ways after aborting its plan. It also initiated a significant media campaign to explain how users could opt-out of their service. DoubleClick had an operable opt-out service for over three years preceding the aforementioned controversy but had not promoted it extensively. The opt-out mechanism required a user to visit a site and download a cookie, which served as notice to the company's system that it is not to download any further cookies on the user's computer. However, it is apparent that DoubleClick's opt-out practice is not the most appropriate solution, mainly due to the fact that users have extremely limited knowledge of it and, fundamentally, it places the burden of the decision on the user to take active steps to prevent electronic monitoring. The opt-out mechanism has also been criticised as misleading

and insufficiently effective. According to [26], opting-out disables cookie based tracking, yet IP address based tracking remains unaffected.

After the aforementioned privacy concerns were raised, DoubleClick had become directly involved in the development of the IAB's Privacy Guidelines. The Privacy Guidelines were intended to form the foundation of a self-regulatory regime with respect to personally identifiable information gathered by electronic means on the Internet.

3.1.3 Phorm

Phorm (formerly known as 121Media) is a United States based company that offers advertising software, and was founded in 2002. It initially distributed programs considered spyware, but after dealing with complaints from various groups in the United States and Canada, announced it was switching directions and would be targeting ISPs in the United Kingdom, in order to deliver targeted advertisements to their subscribers through use of their own advertising system, called Webwise which works in tandem with their advertising network, dubbed Open Internet Exchange (OIX). Webwise is a behavioural targeting service which employs Deep Packet Inspection (DPI) techniques to examine ongoing network traffic. According to Phorm, their system stores zero amount of personal data, therefore rendering the possibility of accidental or malicious disclosure impossible; also, they claim their system maintains user anonymity and prevents any form of user identification. It also offers users the possibility to opt-out of their system and also the ability to undo any previous participation of theirs [27]. Nevertheless, the system, and especially some of its ISP clients in the United Kingdom, have been the subject of much criticism, the former due to the fact that it operates on a grey area concerning user privacy, and the latter due to running secret trials of the service, without notifying their subscribers.

The company claims that the collected data are completely anonymous and that it cannot use such data to identify a user, and adds that its advertising categories exclude certain sensitive terms and have been developed in such a way so as not to reveal the identity of the user [28]. Users are given the choice to opt-

out of Phorm's service via a website operated by their ISP. However, according to a spokesman for Phorm, the way the opt-out works means the contents of the websites the user visits will still be mirrored to its system [29, 30].

The service works by classifying user interests and matching them with advertisers who offer relevant advertisements. Phorm's system, like many websites, uses HyperText Transfer Protocol (HTTP) cookies to store user settings. According to the company, an initial web request is redirected a few times (via the use of HTTP 307 responses) within their system, in order to determine if the user has opted-out. The system then assigns a unique ID for the user (or collects it from the cookie if it has already been set), and adds a cookie that is forged to appear to come from the requested website [31]. For a detailed description of the inner workings of the Webwise system, consult [31]. According to [32], Phorm's system stores a tracking cookie for each website visited on the user's PC, and each of those cookies contains an identical copy of the user's ID. Phorm's system supposedly strips its tracking cookies from HTTP requests before they are forwarded to the target website's server, but it cannot prevent that ID from being sent to websites when the HTTP Secure (HTTPS) protocol is in use. Subsequently, this could enable websites to associate that ID to any details collected on the website itself concerning the visitor.

Phorm has been the subject of many talks regarding the legal aspect of their system (one detailed analysis of legal concerns can be found in [33], which is based on [31]), since user privacy and the actual ability to opt-out of the system are questioned. In fact, many legal authorities from the United Kingdom, as well as from the European Union have ruled that the service could be deemed legal should it be offered as an opt-in to users [33, 34].

3.1.4 NebuAd

NebuAd was an online advertising company based in the United States, founded in 2006 and operational until 2009 [35]. It developed behavioural targeting advertising systems and its target audience were ISPs, whom provided with solutions to analyse their customers' browsing habits in order to serve targeted advertise-

ments. It falls in the same category as Phorm. At its peak, the company had signed deals with more than thirty ISPs and managed to cover approximately ten percent of all the broadband users in the United States [36, 37]. The company's solution was comprised of three parts: custom hardware hosted within the client ISP, tasked with the advertisements' injection into content pages, a separate server complex tasked with the analysis and classification of the ISP's subscribers' traffic and finally the intermediary role of contacting advertising networks to act as advertisement suppliers to NebuAd's system. A hardware device was installed inside the client ISP's network. Each device was able to monitor up to 50,000 users [37]. Users were given the choice to opt-out of NebuAd's system, yet there was no actual block between the transfer of data from the ISP to NebuAd, even if the user opted-out. According to Nebuad's sales, less than 1% of users opted-out. Since ISPs route all of their customers' traffic, they present the perfect vantage point from which to monitor all the traffic to and from a consumer using DPI. By such traffic analysis, as has already been mentioned, it is possible to gain more information about a customer's preferences than any other method. The company's privacy policy claimed that the collected data would not contain any sensitive information (medical, confidential, etc). NebuAd argued that behavioural targeting enriched the Internet on several fronts: first of all, website owners were offered better chances to earn revenue, by displaying more relevant advertisements on their sites. Secondly, advertisers were offered better targeted advertisements, hence reducing the need to publish as many advertisements as possible in the hope of attracting clients and finally, users were being offered more relevant advertisements. The company's clients, the ISPs, gained profits for allowing NebuAd access to their network on a per-user per-active profile basis. NebuAd's system utilised data such as search terms, page views, page and ad clicks, time spent on specific sites, zip code, browser info and connection speed to profile the users [38], yet it did not have access to user identification information from the ISP, although such a thing could have been viable through traffic monitoring.

The criticism concerning the NebuAd system, revolved around privacy concerns, the lack of user awareness and the weak opt-out mechanism. Privacy con-

cerns were evident, since a third party (NebuAd) had access to a user's online behaviour, without them being aware most of the time. As such, there was no assurance of the user's identity remaining safe or their private data not being used maliciously. User awareness was another major drawback to NebuAd's system, since neither the company, nor its client ISPs (at least most of them) informed the public of the actual use of the system. The supposed opt-out mechanism permitted the user to stop the monitoring of their traffic by NebuAd's system; nevertheless, despite the users opting-out, the monitoring process did not stop, instead the advertisements were just not being displayed [39].

3.2 Criticism of Commercial Solutions

The most prominent issue that plagued the aforementioned commercial attempts at serving personalised advertisements online, is that of user awareness and consent: the fact that the ISP is cooperating with a third party company and grants that company access to its web logs in order to monitor and track its users in a system not owned and operated by them, presents a major privacy and subsequently trust breach. Specifically, in order for the ISP to be legally covered, user consent should be attained; as such, the ISP should inform the users of such an activity taking place in the contract they are called to sign, and should at the same time approach content publishers and inform them. All of the implicated parties (users and publishers) should be aware that:

- User behaviour is monitored in the ISP's servers
- User activities are being processed by a third party company (in cases such as Phorm and NebuAd) for personalised advertising purposes

User and publisher consent is a demanding and difficult process, especially concerning the publishers' side. Indicatively, there could be a conflict of interests and competition, since for example a search engine operator offers advertisements and is thus not willing to allow a competitor a share of their revenues.

The second most important issue is that of the choice users have, in regards to the system's activation and operation: the third party company was usually offering an opt-out to users, which, as has already been mentioned, was most of the time rudimentary and not of actual significance; it was mostly provided as a means to let the users *think* they had a choice. Even if a user opted-out, there was no way to know if the system stopped monitoring their traffic and storing information regarding their profile. That was mostly because all the systems were supplied by a third party and were closed to inspection. As such, it was not in the ISP's control to check whether the user's choice was respected. The solution to that problem, as has been indicated, is to make such services available as opt-in for the users, instead of opt-out, in order for them to be able to control whether they want or not to be provided with personalised advertising, and thus be subjected to the third party's monitoring of their traffic.

Another important factor in the commercial failure of such services, was the anonymity of the users: the third party company that handled the ISP's web logs containing user traffic, along with the ISP, had to assure that users were kept anonymous and could not be identified by the logs and the profiles. The main problems concerning that are the following: the ISP needed to store cookies on the user's computer, in order to enable the personalised advertising system of the third party company to operate properly. In order to accomplish that, as has already been mentioned, various techniques that included cookie masquerading were used. Most of the time, the cookies were made to look like they originated from the web site requested by the user, in order for the user's browser to accept them. Further problems could be caused when the HTTPS protocol was used, since as we have already mentioned, these cookies had to be transmitted to the web site owner, who could then use the information contained in them to possibly identify the user and exploit sensitive information regarding them.

The final issue that plagued the aforementioned commercial attempts at personalised advertising, was that of the web logs and the fact that they were accessed by a third party company: the user had to be assured that such web logs were not stored for any further processing besides what was necessary to provide the rec-

ommended advertisements. Furthermore, these web logs were not to be passed to any extra third party, for any other use than the provision of personalised advertisements. Of course, such assurances could not be provided by the ISP, nor by the third party company, on a satisfying degree.

3.3 Academic Research

In this section, we are going to present the related work in two specific sections related to our work: firstly, the work concerning the scoring algorithm employed in the proposed system, the BM25 algorithm and one of its variants, dubbed BM25F. Secondly, a selection of techniques and systems concerning key word extraction.

3.3.1 BM25 & Variants

There is a multitude of scoring algorithms used in IR tasks, in order to properly select the most representative features from a collection of data. In this thesis, a specific scoring algorithm, the BM25 algorithm [40], is used as the basis of the scoring algorithm employed in the proposed system. As such, in this section we are going to present a selection of the academic research related to that scoring algorithm. Specifically, we are going to focus on the original BM25 algorithm, and one of its most widespread variants, the BM25F, which is the original BM25 algorithm with some extensions to support different weights for different fields in the text to be scored.

The BM25 scoring algorithm is based on a 2-Poisson model of term frequencies in documents. This is essentially a unigram language model, where the model parameter for a given term in any document depends on a binary hidden value referred to as ”eliteness”. Due to that binary value, each document’s terms are divided in two categories, the ”elite” and ”non-elite” terms: the ”elite” terms are those that represent the document and its subject in the most appropriate way, while the ”non-elite” are those that do not possess that characteristic. The main

formula describing the scoring algorithm for a single term is the following:

$$w_j(d,C) = \frac{(k_1+1)\times d_j}{k_1 \times \left((1-b)+b\times\frac{dl}{avdl}\right)+d_j} \times log\left(\frac{N-df_j+0.5}{df_j+0.5}\right)$$

where

- d = the current document
- C = the collection of documents
- k_1 = BM25 parameter
- d_j = the term frequency of term j
- b = BM25 parameter
- dl = the document length
- avdl = the average document length in the collection
- df_j = the document frequency of term j

In the formula above, we mention two parameters used: k_1 and b. The k_1 parameter is used to control the non-linear term frequency effect, while the b parameter is used to control the normalisation of the document length.

According to [41], the most common way structured documents used to be dealt with in relation to scoring their terms, was to employ a simple scoring algorithm, such as the BM25, to score each of the terms, then use the same algorithm to calculate a separate score for each of the terms appearing in a different section of the document: for example, in a simple scientific document, a simple structured format would be a title, an abstract, and the body of the document. Usually, most approaches would score each segment of the document separately, and then add all the individual scores for each term; in that way, the scoring algorithm employed was decoupled from the actual problem of combining different segments of the same document. In [41], it is clearly illustrated that such an approach is

deemed quite hazardous, since it can significantly alter the score of a term appearing in more than one of the aforementioned document segments. Due to that, a modification of the way BM25 is used is proposed, which instead suggests to alter each term's frequency, according to that term's occurrence in different segments (or fields) of the same document, then employing the regular BM25 formula. In other words, they suggest that instead of linearly combining the scores for each document segment, to linearly combine the respective term frequencies and then proceed to employ the regular scoring algorithm formula. This is the core concept used in the scoring method employed in the proposed system, where we modify (augment) the term frequency of each individual term, by boosting it depending on the different elements that contain it. The requirements set by [41], that need to be satisfied by the scoring algorithm when dealing with multiple different fields, are the following:

- To preserve the non-linear nature of term frequencies, something that improves performance of the retrieval process
- To revert to the unstructured document case, when the weight for each individual field is set to one
- To provide a simple interpretation in regards to statistics such as document length with the field weights included

According to [42], a variant of BM25, called BM25F is proposed, where weights and length normalisation parameters are distinct per field. What differentiates this approach from the one in [41], is that a different field normalising factor b is used for each different field of the text. This is accomplished by computing a field dependant normalised term frequency, according the the formula:

$$\bar{x}_{d,f,t} = \frac{x_{d,f,t}}{\left(1 + B_f \times \left(\frac{l_{d,f}}{t_f} - 1\right)\right)}$$

where

- $x_{d,f,t}$ = the term frequency of term t in the field f of document d

- $l_{d,f}$ = the length of field f in document d
- l_f = the average field length of field f
- b_f = field dependant parameter similar to parameter b of the original BM25 algorithm; if b_f is equal to zero, then no normalisation takes place, while if it is equal to one, then the frequency is completely normalised in regards to the field length

Besides that modification, a similar approach of linear combination of term frequencies, before the application of the standard BM25 algorithm is followed. The respective formula for calculating the score of a single term is:

$$\bar{x}_{d,t} = \sum_f W_f \times \bar{x}_{d,f,t}$$

where

- $\bar{x}_{d,t}$ = the BM25F score for term t in document d
- W_f = the weight for the specific field f
- $\bar{x}_{d,f,t}$ = the field dependant normalised term frequency, defined above

The above is employed in the formula below, to calculate the total score for a given document d:

$$BM25F(d) = \sum_{t \in q \cap d} \frac{\bar{x}_{d,t}}{K_1 + \bar{x}_{d,t}} \times w_t^{(1)}$$

where

- $w_t^{(1)}$ = the usual RSJ relevance weight for term t

The aforementioned functions require separate B_f and W_f parameters per field, and a single saturating parameter K_1.

When compared against the original BM25 algorithm and the approach presented in [41], BM25F is shown to perform better than both [42].

3.3.2 Keyword Extraction

Another research area that is of special interest in this thesis, is that regarding keyword extraction from web pages, used mainly in contextual advertising. Although an extensive research was conducted, a short selection is going to be presented in this section, in order to showcase some of the more distinguishing approaches.

The first, presented in [6], details a system that extracts keywords from various elements of a web page, that also takes advantage of search engine query logs and linguistic features of the text. The system comprises of four stages:

- Preprocessor
- Candidate Selector
- Classifier
- Postprocessor

The Preprocessor stage is responsible of transforming the HTML document into a plain text document, effectively removing all the included markup, while at the same time keeping all the necessary information included in some of the HTML tags. Specifically, the HTML tags that are of interest to this system, are the <a>, <title>and <meta >tags. It also employs two additional tools, a part of speech tagger and a chunker, where the former is used to extract linguistic information regarding the text, and the latter to detect the base noun phrases in each document. The system considers a word or consecutive words up to a length of five, as candidate keywords, including those words that appear in the title or meta sections. The approaches followed by the Candidate Selector stage are the following:

1. Monolithic, Separate (MoS): words or phrases that appear in different parts of the document are regarded as different candidates, regardless of their content.
2. Monolithic, Combined (MoC): words or phrases with identical content (ignoring case) are combined.

3. Decomposed, Separate (DeS): each phrase is decomposed in its individual words, and a label is assigned to each of these words.

The Classifier stage is the core of the system, where the classifier employed depends on the approach used in the Candidate Selector stage: a binary classifier is used in the MoS or MoC approaches, while a multi-class classifier is needed in the DeS approach. The features this system takes into consideration, are the following:

- Linguistic features, regarding the part of speech tags assigned to each word
- Capitalisation, which indicates a different word to the system
- Hypertext links
- Meta section
- Title
- URL
- IR related features, such as Term Frequency (TF) and Document Frequency (DF)
- Location of the word
- Sentence and document lengths
- Candidate phrase length
- Search query log

Finally, the Postprocessor stage, is responsible for generating the list of keywords ranked by the probabilities.

The second system, described in [12], follows a quite different approach, by expanding a web page with additional terms, in order to facilitate a better match between available advertisements and the web page itself. This is motivated by the writers' observation that on frequent occasions, a mismatch exists between

the vocabularies used in advertisements and web pages; that mismatch is referred to as the vocabulary impedance problem, and the approach proposed to resolve it in [12] is called impedance coupling. The concept behind the motive of this work, is that a web page usually belongs to a broader contextual scope than the scope advertisements belong to. This can happen because a web page can refer to any number of subjects, while an advertisement tends to be more specific and concise; advertisements can be considered as topic specific. Additionally, there is a frequent case where the association between an advertisement and a web page might depend on a topic that is not mentioned in the web page at all. This is related to the fact that most advertisers place a small number of advertisements, thus the terms used tend to be of a more general nature. Due to the aforementioned issues, web pages' and advertisements' vocabularies tend to have low intersection, even in cases where an advertisement is related to a web page. The driving force between the system's design is the knowledge that the addition of keywords to advertisements can lead to improved results: as such, the concept is to apply the same approach for the web page, by adding some extra terms. The system uses many approaches employing impedance coupling, such as:

- matching the advertisements and their respective keywords to the set of expansion terms
- matching the advertisements and their respective keywords to the expanded web page (original web page including the set of expansion terms)
- matching the web page with the target web page of the advertisement
- matching advertisements and their respective keywords, considering the target page of the advertisement
- matching the advertisements and their respective keywords with the expanded web page, considering the target page of the advertisement

Finally, as presented in [43], a simple approach can be used to effectively get decent classification results for a web page, by employing just its URL. The approach used in [43], is two step, where the URL is initially segmented into tokens,

which are subsequently analysed as features for classification of the web page. A recursive, entropy reduction based technique is employed to derive tokens from the URL during the first step. The second step, concerning the analysis of features for classification, makes use of the following features contained in each URL:

- URL components and length: a token that occurs in different parts of the URL contributes differently to the classification procedure, while the absence altogether of a component also affects classification. The URL's length can also be an influencing factor to the classification of the web page.
- Orthographic: a token's surface form can also be used for classification purposes, where terms consisting of digits only, or different capitalisation, can be generalised and distinguished by their length
- Sequential n-grams and Precedence: the sequence in which tokens are present, also affects classification

The evaluation present in this work, shows quite promising results originating from the employment of the URL for classifying a web page, at a minimum processing cost, since no actual fetching and analysing of the web page's content is necessary.

Chapter 4

System Architecture

The proposed system architecture is going to be described in detail in this chapter. First we begin by providing a concise overview of the system's layout and its components, followed by a description of the path of execution during all the stages of the system's operation. Afterwards, a more detailed view is given in regards to each of the components design and implementation. Wherever deemed necessary, we justify the choices made during the course of the system's design and implementation, by describing any difficulties, problems or alternatives. Finally, we conclude this chapter by providing the reasons why the proposed system counters all the problems mentioned in the related work chapter, its major advantages and innovative features, any difficulties and problems that were dealt with during the course of development and finally, some of the alternative approaches that were tried and tested before settling on the proposed design.

The complete system architecture is comprised of two separate parts: the client and the server side. The client side is responsible of monitoring the user's browsing activities, extracting the most definitive keywords found on each web page, maintaining a profile describing the user and their interests and transmitting these two pieces of information to the server side, in order to receive and display the most relevant advertisements in respect to the user's current focus (the web page they are viewing) and their most recent fields of interest (defined by their profile). On the server side, the web page and user profile vectors are received, a tri-part

match is performed against the advertisements currently available on the server, and those that are deemed most relevant are selected and transmitted back to the client. This thesis is focused on the client side. As can be derived from the system's description above, a thick client approach was followed, which offers the following advantages over the thin client approach: a decentralised mode of operation is adopted, since most of the necessary actions for the system's operation (monitoring of user traffic, construction of user profile and web page vectors) are executed on the client side, thus allowing for a more streamlined operation on the server side. The server side is thus no longer a possible single point of failure, and is solely responsible for the advertisement matching and serving. Another reason why such an approach offers additional benefits over the thin client, is that user profiling and monitoring, if selected to be performed on the server side, would require a much larger amount of system resources, since monitoring all traffic passing from the server would need to be identified and matched to each of the active users; such an option would of course lead to higher loads on the server, since it would need to be able to handle multiple concurrent user requests, while at the same time performing all the aforementioned functionalities. Therefore, by opting for a thick client approach, each client system is responsible for monitoring and profiling their user, thus requiring much fewer resources.

The client side system operates on a basic home router and as such, needs to be characterised by the following attributes:

- speed
- efficiency (mainly in terms of RAM)
- transparent / unobtrusive operation

4.1 Hardware

Specifically, the router on which we developed and tested our system, is the Linksys WRT160N [44] (version 3.0). The router's hardware specifications are

provided, in order for the reader to be able to comprehend the limitations set by the hardware itself, in regards to the aforementioned desired attributes:

- Chipset: Broadcom BCM4716
- RAM: 32 MB
- FLASH: 4 MB
- Wireless Standards: Draft 802.11n, 802.11g, 802.11b, 802.3, 802.3u
- Number of Antennas: 2
- Security Features: WEP, WPA, WPA2
- Security Key Bits: 128-Bit, 256-Bit

It is worth noting that the specifications concerning the chipset, RAM and FLASH components, are provided via the DD-WRT router page, accesible from the Router Database link on [45].

4.2 Firmware

In order for our system to be able to operate on a router, there is a need to be able to install and execute the developed software on it; such capabilities are of course not provided by default in current commercially available routers. Consequently, the router's shipping firmware was replaced by a firmware that provides the necessary functionality. Many such options are available, with the most prominent being those of OpenWrt [46] and DD-WRT [45]. Both of these options are open source, are based on the Linux kernel and as such provide a familiar development platform with the added bonus of a wide range of available libraries and software for development. They are regarded as embedded Linux distributions. They also enjoy the support of a large community and have extensive documentation available. The former started as an open source evolution of the firmware originally

developed by Linksys for their WRT54G wireless router (which was released under the GPL license and as such third party customisation was legitimately permitted). The latter offers commercial support, while free community supported versions are available for download and installation on supported routers.

The firmware of choice for the designed system is that of DD-WRT, since it provides support for the Linksys WRT160N router that was in our disposal. In order for the reader to be aware of the range of features provided by the DD-WRT firmware, its most prominent features, along with a short description, are the following [47]:

- more than 200 different devices are supported
- comprehensive functionality
- all current WLAN standards (802.11a/b/g/n) are supported
- enhanced frequencies are supported
- VPN integration
- various Hotspot systems are supported
- bandwidth management
- multilingual user interface

DD-WRT is suitable for a great variety of WLAN routers and embedded systems. Its main emphasis lies on providing the easiest possible handling while at the same time supporting a great number of functionalities within the framework of the respective hardware platform used. Its graphical user interface is logically structured, and it is easily operated via a standard Web browser, so even non-technical users are able to configure the system in only a few simple steps. Apart from the simple handling, speed and stability are also essential. DD-WRT allows a reliable operation with a clearly larger functionality than that provided by factory default firmware, which also fulfils the demands of professional deployment. Being supported by a large user community, users and developers are able

to acquire the necessary help. Thanks to this, potential flaws in the system are detected quickly and as such corrected in the most timely manner. DD-WRT users can find help and suggestions from other users in the project's forums, while the community maintained Wiki contains further information and how-to guides.

4.3 Software

In the following sections we are going to present the software parts of the proposed architecture, starting with the software necessary for traffic monitoring, then moving on to the implemented software modules that comprise our complete architecture.

4.3.1 Traffic Monitoring

Having established the main hardware and software base of the client side system, research was invested in the examination of the possible alternatives concerning traffic monitoring on the router. The primary consideration was that of speed and efficiency, and as such the suitable approaches were narrowed to two:

1. use of traffic monitoring and analysis software
2. use of a caching proxy server

The first option involved the use of a packet analyser, with the most obvious choice being that of Wireshark [48]. Wireshark is an open source cross-platform tool, with its most prominent features being the following:

- Deep inspection of hundreds of protocols
- Live capture of traffic and offline analysis
- Standard three-pane packet browser (in GUI mode)
- Captured network data can be browsed via a GUI, or via a terminal utility (TShark)

- The most powerful display filters in the industry
- Rich VoIP analysis
- Read/write many different capture file formats
- Capture files compressed with gzip can be decompressed on the fly
- Live data can be read from a wide range of network standards (Ethernet, IEEE802.11, et al)
- Decryption support for many protocols, including IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP, and WPA/WPA2
- Colouring rules can be applied to the packet list for quick, intuitive analysis
- Output can be exported to XML, PostScript®, CSV, or plain text

Wireshark makes use of the libpcap library [49] to capture network traffic and offers great interoperability with the tcpdump command-line packet analyser [49].

The alternative option in traffic monitoring was the deployment of a caching proxy server on the router. The most prominent option in that approach is that of the Squid proxy server [50]. Squid is an open source caching proxy server, that can be deployed in a variety of uses ranging from speeding up the content delivery of a web server, to caching data between people sharing network resources. To describe how the Squid system (as well as caching proxies in general) operate, an example might be more appropriate: a caching proxy (in our case Squid), acts as an intermediary between two systems, one being the client requesting some data and the other the server which provides the response. By monitoring the request and caching the response, a caching proxy can speed up the communication between client and server, since a request concerning a web page, can be the same for a specific amount of time; when that is the case, the caching proxy can provide the client with the response directly from its cache, thus serving a twofold purpose: the server is spared extraneous traffic, therefore has less load to handle, while the client receives a faster response to their request, thus enjoying better performance. According to a quote from Wikimedia Deployment Information found

in [50], ”(The Squid systems) are currently running at a hit-rate of approximately 75%, effectively quadrupling the capacity of the Apache servers behind them. This is particularly noticeable when a large surge of traffic arrives directed to a particular page via a web link from another site, as the caching efficiency for that page will be nearly 100%”. Most of the time, Squid offers a noticeable improvement in performance and its advantages can be enjoyed by many parties: from the end users, to the ISPs, the content providers or even the Content Distribution Networks (CDNs). It is worth noting, that the Squid caching proxy is available for all Linux systems and is packaged properly for the DD-WRT firmware distribution.

Having presented both of the approaches available, a short comparison is to be given, in order to justify the choice made during the system's design. The former route, that of employing Wireshark in order to capture traffic moving through the router, was considered at first as the prime means of monitoring traffic, yet upon further research, was deemed too cumbersome as for the speed and performance requirements of the system: all traffic would need to be monitored, an analysis of each passing packet would need to be performed and in order to properly gain access to an HTML page, we would need to maintain a constant status of all ongoing TCP connections in order to be able to reconstruct the HTML response page, since Wireshark monitors traffic on a packet level. Therefore, packets would need to be maintained in memory and then reconstructed by being put together in the correct order by using their corresponding TCP sequence numbers. On the other hand, by using a caching proxy server, we had a twofold advantage: first, the traffic monitoring occurs automatically for the type of traffic we are interested in (that is HTTP, since we only need to analyse the web pages the user visits), and the resulting responses are stored in the proxy's cache, thus being immediately available for the system to process, with no need for TCP connection status maintenance and analysis. Secondly, the use of a caching proxy provides an increase in the overall system speed, since it can offer the advantage of speedier responses to the user's requests, as has already been mentioned.

4.3.2 Programming Language

It is a fine time to provide an insight into the programming language that we chose to develop the system on, and the reason why we chose to do so. The language of choice is Python [51], which is an open source, dynamic language, being used in a variety of application domains. Some of its most prominent features are:

- very clear, readable syntax
- strong introspection capabilities
- intuitive object orientation
- natural expression of procedural code
- full modularity, supporting hierarchical packages
- exception-based error handling
- very high level dynamic data types
- extensive standard libraries and third party modules for virtually every task
- extensions and modules easily written in C, C++ (or Java for Jython, or .NET languages for IronPython)
- embeddable within applications as a scripting interface

The main reasons why we chose Python over an implementation in a more traditional language such as C or C++, are the following:

- it has a very capable and fully featured standard library, which means we have access to most of the functionality needed in our design, from the language itself, with no need to load external libraries which, considering our development platform (the DD-WRT firmware) may not be available or easy to compile
- Python is included in all major Linux distributions, subsequently it is readily available on DD-WRT

- it strikes a perfect balance between performance and inherent capabilities: it may not be as fast as C or C++, but it has more than the best attributes of high level languages such as Java, with the added bonus of improved performance: it is noticeably faster than Java, and requires much less in terms of resources
- resource friendly: it has a very efficient garbage collection mechanism, that automates memory management and frees the programmer from such tedious tasks

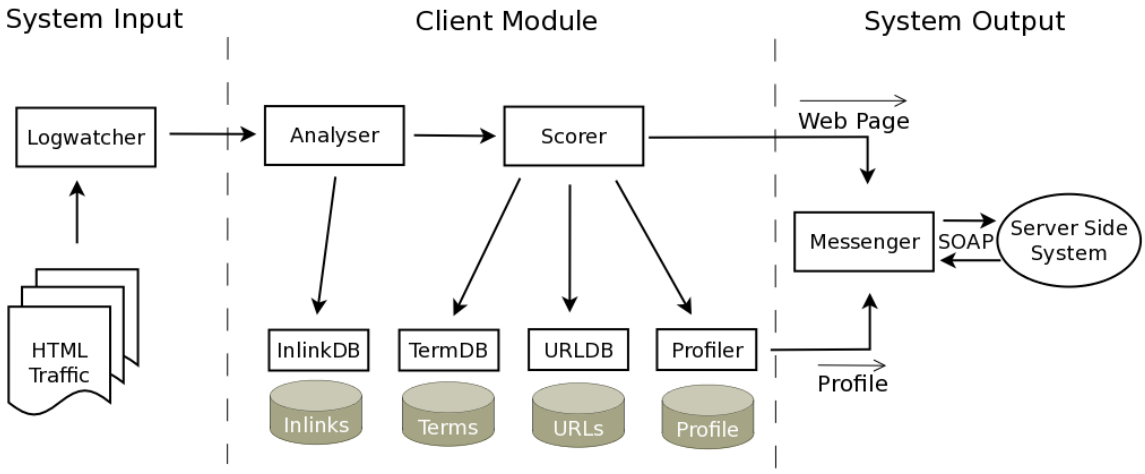
4.4 System Modules

Having laid the foundations of the design, we are going to proceed with the description of all the separate modules that execute the core logic of the system. In short, the modules are the following:

- Logwatcher
- Client
  - Analyser
  - Scorer
  - Database Modules
    - InlinkDB
    - URLDB
    - TermDB
    - Profiler
- Messenger

To provide a brief overview of how the modules interoperate, and the overall flow of execution, let us describe the system in operation, describing in more detail what figure 4.1 illustrates in a concise manner: the first module, the Logwatcher,

Figure 4.1: System modules & their interoperability



monitors Squid's logs in order to detect when a web page has been requested by the user. When such a page is found, its location in Squid's cache is fetched, the corresponding file path is retrieved and the second module, the Analyser is passed that file path. The HTML file is then parsed, its targeted features are extracted and then execution moves on to the Scorer module, which is tasked with the ranking of the terms found in the page, according to their frequency, their appearance in certain HTML tags and of course their BM25 score, based on the corpus of terms that is kept. Various database modules are used in the process, such as the Profiler, the TermDB (handles the terms found in all the documents), the InlinksDB (handles the URLs along with their anchor texts) and the URLDB (handles all the URLs visited by the user). The Profiler and Scorer modules, produce two vectors, the User Profile and Web Page vectors respectively, which are then passed on to the Messenger module, which is tasked with the transmission of these vectors to the server side of the architecture. The server side responds with the most relevant advertisements which are received and displayed by the Messenger module.

Now that the reader is familiar with the mode of operation of the system, let us delve in more details about each of the aforementioned modules.

4.4.1 Logwatcher Module

The first module in our design is the Logwatcher module, which is tasked with monitoring Squid's logs and locating the requested web page in Squid's cache. The Squid caching proxy operates all the time, as long as the router is powered on and the network connection is established. It is set up in transparent proxy mode, thus requiring no further configuration from the user, and constantly logging all traffic going through the router. In order for someone to be able to use Squid to build software that works with it, Squid has standardised methods of providing log files that offer a third party the functionality of monitoring, in real time, the network traffic as it is handled by Squid. By monitoring these logs, we are able to determine which object the user requested (using its Unique Resource Location (URL)), the type of that object (usually in the form of MIME types such as "text/html"), and whether that object was found in the proxy's cache or not. Squid has many logs available, but the ones used by our system, are the following two:

1. access.log: this log maintains information about when a request was monitored by Squid, and its resulting response from the server. To be precise, the standard format of this log usually has ten space separated columns [52], which server the following purposes:

    - the first depicts the Unix timestamp, the time of the request
    - the second depicts the time the cache was busy handling the request, in milliseconds
    - the third column notes the requesting client's IP address
    - the fourth contains the result codes; this column is made up of two elements, separated by a slash: the first element refers to Squid's result code (see [53] for all Squid result codes) while the second contains the respective HTTP status code (a complete list of all HTTP status codes is also available in [53])
    - the fifth column lists the number of bytes delivered to the client

- the sixth column depicts the request method employed (such as GET, POST, et al - see [53])
- the seventh column has the requested URL
- the eighth column contains ident lookups for the requesting client; however, in all default Squid configurations ident lookups are turned off. As such, this column contains a '-' character instead
- the ninth column has a hierarchy code (see [52] for further details)
- the tenth and final column has the content type, as is contained in the HTTP reply header - the MIME type that we described above

2. store.log: This log is used to keep track of all the transactions that occur on the cache. When an object is stored in the cache or removed from the cache, it is logged here. This log's main purpose and the reason we make use of it in our system, is that it keeps a record concerning each web page requested and its subsequent location in the proxy's cache. This log usually consists of thirteen columns, each with their respective interpretation:

   - the first is once again the Unix timestamp
   - the second is the action the object in the cache was submitted to (see [53] for details)
   - the third is the directory number of the cache the object is located at
   - the fourth is the file number of the object in the cache
   - the fifth contains the hash value used to index the object in the cache
   - the sixth has the HTTP status code
   - the seventh the value of the HTTP Date reply header
   - the eighth the value of the HTTP Last Modified header
   - the ninth the value of the HTTP Expires header
   - the tenth the value of the HTTP Content Type header
   - the eleventh has two slash separated values: the first the value found in the HTTP Content Length header and the second the actual size read

- the twelfth contains the request method
- the thirteenth and final the key to the object, most common being the URL

In order to illustrate how the access.log stores its entries, let us present a sample line and decipher each column:

1291736793.799 319 127.0.0.1 TCP_MISS/200 637 GET http://www.google.com/url? - DIRECT/209.85.227.99 text/html

On the above line, the important parts are the URL (column sever), the content type (column ten) and the result codes (column four).
Another sample line from the store.log output, in order to help the reader understand what information is important to our system:

1294831631.679 SWAPOUT 00 0000004B DD34FF96299B17CD74B0C60E85046E24 200 1294831631 -1 1294835231 text/html -1/15659 GET http://wiki.squid-cache.org/

From the above line, the important information is the URL (column thirteen), the content type (column ten), the cache action (column two), the directory and file numbers (columns three and four, respectively). The possible cache actions, according to [53], are the following four:

- CREATE: seems to be unused
- SWAPOUT: the object was saved to the disk cache
- SWAPIN: the object was retrieved from the disk cache into memory
- RELEASE: the object was removed from cache

The cache actions that is of interest to the system is SWAPOUT, since it means that the object can be processed using the cache. Squid's cache is organised in a folder hierarchy, and as such, the directory number (column three) indicates the top level directory of the cache the object is stored into. Afterwards, according to the way Squid's cache has been configured, the file number can be used to find the sub directory the file is located in and access it. It is worth mentioning that a frequently occurring combination of directory and file numbers are -1 and FFFFFFFF, which mean that an object exists only in memory and not on the disk. To put matters

into perspective, in the above line, an HTML page (text/html) originating from http://wiki.squid-cache.org/ was saved to the disk cache (SWAPOUT), specifically in the top level directory 00, with a file name equal to 0000004B.

Having mentioned all the necessary details about Squid and its logging functionalities, let us resume with the functionality of the Logwatcher module: by constantly monitoring Squid's access.log, Logwatcher examines each new entry in that log, searching for "text/html" content types - since we focus entirely on HTML pages for keyword extraction, where the result code is equal to 200 or 304 (HTTP OK and HTTP Not Modified, respectively). If it encounters such a combination on the same line, it stores the URL, and then examines the last few lines in store.log, searching for a matching URL and content type (access.log and store.log are updated in parallel by Squid, consequently, when a new entry is appended in access.log, a respective entry is appended in store.log); when matches are detected, a further inspection of the cache action column is instigated, in order to check whether the requested object is available in the disk cache (some objects forbid caching by including headers that render them expired in very short time spans). If it is, then the directory and file numbers are used to detect the object, which is retrieved, opened and stripped of Squid's additional header lines (Squid stores some additional header description lines in each cached object, which are not of any value to our system) before passed on to the next module, the Client module.

4.4.2 Client Module

The Client module is a simple top level module, that encompasses the functionality of all the smaller modules such as the Analyser, the Scorer, and the various database modules. It aims to unify the data input from the Logwatcher module to the Analyser module, the data output from the Scorer module towards the Messenger module and most importantly, the link between all the data exchanged between the modules themselves: the information extracted by the HTML document by the Analyser module, is used by the Scorer, InlinkDB and ProfileDB modules. It has no added functionality of its own, yet serves the purpose of simplifying and

clearing the system's overall path of execution.

4.4.3 Analyser Module

The Analyser module is where each HTML file is parsed, in order to extract the terms included in the most important tags and then remove all HTML markup afterwards, in order for execution to proceed. Let us delve into the specifics: each HTML object found in Squid's cache by the Logwatcher module, is passed on this module, which opens the file, reads its contents, transforms the whole text in lower case characters, removes all the HTML entities from it (HTML entities are specific tokens in the HTML language, which are needed to represent special characters and symbols: as an example, the '&' character is represented by the HTML entity '&'; for a full list of all HTML entities, see [54]), along with all the CSS and JavaScript code (contained in <style>and <script>tags, respectively). As a module, it also maintains a list of stopwords, that it loads from a provided text file on run time - thus providing the ability to update these stopwords without the need to stop the execution of the program - and initialising a stemmer object, that is used in order to stem the terms found in the text. The stemmer algorithm that has been selected for use in our system, is the Porter algorithm, with the official implementation in Python retrieved from [55]. The module provides specific functions that parse the HTML code and return lists of the stemmed terms found in specific tags, specifically the following:

- parse_url: returns a list of all the terms contained in the web page's URL, excluding the protocol part ('http://', 'ftp://', et al) and the 'www.' part, if present
- get_title_tag: returns a list of the terms present in the document's <title>tag
- get_meta_tags: returns a list of the terms found in the "content" attribute of the <meta>tag whose "name" attribute contains the term "keyword". Since <meta>tags can be used in a variety of ways, we are interested only in those who are used to help search engines, by providing specific keywords that

are deemed descriptive of the page by its publisher. An additional attribute is examined, the ”lang” attribute, which is present in multilingual pages, where keywords can be provided for different languages. If it is encountered, we only take into account keywords that are in a <meta>tag whose ”lang” attribute contains the language ”en” (for English); if not present, we assume that the keywords are in English and proceed to extract them. Keywords are unfortunately non standardised concerning their syntax, and as such we follow the most common approach, which has keywords written separated by commas [54].

- get_a_tags: parses the HTML document and returns a list of all the terms contained in the <a>tag as anchor text, along with their respective URL found in the tag's ”href” attribute, if present. It treats anchors inside pages as links to the page, and as such removes the suffix of the URL that begins with the '#' character (which indicates a named anchor in the referring URL). To illustrate what this function returns, let us provide a short example: supposing the page contained two links to ”http://www.sample.url/index.html” and ”http://www.sample.url/index.html#named_anchor” with anchor texts such as ”search here” and ”find out here”, the function would return ”http://www.sample.url/index.html” : 'search', 'here', 'find', 'out', 'here'.

- get_h_tags: returns a list of all the terms found in <h[1-6]>tags, grouped in the respective heading class. <h>tags are used in HTML to display headings, and have six different types, ranging in size: <h1>is the largest and scaling to the smaller in <h6>. Although headings are rated the same later during the scoring process, the fact that the function returns all the terms classified allows for easy modifications of the weighting scheme in future iterations of the system.

- get_img_tags: according to the HTML specification [54], every <img>tag must contain an ”alt” attribute, that is a surrogate text to be displayed should the image not be displayed. Unfortunately, most of the time, ”alt” attributes, while included, tend to be left empty; nevertheless, this function returns all

the terms found in this attribute, which are weighted in a distinctive manner by the Scorer module later on.

- get_bui_tags: this function returns a list of all the terms enclosed in the HTML <b>, <u>and <i>tags, which represent bold, underlined and italicised text respectively. These tags are still used, despite of the high rise of the employment of CSS for styling purposes.

- write_clean_html: this function is called after all the aforementioned functions, since it strips the HTML markup from the document, removes all the stop words and those terms that are made up entirely of characters or numbers, since we treat those as insignificant. It stores the plain text of the HTML page in a temporary file, which is used later during the process of updating the inverse index of the terms encountered so far.

It is worth noting in this point, that the approach used during the extraction of all the aforementioned tags from each given HTML document, was that of regular expressions instead of a full blown HTML parser (the most feature complete in Python is Beautiful Soup [56]). The rationale behind such a choice is simple: Python started out as a scripting language, before evolving into so much more; as such, it is extremely efficient in scripting tasks, such as parsing large quantities of text - in our case HTML markup code - by means of regular expressions. A full blown HTML parser would have been the easy choice in terms of development time, since we could have just opted to use the already working code base of other projects such as the one mentioned earlier. Nevertheless, since one of the main requirements of our system is speed and efficiency, the aforementioned choice was easy to rule out: Beautiful Soup's HTML parser is a complete solution, yet it is quite cumbersome in terms of performance for our needs. Consequently, we opted to develop our Analyser module using regular expressions, which cost very little in terms of resources: in Python, a regular expression is represented by an object that needs a string representation of the regular expression during its compilation. That is all the cost regular expressions have: once compiled, they can be used quite efficiently as many times as needed, with no additional cost. In order to clar-

ify the way regular expressions have been utilised in this module, let us present a small example: an <img>tag, according to the HTML specification by the W3C [54], must be written in the following manner, in order to comply with the standard: <img src="http://sample.url.of.image/" alt="alternative text for sample image">As such, in order to capture all images using regular expressions, we will use the following: "<img .*?>" The above simple line, translates to the following for the regular expression parser in Python: fetch me all those strings in the given text, that begin with the sequence of characters "<img " (space included) and then all intermediary characters (including whitespace, symbols, etc), regardless of their number, up until the first ">" character you encounter. All the intermediary characters, regardless of their number, are represented by the ".*" part of the regular expression. The "?" character, signals that the "*" modifier should not be greedy: if we omitted the "?" character, then the regular expression would fetch us all the characters it found after the "<img " sequence up until the last ">" character of the text, something that of course would not be correct. In a respective manner, we have constructed regular expressions for all the information we want to extract from each HTML document. It is worth noting that Python's regular expressions follow the syntax defined in another scripting language, Perl [57].

4.4.4 Scorer Module

The Scorer module is the heart of the system: it is where each web page's terms are scored, based on two different approaches; first, based on their frequency of occurrence in the plain text, along with their occurrence in any of the aforementioned HTML tags extracted by the Analyser module and second, based on the score they receive using a modified version of the BM25 scoring algorithm. The functions it provides are the following:

- plain_text_frequency: when called, this function processes the output file of the Analyser module's last function, write_clean_html, in order to construct a dictionary representation of the document, where a term is matched with its respective frequency in the plain text. It is worth noting that this function

ignores terms with a length of one character, since single character words usually convey no special value. As an example, let us suppose the following plain text: ”the quick brown fox jumped over the slow brown cat”. Given that text, the function would return the following: [”the” : 2, ”quick” : 1, ”brown” : 2, ”fox” : 1, ”jumped” : 1, ”over” : 1, ”slow” : 1, ”cat” : 1]. The function returns the total number of words found in the plain text (excluding as we already mentioned single character words). Following on the example above, the function would return the number ten.

- get_top_k_terms: this function receives all the extracted information from the Analyser module, and uses them along with the dictionary produced by the plain_text_frequency mentioned above. It also receives a number, k, which indicates the number of terms it is expected to return. Upon initialisation, this function calculates the total number of terms found in each of the lists received from the Analyser module (e.g. the number of terms found in the title, in headings, in links, etc), in order to use these numbers later on, when altering a term's frequency based on where it appears in the HTML tags extracted. By supplying the current URL to the InlinkDB module, a search is performed in the database file that holds all the URLs encountered thus far in HTML documents as links, in order to check if the current URL has been mentioned before and if so, the terms used as anchor text for that URL are returned. The URL is also added to the history of URLs visited so far that is maintained by the URLDB module, along with its word count (as returned by plain_text_frequency). Afterwards, each and every term found in the term dictionary, is added to the inverse index of terms maintained by the TermDB module and then is examined in the tag lists returned by the Analyser module, in order to determine whether it appeared in any of the tags of interest. If it has, a bonus to its frequency is calculated, using the following simple formula:

$$wtf(t) = ptf(t) + \sum \left(\frac{tf(t) * tw}{|t|} \right)$$

where

- wtf(t) = weighted term frequency of term t
- ptf(t) = plain text frequency of term t
- tf(t) = tag frequency of term t
- tw = weight of tag
- $|t|$ = number of terms in tag

By applying this formula, a term that appears in HTML tags, gains a greater frequency than another that occurs only in plain text. The weights that have been assigned to each element are presented in table 4.1.

Table 4.1: Elements and their assigned weights

Element	**Assigned Weight**
<title>	50
Inlinks	50
<meta>	40
<h[1-6]>	25
<a>	20
<[b,u,i]>	15
<img>	10
URL	10

The rationale behind the assigned weights is that the <title>tag is what the web page's publisher thinks describes best their web page, and subsequently receives the greatest weight. <meta>keywords are also the way the publisher describes their page in short for indexing purposes by automated search engines, and as such can be deemed of great importance as well. Inlinks, on the other hand, are what others describe the page as, which is equally important to its own title. Inlinks describe the terms what have been used in other pages as anchor text leading to the current URL: for example, a link to "http://www.google.com/" can usually be attributed to

an anchor text such as ”search engine”. This means that when processing ”http://www.google.com/”, we shall take into consideration the fact that the terms ”search engine” have been used to describe this link. Heading tags are also quite important, since usually they are used to signal something that is descriptive of the text following it, in other words sectioning text. Anchor text is also important, since it contains the terms that link to other pages and usually contain special meaning to the current page: for example, if a text refers to sorting algorithms and has a link to another page with anchor text such as ”merge sort”, it means that it links to something with a common topic, and is thus of greater importance than the respective plain text. Text style tags such as <b>, <u>and <i>are of lesser importance, due to their less frequent usage. Nevertheless, they also deserve special handling, since they change the appearance of the text they are applied to, thus signalling something distinctive. Finally, the <img>tag's ”alt” attribute and the URL terms are assigned the smallest weight, the former due to the reason we have already mentioned before (infrequent use) and the latter due to not always containing easily discernible terms also present in the plain text. Concerning the ”alt” attributes, they are assigned a different weight than plain text, since if employed, provide a surrogate text to be used instead of the image itself: this means these terms should in a way convey the same meaning with the image to the reader of the document and as is common, ”a picture is worth a thousand words” (ten in our case).

Having completed the reference to the way the term weighting scheme operates, let us resume with the presentation of the get_top_k_terms functionality: after assigning each term with an appropriate bonus using the aforementioned formula, each term has a score calculated for it. After matching the term with their respective score, two lists are sorted and returned by this function: the first containing the top k terms in respect to their frequency (augmented with the bonus they received) and the second in respect to the score the terms have been assigned. The latter is called the web page vector, and consists one of the two vectors that need to be transmitted to the

server in order to receive the recommended advertisements. -score_term: this function calculates the score for each term, using the BM25 algorithm. The formula used specifically, is:

$$score(t) = idf(t) \times \frac{tf(t) \times (k+1)}{tf(t) + k \times \left((1-b) + b \times \frac{|D|}{avg\_words} \right)}$$

where

- idf(t) = the Inverse Document Frequency (IDF) of term t (see below for its respective formula)
- tf(t) = augmented term frequency of term t
- k = a constant, usually set to 2.0
- b = a constant, usually set to 0.75
- $|D|$ = number of words in current document
- avg_words = average number of words in the corpus

The respective formula for the IDF of a term is as follows:

$$idf(t) = log \left(\frac{|N| - df(t) + 0.5}{df(t) + 0.5} \right)$$

where

- $|N|$ = the total number of documents in the collection
- df(t) = the total number of documents that contain the term t

The df of each term is provided by the TermDB module, while the average number of words and the total number of documents in the corpus is returned by the URLDB module. The details of each are presented in the respective section. While the BM25 formula is standard, we mentioned before that we use a modified version of the BM25 algorithm: this is valid, since in the above formula, we make use of the augmented term frequency

of each term, instead of their plain text frequency. As such, each term's frequency has been boosted according to their occurrence in any of the special HTML tags, thus receiving a higher score than terms appearing exclusively in the plain text.

4.4.5 Database Modules

The system employs many kinds of databases using the Atomicity-Consistency-Isolation-Durability (ACID) compliant embedded Relational Database Management System (RDBMS) of SQLite [58], which is part of the Python standard library. SQLite is a software library written in C (with bindings available in a multitude of other programming languages), that offers a complete working RDBMS system to be used in environments where a complete SQL server solution cannot be afforded (in terms of resources). Its source code is available in the public domain, and it implements most of the SQL standard. Additionally, one of its distinguishing features is that it is an integral part of the client application that uses it, instead of a separate process to be accessed by the client application like others SQL database systems. The entire database (definitions, tables, indices and actual data) is stored in a single cross platform file on the host machine. As such, it presents the perfect solution for our storage needs concerning permanent historical and statistical data maintained by the system. We have a group of modules that each maintain a separate SQLite database file, used to store data needed by the system. Each module is named after the database that it maintains and processes, according to the purpose it serves. In the following sections we shall present further details about each of these modules.

4.4.5.1 InlinkDB Module

The InlinkDB module, maintains a database file with all the links encountered so far in the pages the user viewed, along with a selection of their most recent anchor text terms. This serves as a source of inlinks for the Scorer module, where the current URL is provided as input and a list of its last X terms present in anchor

text found in other pages linking to it are returned; if the requested URL has not been encountered before in any other visited page, an empty list is returned. X is a number that can be customised according to a parameter provided during the module's instantiation in an object. Another customisable attribute of this module, is the maximum number of inlinks it can store. The defaults of the above two parameters are ten terms per inlink and a maximum of ten thousand inlinks stored. The former has been selected in order to have an efficient retrieval time during the request of a URL's terms, while the latter because during tests, it showed the best storage space requirement to efficient range of available URLs ratio. The table structure of the inlinks database is quite simple, since it stores just a URL along with its last anchor text terms, with the URL being used as a Primary Key. Besides the function get_inlinks that we have already mentioned previously, the module provides another function, called add_inlinks, which receives the list of links extracted by the Analyser module, traverses it and adds or updates (if the URL is already present in the database) each respective entry. When more than the set number of terms have been stored in the terms field, then terms are removed and appended to the list, in a First In First Out (FIFO) fashion, thus maintaining the most recent terms.

4.4.5.2 URLDB Module

The URLDB module, maintains a database of all the URLS the user has visited, added in a chronological order, with the most recent appended to the end of the table. The database structure is yet again quite simple in inception: a URL is stored, along with the number of words contained in the respective HTML document and the visited date; the Primary Key this time is a complex one, formed by the combination of the URL with the visited date. As such, once a URL has been added for a given date, it will not be added again if visited during the same date. The user history is maintained in two respects: a quantitative one, and a chronological one. The former, defines a minimum and maximum number of URLs to be stored in the database, while the latter defines a time window, measured in days, to keep URLs. The mechanism is as follows: when remove_old_urls function is

called (which is scheduled to be called once daily, in order to maintain a consistent state of recent URLs), a check is first performed whether the database holds a number of URLs that exceeds the minimum number set during the object's instantiation (with a default value of ten). If it is found to exceed that minimum threshold, a search is performed to locate those entries that are beyond the time window (which is also customisable and defaults to ten days), meaning those that are considered stale by the temporal requirement, and removes them. Of course a limitation is imposed to the number of deletions to be performed, since we cannot delete beyond the minimum number of URLs. Another essential function to the module's overall purpose is the add_url function, which receives a URL along with its total number of words and stores it on the database. If the addition causes the database to exceed the total number of URLs stored, a subsequent deletion is issued, of the oldest record, thus the first one in the table; yet again, a FIFO approach is followed. Three more functions are provided by the module:

- get_word_count: receives a URL and returns its word count
- get_average_word_count: returns the average number of all the URLs currently stored in the module's database
- get_num_of_docs: returns the total number of URLs present in the database

Since we have already mentioned the word count of each URL too many times, it is time to justify the purpose it serves: since we make use of the BM25 scoring algorithm, we need a way to keep track of the number of words in the current document (which is available for each document during the actual scoring of each terms), along with the average word count of all the documents in the corpus. This is where the word count attribute of each URL is involved: by calling the get_average_word_count function, a summation of the word count of all the URLs present in the database is performed, followed by a division of the total number of URLs. The get_num_of_docs is also used during the scoring process, since it is used in the IDF formula mentioned before.

4.4.5.3 TermDB Module

The TermDB module maintains a database of terms and their respective number of appearances in different documents in the collection of visited URLs maintained by the URLDB module. The table it maintains is structured as follows: a term is used as the Primary Key, and along each term, its respective df value and last date seen is stored. The df value is updated for each occurrence in a new URL, while the last seen date is updated with the current date when a term is encountered in a new URL. This module also contains a customisable parameter, which defines a temporal restriction, concerning a time window (which is to be explained later on) with a default value of ten days. The functions provided by this module are the following:

- add_term: a term is provided as input to this function, which is responsible of adding it to the database. If it is a new term (is not present in the database), it is added with a df value of one and the current date; otherwise, its current df value is increased by one while its current last seen date is replaced with the current date

- get_term_frequency: a term is provided and its respective df value is returned; should a term not be available, a value equal to zero is returned

- update_old_terms: this is another function that is scheduled for a daily execution, preferably at the same time the remove_old_urls function of URLDB module is scheduled. It is responsible of updating the df value of all the terms in the database that fall in one of the following three categories of temporal classification:

  - for terms that have not been seen for the last time for up to half the days of the current time window (in the default situation, that have not been seen in the past five days or less), a decrement of their df is performed, with its value decreased by a penalty of ten percent for each day it is not encountered

- for terms that have not been seen up to the time window and a half (for the default case, up to fifteen days past), the penalty is increased to a twenty percent per day decrease of the respective term's df value
- for terms that have not been encountered for exactly the time window and a half days (default of fifteen days), the terms are removed altogether from the database

An example might help to better illustrate the mechanism: supposing that the term "python", having a df value of 10, has not been encountered since yesterday, its current df value will be decreased from 10 to 9 (a ten percent decrease). This will also happen every day the term is not seen, up to the fourth day, where its df value will have reached a value of 6.6. Should the term continue to not be seen, the decrement step will be increased from the fifth day up to the fourteenth day from ten to twenty percent. Consequently, on the fifth day its df value will be 5.2, and leading to the fourteenth day, will reach a value of 0.7. If not seen for a fifteenth day, it will be set to 0, or for matters of database performance, removed altogether. Should the term have been encountered sometime within these days, its df value would have increased by one and the whole procedure would have started over. The rationale of such a choice is clearly in terms of maintaining the most current database in respect of the terms it maintains, while allowing for terms that are less frequent the chance to be present for a period of time larger than the specified time window. The change in the decrement step is justified by the fact that we wanted the df value of less frequent terms to reach a value close to zero as days go by without it being seen: should we have maintained a steady decrement step of ten percent, the df value in the example above would have reached the value of 2.3 in the fourteenth day, thus not quite close to be deemed as stale. Of course, a more raw approach could have been followed, by simply removing the term upon reaching the time window, yet since this procedure can be scheduled to happen at a time when the user is not browsing, we opted for a more sophisticated approach (thus more resource hungry) that we deemed more representative of the

staleness of terms.

Having mentioned two functions so far that need to be executed on a daily basis, let us mention the reason why: these functions, should they occur on the path of normal execution of the system, would have imposed a great drawback in performance, since when the size of these databases nears its maximum number of items stored, their functionality would have required a lot of time and resources to complete. As such, we opted for an offline execution, meaning they can be scheduled to be performed at times where users are usually offline, such as early in the morning. By that choice, we attain the desired functionality with the least disruption to the user's experience.

4.4.5.4 Profiler Module

The Profiler module is the last of the database modules used in the system. It is responsible of maintaining the list of terms that correspond to the user profile. The table that stores the user profile data in the database is structured as follows: a term acts as the Primary Key, and is associated with a respective weight, indicating the augmented term frequency of that term, normalised using the total document length. The module has three parameters, concerning the weights assigned to a term already in the profile, the weights assigned to a term meant to update a term in the profile, and the maximum number of terms the profile should maintain. As a module, it provides the following two functions:

- get_items: returns the terms the profile currently holds, along with their respective weights; this function is only used internally in the module
- update_profile: this provides the main functionality and logic concerning the way the profile is maintained and updated. This function receives the list of the top terms in regards to term frequency as they are extracted by the Scorer module (via its get_top_k_terms function), in order to add them to the user profile. The criteria with which a term is admitted in the profile are the following: initially all the terms already present in the profile, have their weight reduced by a ten percent rate, upon each call of this function. This

is performed in order to decrement each term's weight in order to facilitate the renewal of those terms: if a term already present in the profile has an extremely high weight, if that weight was never reduced, it would never be replaced by another term. As such, this would contradict the nature of a user profile: it needs to be temporally updated, since a user's preferences can evolve over time. Afterwards, each of the new terms is examined to determine whether it is already present in the profile. If it is, its new weight is calculated by adding the new weight to the weight already present in the profile. The formula is as follows:

$$ptw_{updated}(t) = ptw_{current}(t) \times profile_{factor} + ntw(t)$$

where

- $ptw_{updated}$ = the updated profile term weight for term t
- $ptw_{current}(t)$ = the current profile term weight for term t
- $profile_{factor}$ = the factor of the profile term weight, by default 90%
- ntw(t) = the new term weight for term of

By the above formula, it is easy to deduce that when one of the terms present in the updating term list is already present in the profile, it updates that term's profile weight by addition of its actual value: that is justified by the fact that we want a new term's weight to influence the profile term's current weight, since the profile weight is reduced by a ten percent factor each time the profile is updated; in other words, every time the user visits a new web page.

This is what happens when a term is already present in the profile and it gets updated. The procedure is altered significantly when a term is altogether new to the profile: in order for such a term to be included, a further check is necessary in order to determine whether the profile has reached its maximum size (regarding the terms it contains) or not. If it has, we search for all the terms in the profile that have a lesser weight than the new term's

weight multiplied by a factor of 0.9. If any are found, the lesser of them is replaced by the new term. On the other hand, if the profile has not reached its maximum allowed size, the term is added anyway, with its weight unaltered. This happens since we intend to fill the profile as soon as possible (from an initial empty state that is), and then slowly refine it by using the procedure described above.

After calling this function, a list of the updated profile terms and their respective weights is returned to the caller, in our case, the Client module.

4.4.6 Messenger Module

The Messenger module is responsible for the transmission of the web page and the user profile vectors to the server, via a Simple Object Access Protocol (SOAP) message. Since the server is written in C++, we have dealt with some problems concerning the communication and transferring of data through use of the suds Python SOAP library [59], mainly due to incompatibilities with specific data types defined on the server side as custom structs, that could not be simulated properly in Python. As such, and in order to circumvent that problem, the Messenger module combines the web page and user profile vectors in a predetermined file format, stores that file and calls a compiled client program in C++ which parses that file and handles the communication with the server via the SOAP protocol. After transmitting the data, a response containing the recommended advertisements is received, and these advertisements are displayed, for the time being on a terminal window.

4.5 System Advantages & Innovative Features

The system we designed, offers a complete solution in terms of personalised advertising from the ISP to their subscriber. Having analysed all the drawbacks and issues that caused other commercial solutions to fail in the Related Work chapter, let us justify how our system circumvents these problems:

1. Concerning user privacy: the system is fully controlled by the ISP, so all the data exchanged between client and server remain private and totally secure. The system is fully open to the ISP, in terms of its inner workings, as opposed to the approaches used by Phorm and NebuAd, where the ISPs was provided with proprietary systems, which were black boxes to them, since they could not be aware of what they did and how they did it.

2. Concerning the user awareness and consent issues: the client side of the system can be provided to users pre-installed on the router shipped by their ISP, and a clear indication in the contract that mentions the system's existence and purpose; thus, a user will be fully aware of what the system does and how it does so.

3. In regards to the possibility of disrupting a content publisher's revenue stream, in a similar manner to how Phorm and NebuAd operated: the system does not replace any of the existing content on a web page, thus allows all the advertisements the content publisher may have embedded on their page. The recommended advertisements originating from the ISP's server can be displayed on a separate element in the page, or via a browser plugin, that could also be provided to the ISP's clients should they opt-in on the system.

4. Concerning the user's ability to opt-in instead of opt-out of the personalised advertising service provided by the system: the router should by default be shipped with the client side system turned off, and proper documentation should be provided to the end user that mentions the necessary steps in order to enable the system's operation, should they want to. This could easily be attained by providing a simple check box on the router's configuration page that enables or disables the system. Should the system not be operational, no keyword extraction from the web pages the user visits should take place, and no user profile should be maintained. Furthermore, since the system in fully independent for each client/subscriber, no monitoring of their traffic can take place without their permission. As such, when they opt-out of

the service, they can be absolutely safe and doubtless of their choice being respected.

5. Concerning user anonymity, the problem is non existent due to the way the system has been designed: no tracking cookies are employed, and as such, there is no need for masquerading cookies and transmitting them to any third party web site, thus risking any chance of sharing sensitive information regarding the user, that could possibly lead to their identity being revealed. Furthermore, even the exchange between the client and the server sides, pertains no possibility of identifying a user, even by the ISP: the messages exchanged are anonymous, and once the server processes the web page and user profile vectors, discards them instantly, thus leaving no room for malicious actions originating even from within the ISP's staff.

6. Finally, concerning the possibility of sensitive user data being disclosed to a third party: the system, by design, communicates with no intervening third party, besides the ISP's own servers. As such, no disclosure can occur to any third party. As was mentioned above, no data is stored beyond the processing of each request, subsequently no data can be shared to any third party.

The system also offers some major advantages to the ISP:

- The ISP takes up a new role, as it acts as an advertisement network towards its subscribers. This has the following gains:
  - First and foremost, it presents a new revenue channel for ISPs, since it allows them to provide another service to their subscribers, and also opens up a whole new list of possible clients, since while operating as an advertising network, they can attract all those individuals that want to advertise their products or services in order to publish their advertisements to a targeted audience.
  - ISPs are in possession of a great asset as has already been mentioned: they are situated very close to the targeted recipients of advertise-

ments, consequently, they have access to the most powerful profiling method, that of user history. As such, the ISPs have a better overview of a user's activities and interests. By combining the latter, they can construct a much more precise user profile for each of their users, which can lead to them serving their subscribers better targeted advertisements. This is a twofold advantage, since the user is presented with advertisements that are tailored to their personal preferences and interests, and the advertising parties enjoy better and more targeted impressions of their advertisements to people that are more likely to be interested in those. As such, both the user and the advertiser enjoy benefits. Since both of them are clients of the ISP, by increasing their satisfaction, the ISP is bound to increase its own revenue.

- As we have already mentioned, the client side of the system is developed as a thick client, since most of the functions necessary for the system's operation are performed on the client: monitoring user traffic, extracting keywords from web pages and user profiling, are all performed on the subscriber's router. This is an advantage for the ISP, since it facilitates a decentralised operation of the system, thus decreasing the load on the ISP's main server infrastructure and removing any choke points on the server side. Consequently, the system cannot be plagued by a single point of failure.

4.5.1 Innovative Features

The proposed system does not lack in terms of innovation:

1. The corpus maintained and used during the scoring procedure is not domain specific, as is usually the case. It is dynamic, since it consists of many combined fields of interest, due to being constructed according to the end user's focus areas. As such, it can offer much better results to many application domains, instead of being focused entirely on a specific area.

2. It combines two different descriptions of the user, the first focusing on their current interest and the second originating from their most recent browsing

habits. The former is of course a direct product of the current web page they are browsing, which forms the web page vector extracted by the Scorer module, as has already been described. The latter comes in the form of the user profile vector that is maintained by the Profiler module, which is updated using output from the Scorer module. Subsequently, the system can provide a hybrid approach in contextual advertising, since it does not focus entirely on the current web page, but also on the recent history of web pages viewed by the user. As an illustration, if a user is currently viewing a page concerning aeroplane tickets, based on contextual terms alone, they would be shown advertisements regarding travels, holidays, hotels, bookings, et al. By employing their most recent history though, which could contain terms regarding computer science studies, some additional advertisements regarding computers products, hardware, books, et al. would be displayed alongside the travel related advertisements. In that way, the user could be presented with products and/or services that are better suited on their overall recent history and not just their current point of interest.

3. Despite its application on a real time environment, the system makes use of a corpus, in order to take advantage of more than just the term frequency in each document. Consequently, it relies on more than just term frequency in order to distinguish a web page's more distinctive terms.

4.6 Difficulties & Problems

During the development and evaluation stages, many obstacles had to be surpassed, since there were many difficulties and problems that had to be dealt with:

- One of the most common problems encountered during the system's development stages, was the attitude towards the character encoding used during the web page parsing and term extraction: although an assumption of dealing with web pages written only in the English language was adopted, mainly due to the stemming implementations being solely available for that

language, a problem soon arose with the encoding of various characters. When a character match in the latin encoding could not be found, the code would throw an exception and the execution would halt. As a consequence, we switched to a default Unicode encoding, which has a mapping for most of the characters available. The assumption of English being used, still holds of course, yet this has saved us the trouble of individually dealing with all the special characters and symbols that can appear in a given web page.

- Another problem already mentioned before, is the fact that Squid does not cache all the pages a user visits. This happens due to many pages being dynamic instead of static, meaning that the majority of their content originates through JavaScript code. When dealing with such pages, Squid cannot cache them as the content is retrieved dynamically through the browser, as the user navigates in the page. This is the reason we need to inspect the cache action column in the Logwatcher module, in order to determine whether a page is in the cache and can be processed by our system or not.

- Many web pages already contain embedded advertisements, consequently they generate further requests to fetch different URLs during their retrieval from the ISP's servers. This of course generates more entries in Squid's logs, where some advertisements can fall into the same category as the content we are analysing and parsing: the "text/html" content type might appear if the advertisement is not in a banner form, and this can subsequently trigger the execution of the top level Client module (encompassing all the other modules of the system besides the Messenger) with that object as input. This can inject a small amount of noise in our corpus, and trigger an excessive message exchange with the server to request advertisements, based on the advertisement web page. This has been deemed as an accepted noise, since it would require much more effort to block such requests since a black list of URLs would need to be maintained and updated in order to know which URLs refer to advertisements and not actual content pages.

- The actual display of the advertisements on the requested page would be accomplished via the use of the Internet Content Adaptation Protocol (ICAP) protocol [60] for versions of Squid up to (but exclusive of) version 3.1 and Embedded ICAP (eCAP) [61] for versions of Squid like 3.1 and later. The ICAP protocol, in short, is a protocol that allows a separate server process to operate in tandem with Squid and modify the content of a page before Squid dispatches that page to the requesting client. The reason this was not implemented, was that due to the restrictions imposed by the target platform, we could not spare resources for an extra server process to be operating on the router. eCAP, on the other hand, is ICAP's evolution, since it does not need the separate server process and instead can communicate directly with Squid for the content adaptation procedure. It is supported in Squid versions 3.1+. Unfortunately, up until the time of this writing, the documentation provided via the official page is limited to "a couple of hints to get started", which demanded a large investment of time in order to understand how the code operates. It is also worth noting that the provided code was in C++, which meant we had to either code that part in C++ or try to port the code base to Python. Instead, due to the limited time we had available, we opted to display the advertisements on a terminal prompt, in order to be able to evaluate the system's performance.

4.7 Alternative Approaches

During the initial system design process, we considered many alternative approaches before settling to the proposed architecture. The most noteworthy are mentioned below:

- During the first phases of the system's design, we started working to implement a custom HTML parser in Python, after having discovered the Beautiful Soup parser and its non satisfying performance in terms of speed and efficiency. However, in a short term, we discovered that the approach we were following was itself becoming quite cumbersome, and thus decided to

switch to the more versatile selected method of employing regular expressions.

- Another approach initially considered, was to develop outside the router altogether, and create the client side of the system as a browser plugin. This would still offer the opt-in ability we required, and could harness the user's computer as the underlying hardware platform, meaning much more available resources, when compared to the router's specifications. This approach however, faced other problems: most of the modern web browsers, execute their plugins in a sandboxed mode, restricting their access to just inside the browser process and not the user's filesystem, for obvious security concerns. As such, we could not call an external executable such as one implementing our software methods to process the current web page. A web browser plugin can still be developed though, as an alternative approach to using an ICAP or eCAP solutions, in order to display the advertisements the server recommended.

- An initial system that we tried to work with, was the Whoosh indexing and searching library [62] written in Python. It offers an infrastructure that indexes the documents it is given, where the user can define specific fields, besides plain text. It also provides a large variety of scoring algorithms, including BM25F. After close study of the documentation, we tried to experiment with a simple setup, storing the plain text of web pages and all the extracted tag lists from the Analyser modules as separate fields for each web page. However, we soon discovered that the index files' size was disproportional to the volume of data stored, and as such abandoned the idea of using it as the base of our system. Instead we designed our custom indexing scheme, in the form of the Database modules described earlier. Nevertheless, Whoosh is constantly evolving and could be a fine alternative in the future.

- We also inspected the modules and code base of the Natural Language Toolkit (NLTK) [63], which is written in Python. The NLTK is a solid

library containing various modules that can be used in a wide assortment of applications concerning language processing and text analytics. We initially used the implementation of the stemming algorithms provided by the project in our system, before switching to the implementation of the Porter stemming algorithm used in the proposed design.

- Other libraries we investigated before concluding on the final system design, and are worth mentioning, are the following:
  - The Terrier IR Platform [64], which is an open source search engine, written in Java, and developed at the School of Computing Science at the University of Glasgow. It offers an implementation of the BM25 scoring algorithm, and a full indexing backend, that could be employed by our system. However, the implementation is solely available in Java, thus it was considered a non viable choice due to the performance restrictions imposed by the initial system design.
  - The PyLucene [65] wrapper of the well known Lucene project of the Apache Foundation. It provides an Application Programming Interface (API) written in Python, that allows direct access to the original Java code base of the Lucene project. Since it is not a port of that code, it was rejected as a choice due to the same performance requirements of the system.

The aforementioned systems, libraries and solutions, are still very fine choices and can in fact be used in many other projects of similar interest. Nevertheless, none of them were used in our proposed system, due to failing to meet our required performance standards.

Chapter 5

Evaluation Method & Results

In this chapter we shall describe the evaluation method followed to test the performance of the system, how and why it was chosen, and the results it produced. Following the presentation of the results, is a short discussion concerning their quality.

5.1 Method Description

In order to test the system as a separate entity, meaning the client side alone, we had to follow an approach specifically tailored to the functions it offers: as such, we had to select a method that could evaluate the results of the two separate outputs produced by the system; the extracted key words from each web page, forming the web page vector, and the constructed user profile and its ongoing evolution as web pages are used to update it.

In order to be able to evaluate the quality of the results, we had to create a set of use case scenarios, that would offer adequate and varying test data, able to provide a means to emulate a user's browsing behaviour. The reason we chose to emulate a user's behaviour is due to the sheer amount of time it would take to conduct our tests with actual human users, operating the system in real life conditions. As such, we had to create a selection of web pages, that would be ”visited” in succession, each being processed by the system each time, producing

their respective web page vector by extracting their most significant keywords, while at the same time updating the respective user profile.

Users have short and long term interests [19], that are usually represented by specific re-occurring categories of topics, and various ”surges” of topics belonging to altogether different categories. In order to make things clearer, an example might help: a computer science student, during the course of their studies, is for instance interested in computer programming, thus searches the Internet for various programming tutorials, guides, and forums; since the student is an avid sports fan, they constantly check various sports sites in search for the latest news regarding their favourite sport and their favourite team. However, they tend to spend some of their time, browsing for topics and sites that do not fall in any of the aforementioned categories: for example, during the coming of the summer holidays, the same student might be interested in booking a flight to an island, and reserving a hotel room to spend a quiet, relaxing week, away from their studies. They might also be looking for tickets to their favourite rock band performing in a concert next month. The former activities represent the user's long term interest areas, while the latter their short term ones. These short term interest areas tend to fade, once the needs regarding these areas are satisfied: once a user booked their flight and hotel room, they will not keep searching for flight schedules and vacant hotel rooms.

The approaches followed during the course of evaluating the system, were the following:

- Run test suites consisting of a mix of documents, originating from two specific categories/topics and a number of random ones.
- Run a single topic test, where the selected documents are selected in a biased mix favouring a single topic.

In order to be able to collect a large, varying number of web pages, that could meet the aforementioned requirements (categorisation of subjects, large variety of available categories), we decided to form our pool of web pages based on Wikipedia articles. Wikipedia, is the free encyclopaedia, that offers user edited

and maintained content, and is accessed by millions of users daily, on a global scale. As every encyclopaedia, it offers articles concerning a wide range of subjects, covering almost any topic imaginable, even topics not found in traditional encyclopaedias.

5.1.1 Wikipedia Corpus

The major strong point that favoured the choice of Wikipedia articles as our collection of test data, was the fact that it offers categorisation of all its articles into a multitude of categories, ranging from the very abstract ones to the more specific and detailed ones. As such, we could select a number of categories to fetch articles from, and then form a proper article selection method that incorporates a proper mix of selected categories and random articles; the former representing the long term and the latter the short term interests of a user. In order to automate the procedure, a script was created, that uses a file[1] obtained from [66], that contains a list of all the article titles contained in Wikipedia. The script loads the contents of this file into a list, and randomly selects article titles and fetches the respective web pages from Wikipedia's servers. After downloading the HTML content of each article, an inspection of that content is performed, in order to check whether it satisfies some predefined criteria: since we need articles that have a certain volume of content, we set a word limit of 750 words minimum. As such, each article, in order to be selected, had to surpass that threshold in total word count. Nevertheless, we had to take into consideration some facts regarding Wikipedia articles, in order to speed up the process:

- Stubs: a lot of Wikipedia's articles are categorised as stubs. This is what is deemed as an article in need of expansion, of more content added, in order to better cover the discussed topic. Articles are marked as stubs by a person authorised to edit Wikipedia content; as such, it is a clear indication of that person's knowledge of the subject, to mark the article as short in regards to its coverage of the subject at hand. Clearly, stubs are short articles; conse-

[1] the filename at the time of writing was *all_titles_in_ns0.gz*

quently, we wanted a quick way to determine whether an article is tagged as stub, and move on to the next article. Thankfully, all stub articles contain a short description at the bottom of the article content, declaring its stub classification. Therefore, the script constructed begins by scanning each article for such a description; if it detects it, it skips the article and fetches the next one.

- Another category of articles that are most of the time poor in terms of content, yet essential to the functionality of Wikipedia, are disambiguation articles: these articles are used to assist users find what they require, when the term they are looking for has many different meanings or applications; as an example, supposing the user is searching for ”recall”, they will most likely be directed to the respective disambiguation article that lists all the different uses of the term ”recall”, from which article they will be able to select the specific use they are looking for. These articles are usually short in terms of content, and can thus be skipped entirely. The same approach is followed as with stub articles: a description is present at the bottom of these articles that declares them as disambiguation articles, so a search is performed for that description, and if found, the article is skipped.

- All of the articles and pages in Wikipedia have a standard form, that is the sidebar with a number of links to other Wikipedia pages, the header with a search bar, edit and article history buttons, and the footer with various links concerning terms of use, privacy policy, licensing and last modification date of the article. Since all the aforementioned are present in every Wikipedia article, and thus account for a standard word count that could influence our word threshold criteria, we decided to focus on the content of an article: thus, a search is performed in the HTML code of each article, for the part that contains the actual content; that is denoted via use of a tag with an assigned ”id” attribute of ”bodyContent”. Since we also need the heading that contains the title of the article, the respective <h1>tag is maintained, with an assigned ”id” attribute of ”firstHeading”.

- Tables: a large number of Wikipedia articles contains tabled data, in order to facilitate easy comparison of items. There is a large number of articles that are essentially comparison articles, such as the comparison of programming languages, where all programming languages and the features under comparison are presented in tabular form, in order to ease the process for the reader. As such, we needed to find articles that do not fall in that type of article, since we needed pure text content. To facilitate this, during the word counting phase of an article, tables and their contents are altogether removed and therefore not accounted for as words in the article. Of course, if an article is selected, these tables are still used as content, since they do contain valuable information. The rational was to avoid purely comparison articles.

The aforementioned procedure is used to randomly select a given number of articles to download from Wikipedia's servers, should they satisfy the aforementioned criteria.

5.1.2 Biased Wikipedia Corpus

The procedure had to be slightly altered concerning the fetching of the categorised articles needed for the evaluation of the system. To do that, an initial human browsing was instigated through a selection of categories contained in Wikipedia, in order to select those categories that had a decent article count. After finding such a category, its respective URL was stored in a text file; the text file containing the total sum of these URLs was then passed on to a sub class of the aforementioned script, that proceeded to fetch all the article URLs contained in each category, perform the same checks for the satisfaction of the established criteria and then store each article. A modification that was necessary was to adjust the script to search only those URLs that refer to articles in a category: a category article (meaning the article representing the category itself), is structured in such a way that its content is made up of two separate parts, the first containing links to any subcategories the category may have, and the second the actual articles that belong

to that category. As should be clear by now, we are interested on the articles belonging to a category. Thus, we perform a search for another specific tag, with an ”id” attribute set to ”mw-pages”, which indicates the section of interest. We then traverse each of the links contained in that section, following the same procedure as mentioned above.

5.1.3 Evaluator module

An extra module was developed, in order to perform the evaluation needed. Besides the construction of files that contain all the URLs available in each of the corpuses (the random and biased ones), the core of this module is the generator of test suites and the actual processing of each test suite.

- Random Test Suite Generator: this is responsible of properly selecting articles from the pool of articles contained in each corpus, mixing them properly in random order and then produce an output text file containing the article URLs and their respective titles. It is provided with parameters indicating the number of articles required in each test suite, and the respective category to fetch articles from: as an example, consider that a test suite consists of a total of 1000 articles, of which 400 originate from the ”health” category, 200 from the ”football” category, and the remaining 400 are randomly selected articles. This is the pattern that will be used during our tests, since the biased article selection indicates two long term interest areas of the user (in the aforementioned case, ”health” and ”football”), while the random selection indicates their short term interests, which are topics that do not necessarily relate to their long term ones.

- Test Suite Processor: this is where the actual evaluation of the system takes place; the output text file of the Random Test Suite Generator is passed to the Test Suite Processor, which parses each line of the file, and initiates the Client module of the system, which contains all the modules that implement the proposed system’s logic. The modules excluded from the test suite are the Logwatcher and Messenger modules, since their functionality

is not necessary in this off-line evaluation method. The results of all the articles' processing (the web page and user profile vectors) are stored in a text file for a twofold purpose: first to compare with the keywords produced by the AlchemyAPI system, and second, to use human evaluators to assess the evolution of the user profile.

5.1.4 Keyword Extraction Evaluation

In order to evaluate the keyword extraction output of our system, we chose to compare its output for each processed web page, against the keywords returned by AlchemyAPI.

5.1.4.1 AlchemyAPI

AlchemyAPI is capable of extracting topic keywords from your HTML, text, or web-based content. According to their official site [67], it employs sophisticated statistical algorithms and natural language processing technology to analyse data, and provides keyword extraction that can be utilised to index content, generate tag clouds, and more. Among the functions provided, our evaluation method will make use of the keyword extraction feature, in order to compare the keywords suggested by their system, against the web page and user profile vectors of keywords extracted by the proposed system. It supports keyword extraction from URLs, local HTML files, or even plain text. The keywords returned by their system can be in a variety of formats, such as XML, JavaScript Object Notation (JSON), Microformats, et al. AlchemyAPI is capable of extracting topic keywords from content written in a variety of languages. Its advanced multi-lingual support enables foreign-language content to be categorized and tagged automatically.

In order to compare our extracted keywords to the ones returned by AlchemyAPI, another script is needed: one that parses the output text file of the Test Suite Processor, and performs API calls to AlchemyAPI for each of the URLs contained in the test suite. Then, a simple parsing of the XML response is needed, in order to keep the keywords and phrases returned. A point must be noted here:

AlchemyAPI supports phrases as keywords, while our system does not; as such, each phrase returned by AlchemyAPI will be broken into its individual terms. Following the aforementioned parsing and phrase breaking up, the evaluation proceeds by using some metrics to compare results between our system and AlchemyAPI.

5.1.4.2 Metrics Employed

The metrics mentioned above, had to be indicative of the fraction of keywords that were present in the system's output (the web page vector, the user profile vector, and the combination of both) and also present in the keywords returned by AlchemyAPI. As such, we were based on the Recall metric, and devised the following two formulas:

$$\text{WP Recall} = \frac{|\text{keywords}_{\text{AlchemyAPI}} \bigcap \text{keywords}_{\text{WP}}|}{|\text{keywords}_{\text{AlchemyAPI}}|}$$

$$\text{UP Recall} = \frac{|\text{keywords}_{\text{AlchemyAPI}} \bigcap (\text{keywords}_{\text{UP}} - \text{keywords}_{\text{WP}})|}{|\text{keywords}_{\text{AlchemyAPI}}|}$$

Where their combination provides the total Recall of the system:

$$\text{Total Recall} = \text{WP Recall} + \text{UP Recall}$$

5.1.5 User Profile Evaluation

The user profile is an evolving vector, used to represent the ever changing long term interest areas of the user, as has already been described in the respective section of the System Architecture. In order to evaluate its consistency, human evaluators are needed, where they are requested to note each URL visited, and rate the profile's representation of these URLs in short intervals. For example, considering a test suite of 1000 Wikipedia articles, the user profile will be inspected in snapshots occurring every 50 articles that have been visited, and its contents will be rated according to them successfully representing the articles visited so far, according to the following metric.

5.1.5.1 Metric Employed

The metric employed to quantify the user profile's contents and its overall evolution, is that of Precision:

$$\text{UP Precision} = \frac{|\text{relevant keywords} \bigcap \text{retrieved keywords}|}{|\text{retrieved keywords}|}$$

where the task of finding the number of relevant keywords contained in the snapshot of the user profile is performed by a human evaluator.

5.2 Results

In this section, the test suites used during the evaluation procedure are presented, mentioning the implicated categories of articles and the respective percentage of each, for every test suite. Following, the results concerning the Recall metric defined previously are presented, in charted form. In the next section, the human evaluation of the user profile and its respective results with the Precision metric are presented for each of the processed test suites.

5.2.1 Test Suites

The test suites that were designed for each of the tests performed, followed the same formula, bar the last one. An average of six hundred articles were downloaded from Wikipedia, for each of the following four categories:

- Computers
- Health
- Football
- Music

As such, we had to form a set of test suites that presented a proper mix of two categories, as was mentioned before, with each suite containing a total of 1000 articles. The suites that were selected and processed are presented in table 5.1

Table 5.1: Processed Test Suites Category Selection

Test Suite #	Category 1 (40%)	Category 2 (20%)	Remainder (40%)
1	Health	Football	Random
2	Computers	Football	Random
3	Computers	Music	Random

5.2.2 Keyword Extraction

The keywords extracted by the proposed system, had to be compared against the keywords returned by AlchemyAPI for each processed URL. Using the metrics presented above, concerning the modified Recall of keywords, we present a graphical representation of our findings for each test suite: specifically, three lines are present in each figure, where the first symbolises the web page vector's recall, the second the user profile vector's recall, and the final the cumulative recall of both of these vectors. Since the proposed system relies on both of these vectors, its overall performance is determined by the latter line; yet both of its individual lines are given, in order to better estimate each vector's contribution to the total system recall. It is also worth noting that the same test suites were processed a second time, after changing the way the user profile is updated during the course of the evaluation stage [2]. Concerning the resulting figures, figures A.1 and A.2 depict the results of test suite 1 and its altered repetition, respectively; in similar fashion, figures A.3 and A.4 results of test suite 2 and figures A.5 and A.6 results of test suite 3.

In order to provide a concise view of the results from the processed test suites, a collection of the averages obtained from each test suite is given in table 5.2.

As can easily be derived from table 5.2, the alteration in the user profile update method bears no significant impact on the keyword extraction process and its results.

[2]Details concerning that change are provided in the following section.

Table 5.2: Average Results from each Test Suite

Test Suite #	Web Page Vector Recall	User Profile Vector Recall	Total Recall
1	28%	13%	41%
1 (Repeated)	28%	12%	40%
2	28%	14%	42%
2 (Repeated)	28%	13%	41%
3	28%	14%	42%
3 (Repeated)	28%	12%	40%

5.2.3 User Profile

During the initial testing of the user profile, the approach used while updating the profile with the new terms returned by the Scorer module, was somewhat different than the one eventually used in the proposed system. The differences concerned mainly the following parameters:

- The initial factor of the weight of a term already present in the profile, was set to 80%, instead of the current 90%.
- When the user profile was full concerning the number of terms it contained, a new term would simply replace the term in the profile with the lesser value, with no penalty, instead of the 90% of its currently attributed value.
- When a term was already present in the profile, and had to be updated, its original weight would contribute by a factor of 80%, instead of 90% that is currently used, while the new weight would be added to the current by a factor of 20%, instead of being added at its entirety as it is now.

As such, we ran the test suites mentioned above, and started evaluating the snapshots of the user profile's evolution; these snapshots occurred once every fifty URLs were processed. By taking into consideration the category mix in each test suite, along with the past 50 URLs before each examined snapshot, we marked

each of the terms that were deemed relevant to the category mix of the suite. After each of these profiles were examined, a tendency to easily ”forget” important category related terms became apparent; as such, a fine tuning lead us to the optimised choices presented earlier in the Architecture chapter, as well as in the aforementioned list of differences. Subsequently, we initiated the test suites once more, and issued a second investigation of the evolution of the user profile. This time, the results returned were improved in a substantial degree, while at the same time managing to maintain most of the category related terms in each profile snapshot for a longer duration.

The findings of the aforementioned evaluation are presented in the figures that follow. Specifically, in figure B.1, we see the evaluation results concerning the user profile originating from test suite 1, in figure B.2 concerning results from test suite 2 and finally, in figure B.3 the results of test suite 3’s user profile evaluation.

5.2.4 Single Topic Test

In order to further evaluate the system’s behaviour, another test was conducted, where the document selection was biased towards a single category: the number of documents remained set at 1000, where 60% originated from the Computers category, since that was the category we were most familiar with and had the widest range of articles available. The results concerning the keyword extraction and user profile evaluations are shown in figures C.1 and C.2 respectively.

5.3 Discussion

In this section, a short discussion concerning the nature of the evaluation’s results is to take place.

First and foremost, the reason we had these results concerning the keyword extraction evaluation, is justified entirely by the reason that AlchemyAPI supports keywords in the form of phrases, while the proposed system does not. As such, in order to properly be able to compare their results against our own, we had to use the individual terms of the returned results, instead of the phrases as a whole.

Consequently, we had to deal with our twenty keywords against an average of sixty to seventy distinct terms from AlchemyAPI. Of course this is just a fraction of the range of terms contained in AlchemyAPI's results, and justifies the average recall obtained by means of the web page vector alone.

On the other hand, concerning the contribution of the user profile vector in the total recall of the combined vectors, it is quite low, most of the time. Yet this is yet again justified: during the calculation of the user profile vector's recall, we count just the terms that are not contained in the web page vector. As such, we exclude the terms that are most representative of the current page, and search only among the remainder of the user profile terms. Due to the nature of the selection of articles, which is biased towards two categories in a certain degree, yet quite random concerning the order these articles are processed by the system, the user profile is constantly evolving towards many directions; as such, it contains many terms that are not related to the current web page, yet to the broader categories the user is interested in. Subsequently, it contributes to the total recall of the system to an average low degree. However, its use is clearly displayed: it can supplement the keyword extraction method, by enriching it with terms not necessarily present on the web page vector, yet closely related conceptually.

Concerning the user profile evaluation, some interesting results were acquired: due to the initial weighting scheme employed, as already mentioned, the user profile vector precision was quite unstable; some of the terms that were most representative of the categories in each test suite, had a tendency to be removed from the profile in a very short time span, in contrary to their importance. Consequently, the snapshots that were examined by the human evaluators tended to have less terms that were relevant to the categories included in the test suite. Having noted that, the alteration that has already been presented occurred, which proved to be in the correct direction: for the same set of articles, processed in the same order, the precision of the profile was boosted significantly, as can be observed in the respective figures. An interesting note is the fact that although it is not visible from the figures, the terms that were most representative of the categories employed in each test suite, were usually ranked higher in the user profile vector in the im-

proved weighting scheme, thus successfully indicating the user's main topics of interest. This leads to the assumption that the user profile can be reduced in size to half and even less, on a production environment: instead of transmitting a hundred user profile terms with each request to the system's server side, we can just transmit a mere thirty instead, striking a better balance between actual user profile representation value and the size of data to be exchanged with the server. The improvements gained by this optimisation are a clear indication of further gains to be attainable, with even more fine-tuning. Concerning the reason the user profile evolution presents such spikes and/or surges during the course of its evaluation, is yet again explained by the randomness of the articles processed by the system: specifically, during most of the surges, the articles before the surging snapshot happened to be mostly random, thus affecting greatly the profile's evolution.

Finally, concerning the results of the single topic test, we can note that the performance of both the keyword extraction and user profile outputs is enhanced: mostly visible in the user profile evolution, we can clearly distinguish the rising trend present. The steep surges on snapshots 7, 10 and 16 are clearly interpreted in similar fashion as with the above occasion: before the aforementioned surges, there is a respective high rising trend, thus indicating the concentration of category related articles. On the other hand, leading to the surge is a high concentration of irrelevant (random) articles. However, noting the 60 to 40 mix (the former being biased articles originating from the Computers category, the latter being random), it is obvious why the graph presents an overall rising trend. It is also notable that the overall precision in this test is higher than on the respective mixed category test suites.

5.4 Performance

Concerning the performance of the proposed system, in terms of its memory footprint and overall execution times, we have the following notes to make:

- Since Python is an interpreted language, we have to subtract the memory cost of the Python interpreter itself in order to calculate the memory foot-

print of our system alone; this is on average on a mere 4 MB, while adding the Python interpreter, it is raised to 6 MB. As such, the system is quite lean in terms of its memory requirements. It is also worth noting, that having just the Logwatcher constantly operating, and generating the Client and Messenger modules when needed, further lowers the average memory consumption of the system.

- Execution times, as expected, are quite low, since we are interested in real time operation. This averaged to a 0.7 seconds per article/web page, for the whole keyword extraction and user profiling to take place. However, as expected, once the databases holding the inlinks, the URLs and the terms grew close to their maximum capacity (regarding the number of terms to store), a penalty to that execution time was inflicted. During the end of our tests, a single article took approximately 2.1 seconds to complete the necessary procedures. This is a clear indication that there is certainly room for fine tuning concerning the maximum number of terms to store on each database file. It is worth mentioning though, that AlchemyAPI's respective request/response times for just keyword extraction over the same single page, were on average timed around 2 seconds: this is of course due to the fact that API calls are performed over a network connection, and are thus influenced by network transmission delays.

Chapter 6

Conclusions & Future Work

In this final chapter of this thesis, a short presentation of the conclusions derived from our line of work is given. Following that, is a mention of approaches that can be followed in order to further evolve the system, by improving its performance, adding further functionality and even considering alternative target platforms.

6.1 Conclusions

The system presented in this thesis, is complete and presents a few novelties in terms of its implementation:

1. It maintains a corpus that is domain independent: it consists of many combined fields of interest, since it is constructed according to the end user's focus areas. As such, it can offer much better results to many application domains, instead of being focused entirely on a specific area.

2. It provides a hybrid approach in contextual advertising, since it combines two different descriptions of the user, the first focusing on their current interest (the current web page they are viewing) and the second originating from their most recent browsing habits (a combination of the most representative terms found in their recent browsing history).

3. Despite its application on a real time environment, the system makes use of

a corpus, in order to take advantage of more than just the term frequency in each document. Consequently, it relies on more than just term frequency in order to distinguish a web page's more distinctive terms.

PENDING CONCLUSIONS REGARDING EVALUATION

6.2 Future Work

The work presented here, is complete and accomplishes all it set out for. However, there is always room for further improvements, and certainly not all possible elements that could be taken into consideration during HTML web page analysis have been exhausted:

- CSS analysis: the CSS style code embedded in an HTML page, is in most cases quite indicative of the look and feel of the text it is applied to; as such, the CSS code can be utilised to augment the weighting scheme presented in this thesis.

- Use of conceptual analysis for better and broader keyword extraction: by employing conceptual analysis of the key terms present in the web page, we can further improve the page's expressiveness, and also take advantage of other related topic areas by expanding the web page's initial set of terms.

- Switch to classification instead of plain keyword extraction: a classifier module can be implemented to replace the keyword extractor module, thus using the results of the classification process to represent the web page and the user profile.

- Alternative target platforms: the system can be altered to operate in other platforms, besides the router. As has already been mentioned, nowadays most people connect to the Internet by mobile means, and have constant access to any information they require. The most prominent mobile platform is that of smartphones, where advertising is used in many non browsing activities. As such, the system can be adapted to operate on a smartphone

environment, and take advantage of the user's browsing behaviour in order to display personalised advertisements on all the applications they use on their smartphone.

- Improve system performance: late in the development cycle of the proposed system, we discovered the existence of alternative Python implementations, such as PyPy [68], which claim faster execution times, due to them employing Just In Time (JIT) compilers. A modification of the code base could be attempted, to test the validity of such claims in the proposed system. Another way to improve the system performance, could be of course to switch to another programming language, such as C or C++, in order to take advantage of its compiled nature. However, this approach would require a lot of external libraries to be included, since these languages have a much more limited standard library when compared to Python.

Appendix A

Keyword Extraction Evaluation Figures

Figure A.1: Test Suite 1 Keyword Extraction Results

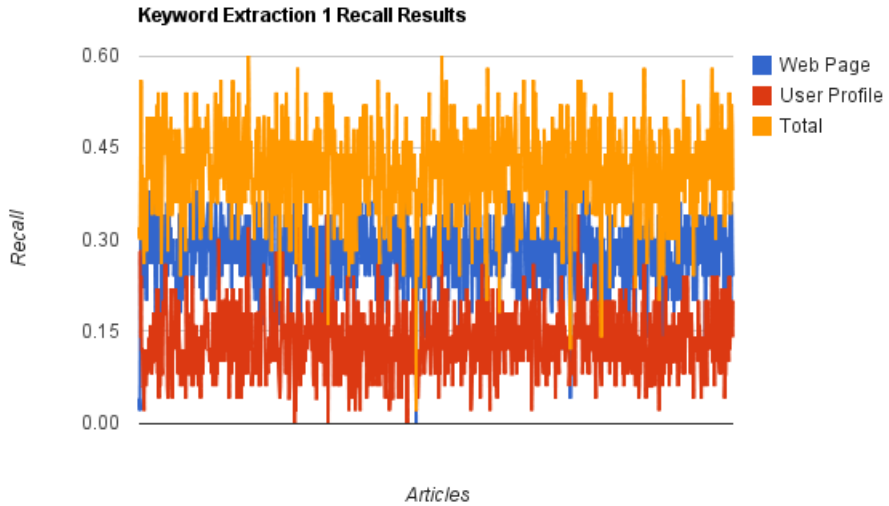


Figure A.2: Test Suite 1 (Repeated) Keyword Extraction Results

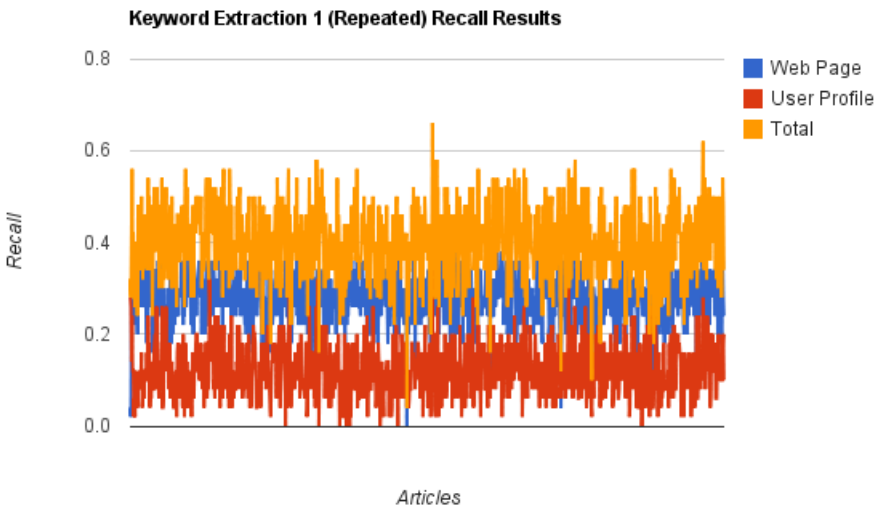


Figure A.3: Test Suite 2 Keyword Extraction Results

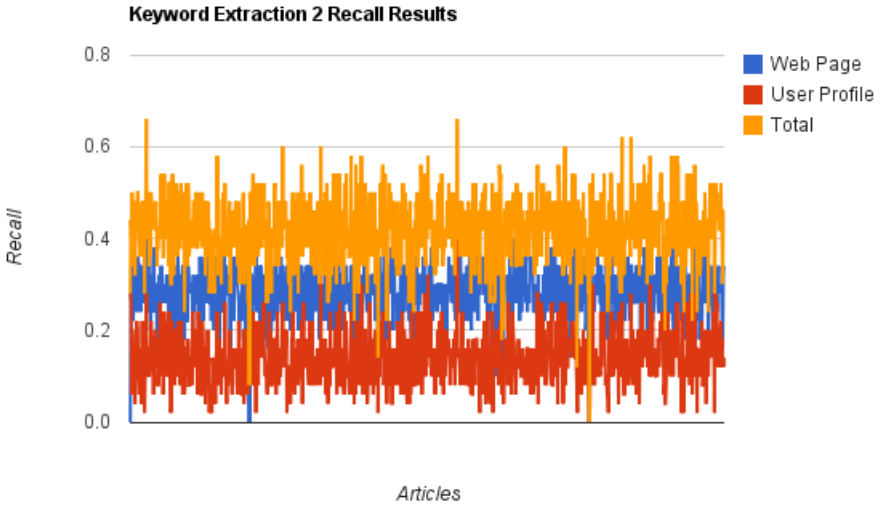


Figure A.4: Test Suite 2 (Repeated) Keyword Extraction Results

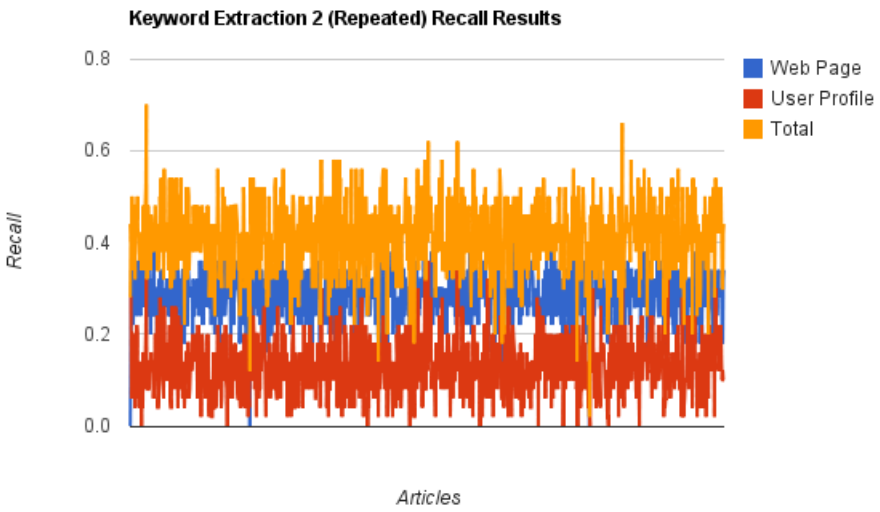


Figure A.5: Test Suite 3 Keyword Extraction Results

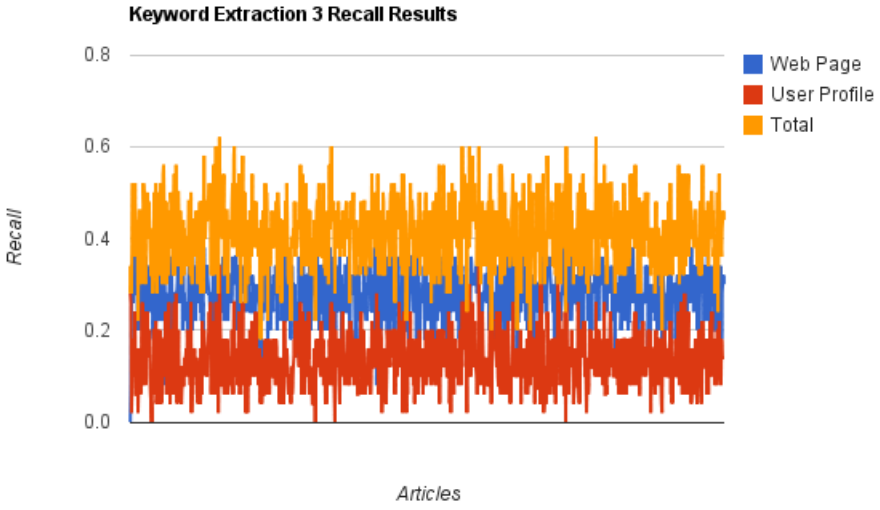
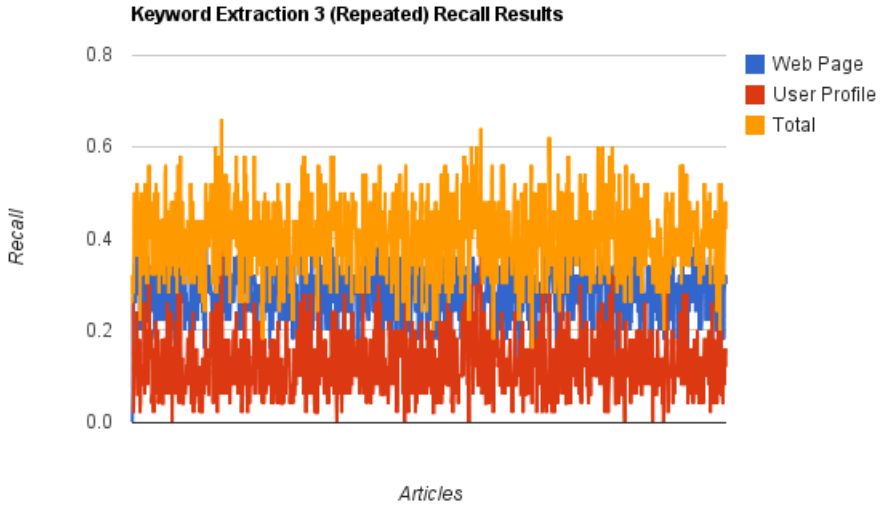


Figure A.6: Test Suite 3 (Repeated) Keyword Extraction Results



Appendix B

User Profile Evaluation Figures

Figure B.1: Test Suite 1 User Profile Results

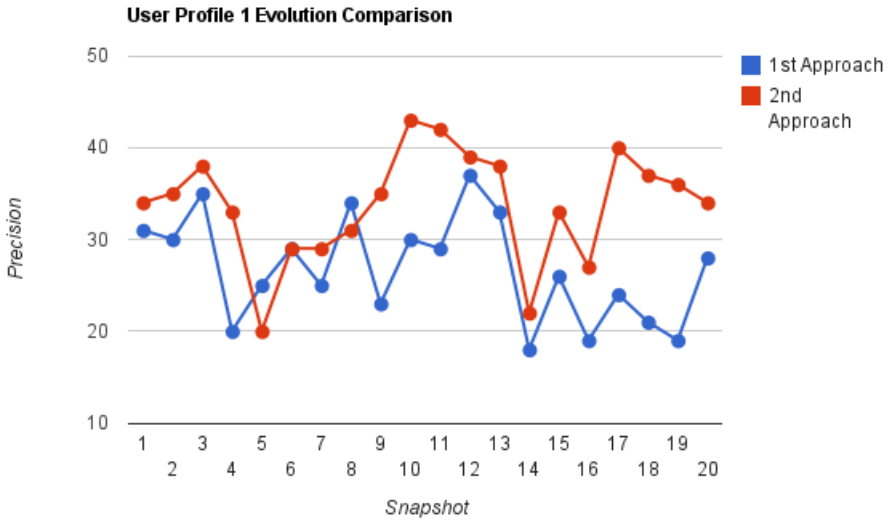


Figure B.2: Test Suite 2 User Profile Results

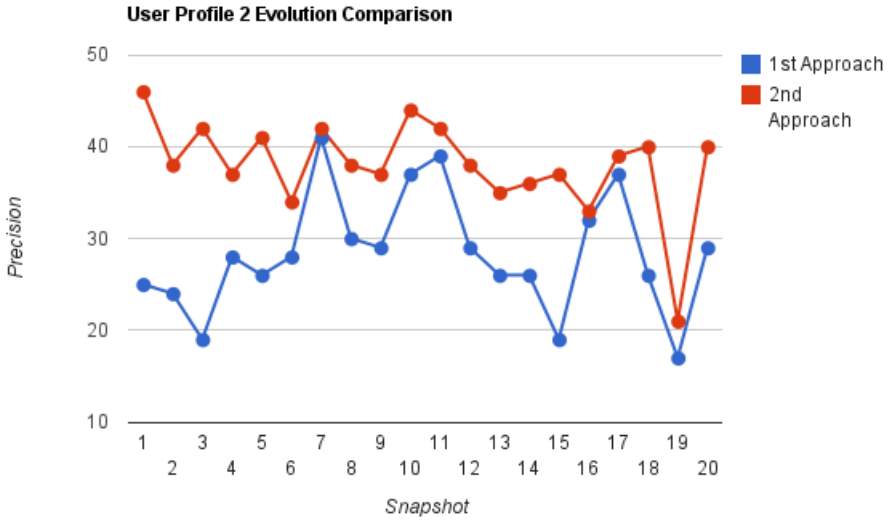
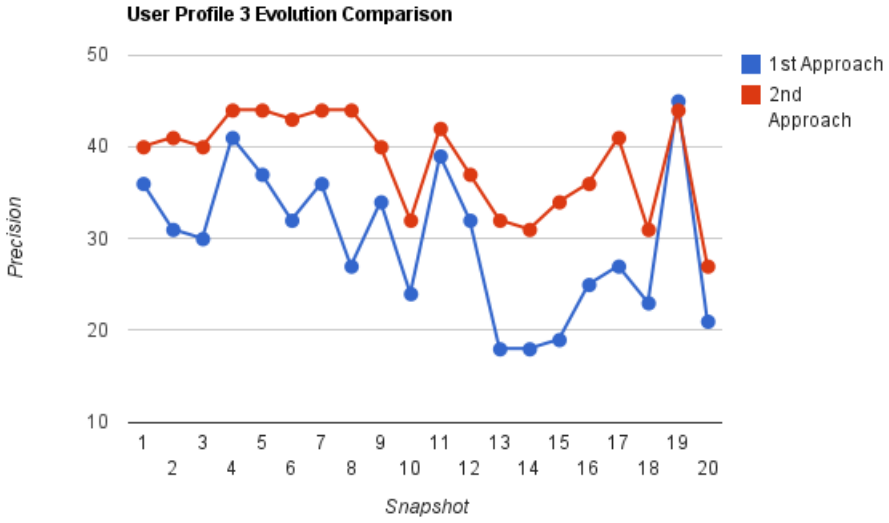


Figure B.3: Test Suite 3 User Profile Results



Appendix C

Single Topic Evaluation Figures

Figure C.1: Single Topic Keyword Extraction Results

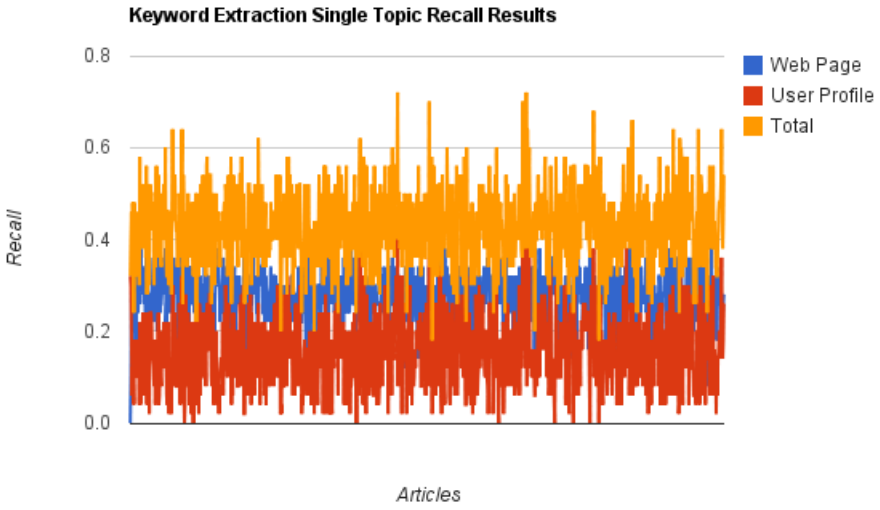
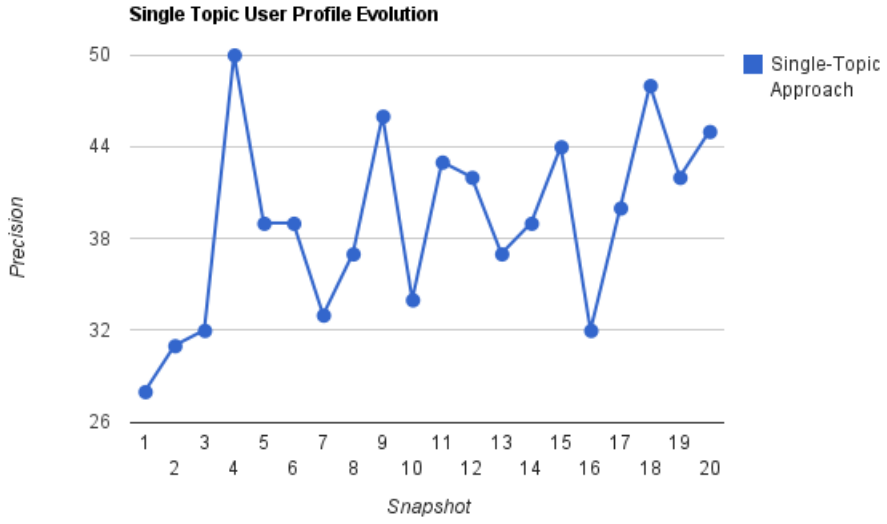


Figure C.2: Single Topic User Profile Results



Bibliography

[1] George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. J. Mach. Learn. Res. 3 (March 2003), 1289-1305.

[2] S. K. M. Wong and Vijay V. Raghavan. 1984. Vector space model of information retrieval: a reevaluation. In Proceedings of the 7th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '84). British Computer Society, Swinton, UK, UK, 167-185.

[3] G. Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. Commun. ACM 18, 11 (November 1975), 613-620. DOI=10.1145/361219.361220 http://doi.acm.org/10.1145/361219.361220

[4] Gerard Salton and Chris Buckley. 1987. Term Weighting Approaches in Automatic Text Retrieval. Technical Report. Cornell University, Ithaca, NY, USA.

[5] Aixin Sun, Ee-Peng Lim, and Wee-Keong Ng. 2002. Web classification using support vector machine. In Proceedings of the 4th international workshop on Web information and data management (WIDM '02). ACM, New York, NY, USA, 96-99. DOI=10.1145/584931.584952 http://doi.acm.org/10.1145/584931.584952

[6] Wen-tau Yih, Joshua Goodman, and Vitor R. Carvalho. 2006. Finding advertising keywords on web pages. In Proceedings of the 15th international conference on World Wide Web (WWW '06).

ACM, New York, NY, USA, 213-222. DOI=10.1145/1135777.1135813 http://doi.acm.org/10.1145/1135777.1135813

[7] Suman Saha, C. A. Murthy, and Sankar K. Pal. 2010. A novel split and merge technique for hypertext classification. In Transactions on rough sets XII, James F. Peters, Andrzej Skowron, Roman Słowiński, Pawan Lingras, Duoqian Miao, and Shusaku Tsumoto (Eds.). Lecture Notes In Computer Science, Vol. 6190. Springer-Verlag, Berlin, Heidelberg 192-210.

[8] Andrei Broder, Marcus Fontoura, Vanja Josifovski, and Lance Riedel. 2007. A semantic approach to contextual advertising. In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '07). ACM, New York, NY, USA, 559-566. DOI=10.1145/1277741.1277837 http://doi.acm.org/10.1145/1277741.1277837

[9] Interactive Advertising Bureau. Internet Advertising Revenue Report 2010 First Half-Year Results. http://www.iab.net/media/file/IAB_report_1H_2010_Final.pdf [retrieved on February 7, 2011]

[10] Interactive Advertising Bureau. Internet Advertising Revenue Report 2009 Full-Year Results. http://www.iab.net/media/file/IAB-Ad-Revenue-Full-Year-2009.pdf [retrieved on February 7, 2011]

[11] Deepayan Chakrabarti, Deepak Agarwal, and Vanja Josifovski. 2008. Contextual advertising by combining relevance with click feedback. In Proceeding of the 17th international conference on World Wide Web (WWW '08). ACM, New York, NY, USA, 417-426. DOI=10.1145/1367497.1367554 http://doi.acm.org/10.1145/1367497.1367554

[12] Berthier Ribeiro-Neto, Marco Cristo, Paulo B. Golgher, and Edleno Silva de Moura. 2005. Impedance coupling in content-targeted advertising. In Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '05).

ACM, New York, NY, USA, 496-503. DOI=10.1145/1076034.1076119 http://doi.acm.org/10.1145/1076034.1076119

[13] M. Ciaramita, V. Murdock, V. Plachouras: Semantic Associations for Contextual Advertising Massimiliano Ciaramita, Vanessa Murdock, Vassilis Plachouras. 2008. Semantic associations for contextual advertising. Journal of Electronic Commerce Research, Vol. 9, No. 1, 2008

[14] Xuerui Wang, Andrei Broder, Marcus Fontoura, and Vanja Josifovski. 2009. A search-based method for forecasting ad impression in contextual advertising. In Proceedings of the 18th international conference on World wide web (WWW '09). ACM, New York, NY, USA, 491-500. DOI=10.1145/1526709.1526776 http://doi.acm.org/10.1145/1526709.1526776

[15] Darren Charters. 2002. Electronic monitoring and privacy issues in business-mrketing: the ethics of the DoubleClick experience. Journal of Business Ethics 35, 243-254.

[16] Jianyi Liu, Cong Wang, Zhengyang Liu, Wenbin Yao. 2010. Advertising Keywords Extraction from Web Pages. WISM 2010. 336-343

[17] Kazunari Sugiyama, Kenji Hatano, and Masatoshi Yoshikawa. 2004. Adaptive web search based on user profile constructed without any effort from users. In Proceedings of the 13th international conference on World Wide Web (WWW '04). ACM, New York, NY, USA, 675-684. DOI=10.1145/988672.988764 http://doi.acm.org/10.1145/988672.988764

[18] Stuart E. Middleton, Nigel R. Shadbolt, and David C. De Roure. 2004. Ontological user profiling in recommender systems. ACM Trans. Inf. Syst. 22, 1 (January 2004), 54-88. DOI=10.1145/963770.963773 http://doi.acm.org/10.1145/963770.963773

[19] Susan Gauch, Mirco Speretta, Aravind Chandramouli, and Alessandro Micarelli. 2007. User profiles for personalized information access. In The adap-

tive web, Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl (Eds.). Lecture Notes In Computer Science, Vol. 4321. Springer-Verlag, Berlin, Heidelberg 54-89.

[20] RFC2965 - HTTP State Management Mechanism, http://www.ietf.org/rfc/rfc2965.txt [retrieved on February 7, 2011]

[21] Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. Comput. Linguist. 16, 1 (March 1990), 22-29.

[22] Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. Comput. Linguist. 19, 1 (March 1993), 61-74.

[23] Yutaka Matsuo, Mitsuru Ishizuka. 2003. Keyword extraction from a single document using word co-occurrence statistical information. In Proceedings of the 16th International FLAIRS Conference, St. Augustine, Floridam (2003)

[24] CNET News. Publushers sue Gator over pop-ups. http://news.cnet.com/2100-1023-940072.html [retrieved on February 7, 2011]

[25] Thomas A. Hemphill. DoubleClick and consumer online privacy: an e-commerce lesson learned. Business and Society Review, 105:3, 361-372.

[26] The Elvey Partnership I.T. Consulting Group. Security Problem Report SPR-2001-01-22. http://www.elvey.com/it/spr/SPR-2001-01-22.txt [retrieved on February 7, 2011]

[27] Phorm Inc. How it works. http://www.phorm.com/consumers/phormdiscover/how_it_works/index.html [retrieved on February 7, 2011]

[28] The Register. BT's secret Phorm trials: UK.gov responds. http://www.theregister.co.uk/2008/09/16/phorm_eu_berr/ [retrieved on February 7, 2011]

[29] The Register. BT targets 10,000 data pimping guinea pigs. http://www.theregister.co.uk/2008/03/05/bt_phorm_trial/ [retrieved on February 7, 2011]

[30] The Register. Phorm launches data pimping fight back. http://www.theregister.co.uk/2008/03/07/phorm_interview_burgess_ertegrul/page3.html [retrieved on February 7, 2011]

[31] Richard Clayton. 2008. The Phorm ”Webwise” System

[32] Light Blue Touchpaper blog. Stealing Phorm Cookies. http://www.lightbluetouchpaper.org/2008/04/22/stealing-phorm-cookies/ [retrieved on February 7, 2011]

[33] Nicholas Bohm. 2008. The Phorm ”Webwise” System - a Legal Analysis

[34] ICO UK. Phorm - Webwise and Open Internet Exchange. http://web.archive.org/web/20080412034139/http://www.ico.gov.uk/about_us/news_and_views/current_topics/phorm_webwise_and_oie.aspx [retrieved on February 7, 2011]

[35] MediaPost Publications. Case Closed: NebuAd Shuts Down. http://www.mediapost.com/publications/?fa=Articles.showArticle&art_aid=106277 [retrieved on February 7, 2011]

[36] The Washington Post. Every Click You Make. http://www.washingtonpost.com/wp-dyn/content/article/2008/04/03/AR2008040304052.html [retrieved on February 7, 2011]

[37] The New York Times. NebuAd Observes ’Useful, but Innocuous’ Web Browsing. ’http://bits.blogs.nytimes.com/2008/04/07/nebuad-observes-useful-but-innocuous-web-browsing/ [retrieved on February 7, 2011]

[38] Digital Destiny blog. Charter Cable to Spy on its Broadband Users to Serve Targeted Ads via NebuAd. http://www.democraticmedia.org/jcblog/?p=586 [retrieved on February 7, 2011]

[39] Ars Technica. Embarq: Don't all users read our 5,000 word privacy policy? http://arstechnica.com/old/content/2008/07/embarq-dont-all-users-read-our-5000-word-privacy-policy.ars [retrieved on February 7, 2011]

[40] S. E. Robertson and S. Walker. 1994. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '94), W. Bruce Croft and C. J. van Rijsbergen (Eds.). Springer-Verlag New York, Inc., New York, NY, USA, 232-241.

[41] Stephen Robertson, Hugo Zaragoza, and Michael Taylor. 2004. Simple BM25 extension to multiple weighted fields. In Proceedings of the thirteenth ACM international conference on Information and knowledge management (CIKM '04). ACM, New York, NY, USA, 42-49. DOI=10.1145/1031171.1031181 http://doi.acm.org/10.1145/1031171.1031181

[42] Hugo Zaragoza, Nick Craswell, Michael Taylor, Suchi Saria and Stephen Robertson. 2004. Microsoft Cambridge at TREC-13: Web and HARD tracks. TREC

[43] Min-Yen Kan and Hoang Oanh Nguyen Thi. 2005. Fast webpage classification using URL features. In Proceedings of the 14th ACM international conference on Information and knowledge management (CIKM '05). ACM, New York, NY, USA, 325-326. DOI=10.1145/1099554.1099649 http://doi.acm.org/10.1145/1099554.1099649

[44] Linksys WRT160N router page. http://www.linksysbycisco.com/EU/en/products/WRT160N [retrieved on February 7, 2011]

[45] DD-WRT home page. http://www.dd-wrt.com/site/index [retrieved on February 7, 2011]

[46] OpenWrt home page. http://openwrt.org/ [retrieved on February 7, 2011]

[47] DD-WRT about page. http://www.dd-wrt.com/site/content/about [retrieved on February 7, 2011]

[48] Wireshark home page. http://www.wireshark.org/ [retrieved on February 7, 2011]

[49] tcpdump & libpcap home page. http://www.tcpdump.org/ [retrieved on February 7, 2011]

[50] Squid caching proxy main page. http://www.squid-cache.org/ [retrieved on February 7, 2011]

[51] Python Programming Language Official Website. http://python.org/ [retrieved on February 7, 2011]

[52] Squid log format wiki page. http://wiki.squid-cache.org/Features/LogFormat [retrieved on February 7, 2011]

[53] Squid logs wiki page. http://wiki.squid-cache.org/SquidFaq/SquidLogs [retrieved on February 7, 2011]

[54] W3C. HTML 4.01 Specification. http://www.w3.org/TR/html401/ [retrieved on February 7, 2011]

[55] Porter Stemming Algorithm home page. http://tartarus.org/~martin/PorterStemmer/ [retrieved on February 7, 2011]

[56] Beautiful Soup home page. http://www.crummy.com/software/BeautifulSoup/ [retrieved on February 7, 2011]

[57] Python official documentation page. http://docs.python.org [retrieved on February 7, 2011]

[58] SQLite home page. http://www.sqlite.org/ [retrieved on February 7, 2011]

[59] suds SOAP Python client library. https://fedorahosted.org/suds/ [retrieved on February 7, 2011]

[60] ICAP-Forum Home Page. http://www.icap-forum.org/ [retrieved on February 7, 2011]

[61] eCAP Home Page. http://www.e-cap.org/Home [retrieved on February 7, 2011]

[62] Whoosh Home Page. https://bitbucket.org/mchaput/whoosh/wiki/Home [retrieved on February 7, 2011]

[63] Natural Language Toolkit home page. http://www.nltk.org/ [retrieved on February 7, 2011]

[64] Terrier IR Platform home page. http://terrier.org/ [retrieved on February 7, 2011]

[65] PyLucene home page. http://lucene.apache.org/pylucene/ [retrieved on February 7, 2011]

[66] Wikipedia Database download. http://en.wikipedia.org/wiki/Wikipedia:Database_download [retrieved on February 7, 2011]

[67] AlchemyAPI home page. http://www.alchemyapi.com/ [retrieved on February 7, 2011]

[68] PyPy official home page. http://pypy.org/ [retrieved on February 7, 2011]