Department of Computer and Information Science

Final thesis

Contextual Advertising Online

by

Jimmie Pettersson

LIU-IDA/LITH-EX-A--08/53--SE

2008-09-01
Final Thesis

Contextual Advertising Online

by

Jimmie Pettersson

LIU-IDA/LITH-EX-A--08/53--SE

2008-09-01

Examiner: Lena Strömbäck, LiTH-IDA, Databases and Information Techniques (IISLAB)
Abstract

The internet advertising market is growing much faster than any other advertising vertical. The technology for serving advertising online goes more and more towards automated processes that analyze the page content and the user’s preferences and then matches the ads with these parameters.

The task at hand was to research and find methods that could be suitable for matching web documents to ads automatically, build a prototype system, make an evaluation and suggest areas for further development. The goals of the system was high throughput, accurate ad matching and fast response times. A requirement on the system was that human input could only be done when adding ads into the system for the system to be scalable.

The prototype system is based on the vector space model and a td-idf weighting scheme. The cosines coefficient was used in the system to quantify the similarity between a web document and an ad.

A technique called stemming was also implemented in the system together with a clustering solution that aided the ad matching in cases where few matches could be done on the keywords attached to the ads. The system was built with a threaded structure to improve throughput and scalability.

The tests results show that you accurately can match ads to a website’s content using the vector space model and the cosines-coefficient. The tests also show that the stemming has a positive effect on the ad matching accuracy.
Acknowledgements

This report is my final thesis on my Master Degree in Computer Science at Linköping University.

I’d like to take the opportunity to thank my examiner Lena Strömbäck.

I’d also like to thank my colleagues at Getupdated Internet Marketing for their patience and input during the making of this thesis.

Thank you!

Jimmie Pettersson

Stockholm, September 1, 2008
Contents

ABSTRACT .................................................................................................................. III

ACKNOWLEDGEMENTS ........................................................................................... V

CONTENTS ................................................................................................................ VI

LIST OF FIGURES ...................................................................................................... XI

LIST OF TABLES .......................................................................................................... XII

1 INTRODUCTION ..................................................................................................... 1

1.1 OVERVIEW .......................................................................................................... 1
1.2 BACKGROUND ...................................................................................................... 1
1.3 GET UPDATED INTERNET MARKETING ......................................................... 1
1.4 METHOD ................................................................................................................ 2
  1.4.1 ANALYSIS ........................................................................................................ 2
  1.4.2 DESIGN ........................................................................................................... 2
  1.4.3 IMPLEMENTATION .......................................................................................... 2
  1.4.4 TEST ............................................................................................................... 2
  1.4.5 EVALUATION .................................................................................................. 2
1.5 READING INSTRUCTIONS .................................................................................... 2

2 PROBLEM DESCRIPTION ....................................................................................... 5

2.1 PURPOSE .............................................................................................................. 5
2.2 PROBLEM DEFINITION ....................................................................................... 5
2.3 CONSTRAINTS ....................................................................................................... 6

3 PROBLEM ANALYSIS ............................................................................................ 7

3.1 OVERVIEW .......................................................................................................... 7
3.2 SYSTEM FUNCTIONALITY .................................................................................. 7
3.3 IDENTIFIED PROBLEMS ................................................................................... 7
3.4 IDENTIFIED REQUIREMENTS ............................................................................. 8
  3.4.1 DOCUMENT DOWNLOADING ....................................................................... 8
  3.4.2 DATA STRUCTURES ....................................................................................... 8
  3.4.3 DOCUMENT ANALYSIS AND AD RANKING ................................................. 8
  3.4.4 DUPLICATE WEBSITES ................................................................................ 8
  3.4.5 PLATFORM .................................................................................................... 8
# 4 Technical Background

## 4.1 Overview

## 4.2 Ranking Models

### 4.2.1 Boolean Model

### 4.2.2 Vector Space Model

### 4.2.3 Probabilistic Model

### 4.2.4 Comparison of the Models

## 4.3 Stemming

### 4.3.1 Definition

### 4.3.2 Explanation

### 4.3.3 Algorithms

## 4.4 Thesaurus

## 4.5 Query Expansion

## 4.6 Clustering

### 4.6.1 Batch and Online

### 4.6.2 Clustering Algorithms

### 4.6.3 Cluster Representation

## 4.7 Dust Detection

### 4.7.1 DustBuster

### 4.7.2 SimHash

## 4.8 Similarity Coefficients

### 4.8.1 Tanimoto Coefficient

### 4.8.2 Cosines Coefficient

# 5 Design Choices

## 5.1 Overview

## 5.2 Vector Space Model

## 5.3 Duplicate Documents

## 5.4 Stemming

## 5.5 Cluster Dendrogram

## 5.6 Similarity Coefficient

## 5.7 Threaded Solution

## 5.8 Database Layer

# 6 Implementation

## 6.1 Overview

## 6.2 Complete System Flow Chart

## 6.3 Communication Between System Parts

## 6.4 The Retriever

### 6.4.1 Purpose

### 6.4.2 Implementation

## 6.5 The Analyzer

### 6.5.1 Purpose
List of figures

Figure 1 - Work Method..................................................................................................................2
Figure 2 - System overview ......................................................................................................7
Figure 3 - System parts ..............................................................................................................25
Figure 4 - System flow chart ..................................................................................................26
Figure 5 - Data Transitions in system ....................................................................................27
Figure 6 - Retriever flow chart ...............................................................................................28
Figure 7 - Analyzer flow chart ...............................................................................................30
Figure 8 - Cluster assigning flow chart ..................................................................................32
Figure 9 - Database structure table-diagram ........................................................................34
Figure 10 - Test system flow chart .........................................................................................38
Figure 11 - The test interface .................................................................................................40
List of tables

Table 1 - Document to Ad accuracy ..........................................................41
Table 2 - Matching ads per document ....................................................41
Table 3 - Average weight score ..............................................................42
Table 4 - Average weight score top 5 ....................................................42
1 Introduction

1.1 Overview

This is a master thesis made for a master degree in Computer Science at Linköping’s University. The project was conducted during the fall of 2007 and spring 2008. The examiner was Lena Strömbäck, Dept. of Computer And Information Science at Linköping’s University.

1.2 Background

The internet advertising market is growing much faster than any other advertising area such as TV or print. The technology for serving advertising online goes more and more towards automated processes that analyze the page content and the user’s preferences and uses that information to decide what type of advertising to display for an optimal advertising effect. The automation increases the efficiency of the advertising giving advertisers more relevant traffic and at the same time increasing the earnings for the websites displaying the ads.

The better you can target the advertisements the more you monetize the ad space at hand.

1.3 Getupdated Internet Marketing

The work was conducted at Getupdated Internet Marketing. Getupdated is a company working within many verticals of the Internet marketing business. Getupdated consists of two main business areas.

Media is working with internet advertising and search engine optimization. Solutions works with Intranets, Portals, Shopping systems and Web design.

This project was made for the Traffic Department which is a division within the Media section of Getupdated. The traffic department works with online traffic generation and optimization for its clients. The traffic department works with products like Display Advertising and Affiliate marketing. The traffic dept. works both directly with clients and through media agencies.
1.4 Method
The method used to perform this thesis is described by this flow chart.

![Flow Chart](image)

Figure 1 - Work Method

1.4.1 Analysis
This analysis started by examining existing literature and research in the area and selecting areas and solutions that seemed to be closely related to this thesis.

1.4.2 Design
The design phase was an iterative process where promising techniques and approaches were tested with regards of both speed and accuracy. In this phase decisions were made by considering time, performance and accuracy constraints.

1.4.3 Implementation
This phase was closely entwined with the design phase, when an algorithm was found appropriate it had already been tested and therefore an implementation was already done and it was a matter of adapting it to the data flow of the system.

1.4.4 Test
The prototype was tested by a group of 3 persons; the test group rated the system’s performance on a 5 level scale. Three test cases were considered during the evaluation.

1.4.5 Evaluation
In this phase an evaluation of the test results were done and also a discussion about the design choices and how well they matched the purpose and goals.

1.5 Reading instructions
This report aims at trying to first explain the basics and gradually building the reader’s knowledge in data mining techniques and contextual advertising.

Chapter 2 defines the problem this thesis report will work towards solving; the chapter also includes restrictions and constraints.

In chapter 3 a problem analysis is performed to identify the needed components and sub problems that have to be solved. Section 3.2 contains a very basic flow chart of the system functionality. Section 3.3 contains the identified requirements of the system.
Chapter 4 goes through the theoretical background and related research for the components that the system will consist of. The most important part in this chapter is section 4.2.2 that describes the implemented ranking algorithm that was used for the system. Section 4.8.2 describes the comparison algorithm that is used to compare web documents with ads and is also an important cornerstone in the system.

Chapter 5 defines the design choices taken when designing the system. This is an important chapter to read since it defines the important choices that were made and what techniques the system is based on.

Chapter 6 describes the way the system was implemented and what trade-offs had to be considered. Section 6.2 shows a flow chart of the whole system. Section 6.6 describes the implementation of the important Analyzer part of the system that does the analyzing of the document and also matches ads to the document.

Chapter 7 defines the test phase and the test results. It’s recommended to read the whole chapter to get a good picture of the test results.

Chapter 8 consists of a discussion about the results and theories to why the results look like they do.

In chapter 9 conclusions of the thesis are presented together with suggestions of future work to develop the system prototype further.
2 Problem description

2.1 Purpose

The purpose of this thesis is to research and find methods that could be suitable for matching web documents to ads automatically. When suitable techniques are found a prototype system needs be developed and an evaluation on how accurate the methods are in matching the ads to the web documents.

The goals of the system are high throughput, accurate ad matching and a fast response time with minimal human input into the system.

Another purpose with this thesis is to make an evaluation of interesting techniques and then make suggestions on further development to improve the accuracy of the matching of ads to web documents. This is so that the system can keep being developed after the completion of this thesis.

2.2 Problem definition

The main problem can be defined as follows.

The main problem is to find techniques that will allow for a scalable, accurate and automatic way of matching a limited number of ads to a website emphasizing the most relevant ads and only requiring human input when adding the ads to the system.

You can divide the main problem into a number of sub problems. We explore this by considering:

- How do you decide what’s important on the website without human input?
- How do you match a website to a limited amount of ads that have a small amount of keywords attached to them that define the ad’s relevance and meaning?
- How do you improve throughput in an application that works both towards the internet as a source as well as a database?

These are questions this thesis is intended to investigate and answer.

The system will communicate towards the internet to download web documents. When a web site is downloaded it should be analyzed and the most important keywords in the document should define what ads are relevant to the web document.

The system also has to handle the problem with duplicate websites with the same content so that they don’t interfere with the ad matching accuracy.
The system will be storing the information it extracts during the analyze phase into a database, this storing procedure has to be efficient to not affect the throughput of the system in a negative way.

### 2.3 Constraints

An obvious constraint is limiting the number of methods evaluated and tested. This thesis is performed by one person and the amount of time is limited.

To get a good evaluation, a good amount of time needs to be spent on each method. Therefore a constraint of maximum 3 test cases was added. The prototype has to be built with .NET technology and the database used has to be MySQL.
3 Problem analysis

3.1 Overview
This main problem this thesis aims at solving is to find techniques that will allow for a scalable, accurate and automatic way of matching a limited number of ads to a website emphasizing the most relevant ads and only requiring human input when adding the ads to the system.

In this chapter the requirements on the system will be presented. First an overview of the system’s intended functionality is presented, and then a number of identified problems are highlighted and discussed.

3.2 System functionality
This is a very simple explanation of how the system is supposed to work.

There’s a database of web document URLs that needs to be matched to a finite number of ads that have a small number of keywords attached to them. The document should be downloaded; the content of the document analyzed and then ads that match the documents content should be associated to the document.

![Figure 2 - System overview](image)

3.3 Identified Problems
In chapter 2, the problem is defined. From this definition and the use case the following problems can be identified.

- Documents are downloaded from the internet which is a slow source. How do you optimize the throughput so that the internet doesn’t become a bottleneck?
- The system is supposed to have a high throughput, be scalable and handle a lot of data. How do you organize the data structures for efficient processing?
- A document contains html-tags and text. How do you automatically identify what the website is about without human interaction?
• How do you match ads with a limited amount of keywords to a document with accuracy and speed?

3.4 Identified requirements

A number of requirements can be identified from chapter 2 and section 3.3. They are the following.

3.4.1 Document downloading

The system will be working with a source that is slow and unreliable, it’s known as the Internet. The document downloading must be parallelized acting as a cache of the documents that are to be analyzed. This is very important to reach the goal of the system being scalable and high throughput. Good error handling is also a requirement as internet sources may become temporarily unavailable.

A batch solution could be considered but the system will be working in an online environment where the new web documents are added to the database continuously and ads needs to be associated to the documents as soon as possible so it’s more suitable with a parallelized downloading solution in real time. The company also required this for the solution to be suitable for the further integration into their products.

3.4.2 Data structures

The data in the system must be structured in such a way that reduces the latencies when data is acquired to make the calculations needed run faster. In memory structures should be favoured over other methods.

3.4.3 Document analysis and ad ranking

The downloaded documents have to be analysed with a method that doesn’t involve human interaction. This is very important for creating a system with high throughput and scalability. A method that can rank ads according to relativity is also needed so that more relevant ads can be shown more frequently.

3.4.4 Duplicate websites

A method has to be used to detect duplicate websites and remove their affect on the automatic document analysis. If this is not done, a big number of duplicate documents may shift the result in the wrong direction by poisoning the systems data that is used to identify what’s important in a document.

3.4.5 Platform

The system has to be developed using .NET techniques and the database used has to be MySQL 5.0 which is the latest stable release of MySQL.
4 Technical background

4.1 Overview

This chapter describes the technology and science on which this thesis is based on. The main field that is investigated is the Information Retrieval area.

First three ranking models are investigated; these are the Boolean, probabilistic and the vector space model. A comparison is made between these models discussing the pros and cons of the models. The ranking models are used to determine the content of a website, this is needed in the web document analyze phase.

Different stemming techniques are also investigated. The stemming technique is considered since it helps increase the number of matches against a limited amount of keywords which is exactly the problem to be solve. Stemming reduces a word to its base form, this means car and cars will be reduced to the same word; car. By applying stemming to all words will reduce the number of possible words and the probability to find matching ads to a web document’s content should increase.

Similarity coefficients are also investigated to enable the possibility to match web documents with ads. A similarity coefficient gives a quantified value on how similar two objects are and you can use the coefficient to find the most similar ads and documents.

Other techniques to improve matching capabilities are also considered; among these are a thesaurus and a technique called query expansion. A thesaurus is a way to improve the accuracy of matching ads to documents by for instance using a table that links similar words to each other. For instance the word car is related to the word automobile and by using a thesaurus, an ad about cars could be matched to documents containing either the word car or automobile.

4.2 Ranking models

These models can be considered to be the classic standard models in the information retrieval area according to [Ribeiro-Yates 1999]. There are other methods available like the fuzzy set model or the extended Boolean model but this investigation will focus on the three standard models.

Some terminology will be used in this section that needs further explanation.

An index word is a word contained within the document that captures the essence of the topic of the document. There are several ways of selecting what terms should be considered index terms, in its base form all words contained within a document maybe be considered to be an index word. You may use a list of stop words that contains the most common words that are present in most document (is, are, for, that etc) and exclude them from the index.
terms since they merely are used to construct sentences and do not capture the essence of the document.

4.2.1 Boolean model

This is the oldest of the three models and very simple method. The index words are given Boolean weights. In the Boolean model a word is either a part of the document or it’s not. The Boolean model is based on set theory. Queries are defined as Boolean expressions which have precise semantics. A query is a search set that is defined by a number of words that describes what is being requested.

The simplicity of the Boolean model has received great attention in the past and was adopted early bibliographic systems.

There are a few major drawbacks with the Boolean model. One drawback is that it’s based on binary decision and therefore has no grading scale (either it’s relevant or it’s not). Another drawback is that you have to write the queries in Boolean expressions which have a precise semantic and therefore it might be hard to express a request with a Boolean expression easily.

In the standard Boolean model there is no such thing as a partial match. Here a more advanced Boolean model has been developed. It’s genially called the extended Boolean model and it features term weighting and partial matching. The extended Boolean model was introduced in 1983.

4.2.2 Vector space model

The vector space model introduces partial matching and the ranking of matching documents which in effect gives more precision in the matching if you compare to the Boolean model.

Definition of the vector space model from [Ribeiro-Yates 1999]

For the vector model, the weight \( w_{i,j} \) associated with a pair \((k_i,d_j)\) is positive and non-binary. Further, the index terms in the query are also weighted. Let \( w_{i,q} \) be the weight associated with the pair \([k_i,q]\), where \( w_{i,q} \geq 0 \). Then, the query vector \( q \) is defined as \( q = (w_{1,q}, w_{2,q}, \ldots, w_{t,q}) \) where \( t \) is the total number of index terms in the system. As before, the vector for a document \( d_j \) is represented by \( d_j = (w_{1,q}, w_{2,q}, \ldots, w_{t,q}) \).

So when defining documents within the vector space, a similarity measure can be evaluated by calculating the correlations between two vectors. Quantifying this correlation can be done with a number of similarity coefficients. More about this later in the chapter.

Since the vector space model works with weight, it doesn’t try to say whether or not a document is relevant, it ranks the documents in accordance to their degree of similarity to the query.

One very important component is how the weights of the terms are calculated. The problem is defined as follows in [Ribeiro-Yates 1999]
Given a collection $C$ of objects and a vague description of a set $A$, the goal of a simple clustering algorithm might be to separate the collection of objects into two sets: a first one which is composed by objects related to the set $A$ and a second one which is composed of objects not related to set.

We have a collection $C$ of object and a set $A$ that is a part of the collection $C$, $A$ needs to be analyzed.

The use of the word vague when referring to the set $A$ comes from the fact that words may not have an exact meaning and can mean two or more different things.

The clustering problem consists of two main issues that have to be resolved.

- What are the features(index terms in our case) that better describe the objects in set $A$? (intra-cluster similarity)
- What features(index terms in our case) better describe objects in set $A$ compared to the rest of the collection $C$? (inter-cluster dissimilarity)

In the vector space model, the intra-cluster similarity is quantified by measuring the raw frequency of a term $k_i$ inside a document $d_j$. This measurement is often referred as the tf-factor which stands for the term frequency factor and gives a measurement of how well a term describes the document content.

The inter-cluster dissimilarity is quantified by measuring the inverse of the frequency of a term $k_i$ among the documents in a collection. This factor is referred to as the inverse document frequency or the idf factor.

There are a number of term weighting schemes using these factors who are trying to balance the effects from intra-cluster similarity and the inter-cluster dissimilarity. Since they use the factors tf and idf they are often referenced to as tf-idf schemes.

The equation for the term frequency:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

Where $n_{i,j}$ is the number of occurrences of the considered term in document $d_j$, and the denominator is the number of occurrences of all terms in document $d_j$.

And the inverse document frequency:

$$idf_i = \log \frac{|D|}{|d_j : t_i \in d_j|}$$
Where |D| is the total number of documents and \(|d_j : t_i \in d_j|\) is the number of documents where the term \(t_i\) appears.

This gives the complete equation:

\[
tf idf_{i,j} = tf_{i,j} \cdot idf_i = \frac{n_{i,j}}{\sum_k n_{k,j}} \cdot \log \frac{|D|}{|d_j : t_i \in d_j|}
\]

The vector space model has been tested and compared with many models but it has always proven itself to be a very fast and reliable method and today it’s a very popular information retrieval model.

### 4.2.3 Probabilistic model

The probabilistic model was introduced in the 70s by Roberston and Sparck Jones. It later became known as the Binary Independence Retrieval model. The model tried to capture the information retrieval problem in the probabilistic domain.

Definition from [Ribeiro-Yates 1999].

For the probabilistic model, the index term weights variables are all binary, i.e., \(w_{i,j} \in \{0,1\}, w_{i,q} \in \{0,1\}\). A query \(q\) is a subset of the index terms. Let \(R\) be the set of documents known (or initially guessed) to be relevant. Let \(\bar{R}\) be the complement of \(R\).

Let \(P(R|d_j)\) be the probability that the document \(d_j\) is relevant to the query \(q\) and \(P(\bar{R}|d_j)\) be the probability that \(d_j\) is non-relevant to \(q\). The similarity \(sim(d_j, q)\) of the document \(d_j\) to the query \(q\) is defined as the ratio.

Starting with the definition of the similarity ratio.

\[
sim(d_j, q) = \frac{P(R|d_j)}{P(\bar{R}|d_j)}
\]

Using Bayes’ rule, assuming independence between index terms and making some small estimations the similarity can be written this way.

\[
sim(d_j, q) \sim \sum_{i=1}^{t} w_{i,q} \times w_{i,j} \times \left( \log \frac{P(k_i|R)}{1 - P(k_i|R)} + \log \frac{1 - P(k_i|\bar{R})}{P(k_i|\bar{R})} \right)
\]

This is the basis of the probabilistic model.
The set $R$ is unknown at the beginning so values have to be estimated at the beginning and the common values used are.

$$P(k_i|R) = 0.5$$

$$P(k_i|R) = \frac{n_i}{N}$$

After the initial documents have been retrieved, the values can be improved by using the following equations.

$$P(k_i|R) = \frac{V_i + 0.5}{V + 1}$$

$$P(k_i|R) = \frac{n_i - V_i + 0.5}{N - V + 1}$$

Where $V$ is subset of the documents initially retrieved. $N$ is total number of documents in the collection. $n_i$ is the number of documents containing index term $k_i$.

4.2.4 Comparison of the models

The Boolean model is considered to be the weakest model of the three above. It has been demonstrated to be ineffective by [Salton 1989]. The lack of partial matching really hurts the performance and reduces the areas of use for the Boolean model.

Comparing the vector space model and the probabilistic model is a bit harder. They both support the same type of functionality. The vector space model is considered to be a bit simpler to implement. [Ribeiro-Yates 1999]

The general consensus among researches seems to consider the vector space model being the more competent model according to [Ribeiro-Yates 1999]

4.3 Stemming

4.3.1 Definition

Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form.

4.3.2 Explanation

Written text contains many grammatical meanings in the word. The idea with stemming is to reverse those and transform all words that mean the same thing to their base form.

For instance the word computer might be written as computers if you are talking about more than one computer. The stemming equivalent is the same for computer and computers.

Another case that will be affected by the stemming is the word computing that also will be transformed to the same as computer and computers.
By using stemming the number of words in the collection is reduced and the probability to match an ad’s keywords to a web documents index terms will increase. If an ad has the keyword computer, it will match web documents that contain the index word computer, computers, computing if stemming is being used.

By applying stemming to all words in a web document and the ad keywords the number of possible words will be reduced and the probability to find matching ads to a web document’s content should increase thus giving a better ad matching.

4.3.3 Algorithms

4.3.3.1 Brute force algorithms
In brute force algorithms a look-up table is used to try to find the stem base of a word. This is generally not a very elegant method, it takes a lot of space for storing the all the data, it doesn’t adapt automatically to new words that appear, that is especially common in today’s internet era. [Ribero-Yates 1999]

4.3.3.2 Suffix stripping algorithms
Suffix stripping algorithms work with a set of rules to reduce the words to their stem. [Ribeiro-Yates 1999]

For instance.

- If the word ends with *ing*, that portion of the word is removed.
- If the word ends with *et*, that portion is removed.
- If the word ends with *e*, that portion is removed.

Suffix stripping is generally implemented as a number of rules that are applied on the word trying to reduce the word to its stemming base. The set of rules used is usually language specific so you need to know the language of the document to achieve a good stemming.

Stemming is not a perfect tool, it works with a finite number of rules and in some cases two words that doesn’t have the same meaning are paired together because the way the words are spelled are very similar.

4.3.3.3 Lemmatization algorithms
Lemmatization is a quite complex approach to the problem of determining a stem of a word. You first determining the part of speech of a word, and applying different normalization rules for each part of speech. [Plisson et al. 2004]

The part of speech is first detected prior to attempting to find the root since for some languages, the stemming rules change depending on a word's part of speech.

In some cases the suffix stripping stemming will produce the same result as the lemmatization algorithms, but an example of when this is not the case is the words computes,
computing, computed would be stemmed to comput, but their normalized form is the infinitive of the verb which is compute. [Plisson et al. 2004]

4.3.3.4 Stochastic algorithms
With stochastic algorithms you use probability to identify the root form of a word. Stochastic algorithms learn from a table of root form to inflected form relations to create a probabilistic model. [Bacchin et. al 2005]

The algorithms are often expressed using complex linguistic rules.

4.3.3.5 Mixed algorithms
You can of course combine the algorithms above creating a stemming system that tries to take advantage of the benefits of more than one type of algorithm.

4.4 Thesaurus
A thesaurus is similar to a dictionary, but instead of definitions and pronunciations, it contains synonyms. It can be used to expand the reach of the search times by also matching other words with the same meaning as the word specified in the query. The use of a thesaurus is an important aspect to consider when doing Query Expansion. More about this later in this chapter.

The words in the thesaurus are ordered by semantic relations, this can be meaning, categories and association.

There are generally two types of thesauruses. [jing] That is a manual thesaurus, the most common commercial thesaurus method. There is also an automatic thesaurus method that uses information retrieval techniques to find relations between words. The automatic thesaurus is not as accurate as the manual ones because it’s hard to define rules so that the matching reflects real life.

4.5 Query Expansion
Query expansion is the method of reformulating a query to improve retrieval performance in information retrieval operations. [Billerbeck et al. 2004]

The following methods can be used to expand a query.

- Finding synonyms.
- Finding the base of words by using stemming.
- Fixing spelling errors.
- Re-weighting the terms in the original query.
One important aspect when expanding a query is the accuracy. The expansion has to be made in such a way so that the results are actually improved. Therefore some caution has to be taken when applying the methods above on a query.

4.6 Clustering

The goal of clustering is to determine the intrinsic grouping of a set of unlabeled data. To do this automatically can be a challenge; there has been a lot of research done in this area.

By for instance clustering ads, similar ads can be grouped together. If a web document match one ad, the similar ads in that group of ads that were clustered together can also be good matches for the same website. So clustering can be used to find more ads that match a web document’s content.

4.6.1 Batch and Online

There are two options you have to consider when you are doing clustering. Should the clustering be performed on a batch basis or should it be an online process? The benefits of a batch solution are that you can calculate the clusters at a single point and therefore optimize the data used. The down side is that if the system is dependent on the cluster information for further processing then the system has to wait for the batch clustering to take place.

When looking at online batching, it’s the opposite, it’s harder to optimize since you can work with a bigger set but the data in the system will always be up to date.

When you are doing online clustering you have to consider the following three issues according to [papka 1999].

Partitioning – Can a document belong to more than one cluster. With partitioning each document is assigned to exactly one cluster. This can also be referred to a set-oriented clustering.

Prototyping – When using prototyping a cluster is represented by a vector in the same space as the documents. The vector is an representation of the documents in the cluster. The positive effect from this is that you only have to compare documents to the cluster vector and not to all documents in the cluster.

Comparison – A measurement has to be used to compare a cluster with a document. Two coefficients are discussed in 4.8.

4.6.2 Clustering algorithms

I will describe briefly the most common types of clustering algorithms.

4.6.2.1 Single-pass

As the name implies, the single-pass algorithms only parse every document once. Processing is made sequentially as new documents arrive to the system. You compare the current document with the existing clusters, if no matching clusters are found, that is similarity
reaches over a certain threshold, then a new cluster is created. The complexity of this algorithm is $O(n^2)$. Below is the single-pass algorithm described in [papka 1999].

**Step-by-step description**

- The document are processed serially.
- The representation for the first document becomes the cluster representative of the first cluster.
- Each subsequent document is matched against all cluster representatives existing at its processing time.
- A given document is assigned to one cluster (or more if overlap is allowed) according to some similarity measure.
- When a document is assigned to a cluster, the representative for that cluster is recomputed.
- If a document fails a certain similarity test it becomes the cluster representative of a new cluster.

### 4.6.2.2 Agglomerative

Agglomerative clustering is a hierarchical clustering technique.

**Step-by-step description**

- Calculate all pair wise document-document similarities.
- Place each of the $N$ clusters into its own cluster.
- Form a new cluster by combining the most similar pair of current cluster $i$ and $j$.
- Update the similarity matrix by deleting the rows and columns corresponding to $i$ and $j$.
- Calculate the entries in the row and column corresponding to the new cluster resulting from the merge of $i$ and $j$.
- Repeat step 3, 4 and 5 until one cluster remains.

### 4.6.3 Cluster representation

A cluster can be represented by a number of index terms in a vector. This vector will be called the centroid from now on. The centroid is a representative description of the documents contained within the cluster.
There are three ways of assigning the centroid.

1. The first documents term vector is fixed as the clusters centroid.

2. Recalculate the dendroid every time when assigning a new document to the cluster and base the dendroid on all documents.

3. Do batch updates of the dendroid. For instance every 5, 10 or 100 documents that gets assigned to the cluster. This way a speedup can be achieved since the centroid calculations don’t have to be calculated each time.

The centroid can be binary, that means that the centroid basically says which index words that best describe the documents in the cluster. If you are using the vector space model that is described in 4.2.2 you can store the weights of the index terms as well to create a more accurate representation of the documents.

### 4.7 DUST Detection

DUST means Duplicate URL Same Text and refers to when a resource can be accessed through duplicate URLs.

An example is for instance a website http://forum.site.com and http://www.site.com/forum/ that may point to exactly the same content but the urls are different so if DUST detection is not used then there will be duplicate documents in the document collection.

Other examples are http://www.site.com, http://site.com and http://www.site.com/index.html which all usually contain exactly the same content.

If you have duplicates in the websites you are analyzing the statistics gathered to perform future analyzes will be shifted because a document may be counted multiple times and give an incorrect image on how common a word is. For instance if the vector space model is used this would make the inverse document frequency to be lower and the word would be considered as less important and this would reduce the accuracy of the web document analysis.

#### 4.7.1 DustBuster

Presented by [Bar-Yossef et al. 2007], a technique developed at Google that relies heavily on parsing URL-lists for a website and identifying a set of rules that can be applied to the URLs for canonization. The rules are verified by retrieving a small amount of the documents in the collection and checking if the identified rules apply, if not the rule is marked as a false-positive and not used further for that website.

4.7.2 SimHash

SimHash is a dimensionality-reduction technique used for near-duplicate detection. You obtain a fingerprint for each document. A pair of documents is near duplicate if and only if fingerprints are at most k-bits apart. [Charikar 02]

Simhash has two important properties:

- The fingerprint of a document is a hash of the words in the document.
- Similar documents have similar fingerprints.

[Charikar 02] talks about something called random hyperplane-based hash functions for vectors.

*Picking a random hyperplane amounts to choosing a normally distributed random variable for each dimension. Thus even representing a hash function in this family could require a large number of random bits. However, for n vectors, the hash functions can be chosen by picking O(log2 n) random bits, i.e. we can restrict the random hyperplanes to be in a family of size 2O(log2 n)*

*Using this random hyperplane based hash function, we obtain a hash function family for set similarity, for a slightly different measure of similarity of sets. Suppose sets are represented by their characteristic vectors. Then, applying the above scheme gives a locality sensitive hashing scheme.*

The simhash fingerprint is calculated the following way. You divide the document by the words it contains. Then you index the words so that you create a high dimensional space, basically the high dimensional space has as many dimensions as the number of words that are available in the language.

To create the fingerprint you do the following:

- You keep a f-dimensional vector V, each dimension in the vector is initialized to zero.

The i:th component corresponding to word w could take on the following values:

- 0 if word w isn’t present in the document,
- 1 if the word w is present in the document at least once, or
- k if the word w is present in the document k times, or we want it’s presence to represent k units of significance

In Simhash you have to specify two parameters. The parameters are f – dimension of fingerprint vector and k which is the number of dimensions that two documents that are compared can differ to still be considered similar.
For further reading about Simhash and other interesting hashing techniques in the same area it’s recommended to read [Charikar 02].

4.8 Similarity coefficients

To be able to match web documents with ads a comparison coefficient is needed. Both the document and ad is represented by word. When representing these words in a vector space we can use a coefficient calculation to determine a quantified value of their similarity.

Below are two common coefficients that are described at [yale]. The methods can be used to calculate a similarity coefficient between any type of documents, clusters or ads represented in the same vector space.

4.8.1 Tanimoto coefficient

The Tanimoto coefficient is often used when comparing fingerprints. The use when comparing clusters would be.

\[
\text{sim}(i,j) = \frac{\sum_{k=1}^{L} (w_{ik} \cdot w_{jk})}{\sum_{k=1}^{L} w_{ik} + \sum_{k=1}^{L} w_{jk} - \sum_{k=1}^{L} (w_{ik} \cdot w_{jk})}
\]

Where \(w_{ij}\) represents the vector for the index term \(j\) in vector \(i\). \(L\) represents all unique terms in the vector.

4.8.2 Cosines coefficient

The cosines coefficient is another coefficient that is very commonly used, especially in the information retrieval area when comparing documents using the vector space model.

\[
\text{sim}(i,j) = \frac{\sum_{k=1}^{L} (w_{ik} \cdot w_{jk})}{\sqrt{\sum_{k=1}^{L} w_{ik}^2 \cdot \sum_{k=1}^{L} w_{jk}^2}}
\]

Where \(w_{ij}\) represents the vector for the index term \(j\) in vector \(i\). \(L\) represents all unique terms in the vector.

The cosines coefficient does obey the triangle inequality in contrast of the Tanimoto coefficient so it can be used for mathematical calculations.
5 Design choices

5.1 Overview

Everything in the background theory can of course not be implemented so a number of decisions had to be taken to reduce the amount of work; these decisions are presented in this chapter.

The systems main components are the following.

In phase one the document will be downloaded, this phase will be a multi-threaded solution so that many websites can be downloaded in parallel.

The second phase is the analyze phase. Here the vector space model will be used to analyze the documents content, the vector space model was chosen because it gives the tool of ranking. Stemming will also be applied to help improve web document to ad matching accuracy. To compare web documents with ads the cosines coefficient will be used to give a quantified value of the similarity between an ad and a web document. The good thing here is that the cosines coefficient can be used to rank the ads and only select ads that are a good match.

A cluster technique will be used to find duplicated documents by clustering them together in groups. Clustering will also be used to find near matches and group them together in a tree structure called a Dendrogram. This allows for finding more ads that match a specific web document by looking at web documents that are similar to the current one and see what ads matched those web documents.

The database storage will be built as a separate layer that will optimize the communication towards the database server by grouping data inserts together and using queues to give efficient data flows to and from the database server.

5.2 Vector space model

The vector space model was chosen for the system. The vector space model is described in section 4.2.2. The reason to why the vector space model was chosen is that it’s the only model investigated that could rank how well the ads match and also don’t need human input
to evaluate a document’s content. The Vector Space Model was presented by [Salton 1975] as an automatic index technique.

The vector space model has three very important advantages.

1. It uses a term weighting scheme which improves retrieval performance.
2. Its partial matching allows for finding documents that approximates the query.
3. With the use of a similarity coefficient the similarity between documents can be quantified.

The Boolean model was too simple for the requirements on the system. The Boolean model was used in search engines in the beginning of the Internet era but has been replaced more frequently by more advanced and accurate models like the vector space model.

The probability model needs human input to converge towards an optimal solution and therefore it’s not possible to use that model in this case.

5.3 Duplicate documents

For detection and eliminating of negative effects from duplicate documents a cluster solution was chosen. All documents are assigned to a cluster. The threshold for the cluster matching was set really high so that only duplicate documents will end up in the same cluster. Clustering is described in section 4.6. The clustering solution will be an online clustering solution that utilizes the single pass clustering technique so each document only has to be scanned once which will help satisfy the goal with high throughput in the system for better performance.

All duplicate documents will be clustered together and then ads will be matched to the cluster rather than to the actual document. This gives the advantage that duplicate documents will not poison the word weighting by misrepresenting the document set. Another positive feature is that when duplicate urls are found, a full calculation to match the ads to the web document is not necessary.

5.4 Stemming

Stemming is a good method to use in this case since the limited number of keywords for the ads. By using stemming the number of matches with ad keywords should increase.

I decided to use as suffix stripping stemmer algorithm because it’s a simple way of implementing a stemmer and good algorithms are available. The system requirements are high throughput and scalability and a suffix stripping stemming algorithm is a perfect match with these requirements. The stemmer algorithm found at [snowball] was decided to be implemented.

The suffix stripping stemmer was investigated in the technology background chapter, section 4.3.2.2
A suffix stripping stemmer doesn’t need human input so it also satisfies that requirement of the system.

### 5.5 Cluster dendrogram

All web documents will be clustered together. All clusters will be arranged in a cluster dendrogram according to their similarity.

![Cluster Dendrogram Diagram](image)

This will then be used as a type of query expansion where if the minimum number of ads couldn’t be matched directly to the cluster, it will check the most similar documents to see if there are any ads there that can be used. The idea behind this is that since the ads we are matching with only have a small amount of keywords it might have matched a keyword that’s category describing but aren’t present in the first website so the ad is relevant since the rest of the documents in the same cluster tree contains it.

For instance, if the system is trying to match ads for cluster A but I can’t find enough ads, then the system will check A:s similarity neighbour B for ads. If the system still hasn’t found enough ads it will look in C and D as well since they are the next neighbours in line.

This technique should work well if the clusters that are paired together are closely related. This of course requires a good amount of clusters to select from for the matching to be accurate. More about this is explained in the implementation chapter.

The cluster dendroid generation uses an agglomerative clustering algorithm. Closer information about this type of algorithm is presented in section 4.6.2.2.

### 5.6 Similarity coefficient

The cosines coefficient was selected to be used as the measurement of similarity between clusters, ads, and documents. The method fits the requirements well since it is a relatively simple and straightforward method. It also satisfies the triangle inequality so it can be used
in mathematical equations which are needed since we need to calculate the matching score for an ad’s keywords against a document’s keywords. The cosines coefficient is explained in section 4.8.2

5.7 Threaded solution

A very important requirement is the scalability and high throughput. To achieve this a threaded system was found to be needed. The three main parts will run with 1 or more threads. The document retriever will work with many threads to parallelize so that the effect of the slow source, in our case the internet is reduced.

Three distinct areas of the system can be identified; the retrieving, the analyzing and the storing part of the system.

Retrieving  Analyzing  Storing

These parts have to work in parallel and data should be queued between these three parts for optimal throughput.

5.8 Database layer

A lot of data had to be processed and later stored into the database. Therefore multiple inserts into the database had to be used to minimize the latency in the database interactions. Also all data that is needed for the analyze phase was decided to be cached in memory for fast access. The database layers will work with queues of data that are to be stored in the database engine.
6 Implementation

6.1 Overview

The system prototype was developed using Microsoft Visual Studio and was written in C# as a console program. All database queries were written for MySQL 5.0. The system was built as a multi-threaded application and divided into three distinctive parts.

![Figure 3 - System parts](image)

The implementation of these three parts is explained in detail in this chapter. The three parts will run in their own threads and communicate with each other through common data structures that work like a cache between the layers. It was structured this way to improve the throughput of the system. The different parts are designed so that they can run multiple threads of themselves as well to improve the utilization rate of the resources.

The basics of the system are that it’s intended to download a web document, analyze the document’s content and then match ads to it.

![Complete System Flow Chart](image)

This is a very simple description and there are a lot of calculations that have to be done to reach this. These calculations and optimizations will be described.

6.2 Complete System Flow Chart

The system is visualized by a multi-purpose flow chart where each band tells which part of the system each process belongs to. Each band will be explained in detail under each section.
Figure 4 - System flow chart
6.3 Communication between system parts

There is no direct communication between the three parts of the system. All communication is made through queues and common objects that the database interaction layer fills with data and also stores data from. The reason for this is that any database interactions can be grouped together to increase the throughput if it’s queued.

You can compare it to write-back memory techniques that are used in processors to increase throughput on the limited and slow memory bus.

![Figure 5 - Data Transitions in system](image)

The queues are constructed in such a way that random access is fast. The structures utilize inverted indexes and hash tables for fast access, the focus is on speed so the system was optimized for speed and extra memory use to give fast access to the data was justified to reach that goal.

As you can see in the figure by the arrows, all the caches except the ad lexicon are two-way caches where information are both retrieved and stored into by the system. The ad lexicon is updated only from the admin side and not the system side so it only provides information to the system it does not need to receive any information from the system.
6.4 The retriever

6.4.1 Purpose
The retriever part of the system works with downloading the document that the system is analyzing. This part of the system works as a cache of documents to reduce the latency in the system due to the fact that the Internet is a slow source.

6.4.2 Implementation
The retriever was implemented as a multi-threaded solution. The flow chart below shows the flow for one of those threads. All common sources are protected by locks to be threading safe.

![Retriever flow chart](image)

**Figure 6 - Retriever flow chart**

The flow chart steps:

1. Start.
2. Get document url to parse from the url queue.
3. Download the content, if try is unsuccessful then retry 4 times. If it still doesn’t work then drop that URL and go back to the queue to get a new url to download.
4. If the download is successful a Document object is created with the Documents data attached to it. The document is stored into a Document Cluster that contains all the
documents scanned by the system. At this point a document id is assigned from the lexicon. The object is put in the “To Analyze” queue.

5. If the “To Analyze” queue is full, then the thread sleeps for X seconds. When there’s space in the queue again the process starts again at point 2.

6.5 The Analyzer

6.5.1 Purpose

The analyzer basically does all the work that originates from the technology background. It’s the heart of the system and is the part that does all the ranking and ad matching. The analyser only works towards data in the memory and communicates with the other parts of the system through queues to keep a high throughput.

The internet contains of web documents with a lot of different encodings. The analyzer converts all the websites to Unicode so the system doesn’t have to take this into consideration later in process. The conversion is made by first detecting the encoding used in the web document by looking at the content and then applying a conversion function to each character.

6.5.2 Implementation

6.5.2.1 Flow chart

This flow chart outlines how the steps involved in the analyze process.
6.5.2.2 Flow chart explained
The steps in the flow chart need some further explanation

1. Start.

2. Gets a Document from the To Analyze Queue.

3. The HTML data is fetched from the object and put into a variable.

4. The important parts of the html code are extracted. That includes the Title, all H1, H2, H3, B and A tags. The content of these tags are then emphasized by adding more weight to these to the html data.

5. All HTML encoded chars get translated to regular Unicode chars.

6. Encoding translations occur, all data is translated to Unicode

7. All HTML tags gets removed leaving text separated by blank spaces
8. Group words by splitting the document up at the white-spaces.

9. Iterates through words applying stemming algorithm to each word.

10. Iterate through the words grouping them by the same stemming equivalent. Now we have the term frequencies.

11. Match each word with the lexicon saving the term index of each word together with its term frequency.

12. If any unknown words are found, they are instantly added to the lexicon. Only words longer than 2 characters and shorter than 20 characters are added.

13. Now the TF-IDF is calculated. When the values have been calculated they are normalized.

14. Matching the document against a cluster. This is the systems duplicate document protection so the thresholds are really high. An inverted index where each clusters top 5 index term point to each cluster is used for increased performance. Then the matching clusters are compared with the document using the cosines coefficient. If the best match is above the threshold for cluster matching the document is assigned to the cluster.

15. If no matching cluster is found, a new one is created with the top 40 index words of the document as a centroid.

16. If a new cluster is created then ads will be matched with this cluster using the Ad Lexicon together with the cosines coefficient. The weight value is stored together with the matching ads and sent to the DB Interaction layer for storing.

17. The new cluster is sent to the DB Interaction layer for storing back to the database.

18. End.

The analyzer part of the system is built in a thread safe way so there can be several threads running an analyzer at the same time. By thread safe it means that the access to the data structures are controlled in such a way so that no conflicts appear when multiple threads try to access the same resources at the same time.

6.5.2.3 HTML parser

To extract the extra important parts of the HTML Document regular expressions is used. Regular expressions are also used to strip the entire HTML out of the document data. The HTML parser then splits the data up into an array. The array is then parsed and the unique words are grouped. The HTML parser also applies the stemming algorithm to the words in the web document before calculating the final term frequencies for the documents.

The HTML parser is explained in point 3-10 in the flow chart above.
Important optimizations in the HTML Parser is that the number of scans of the document is reduced by sorting the document’s words by the index terms and then iterating over them so that a single-pass is only needed.

6.5.2.4 Lexicon
The lexicon is implemented using a singleton pattern. The lexicon data is stored in two hash tables, one indexed by Word Text and the other by Word Id. The data is duplicated but the data is not so big in size and the focus of the system was speed and throughput so it’s a good trade-off.

The lexicon is accessed through two methods, it’s either accessed by Word Id or Word text, if the requested word is not found, a new one is created in the lexicon and that object is returned to the user. The new object is then passed to the database interaction layer where it will be stored into the database.

The lexicon is designed in such a way that multiple servers can be running the system and the data generated is communicated to the database and from which the lexicons are updated at predefined frequencies.

6.5.2.5 TF-IDF calculation
With each word entry in the lexicon an inverse document frequency (idf) is stored. This is retrieved and calculated when iterating over the words from the Document. These values are then stored into the Document object for later storing.

The tf-idf calculations are made with the generated inverse document frequencies that are available from the lexicon so for each word in the document the lexicon has to be consulted.

6.5.2.6 Cluster assigning
The cluster assigning unit works with an inverted index for greater speed. The inverted index consists of a hash table indexed by word-id where each entry contains all cluster-ids that have that word-id in the top 5 words. This way a fast retrieval of potential matching cluster can be done without iterating all the clusters. The threshold is set high for it to only cluster duplicate documents together.

![Cluster assigning flow chart](image)

*Figure 8 - Cluster assigning flow chart*
The cluster assigning works against a comparison method that applies the cosines-coefficient on the parameters and then returns the quantified similarity value.

When creating a new cluster the cluster lexicon is used to make the new cluster available in the system and also for storage back to the database.

When a cluster has been found, the cluster id is stored into the Document object and will later be stored back to the database by the database interaction layer.

6.5.2.7 Cluster Dendrogram generation
This part of the system is batch process that parses all the documents that have been analyzed. It iterates all the clusters and groups the clusters together two and two and generates a new cluster centroid for each pair. The new centroids are then parsed and paired together. This is done for 5 times creating the top part of a dendrogram.

6.5.2.8 Ad matching
The ad matching is made with the cosines coefficient; it’s implemented the same way as the cluster matching algorithm but works with a cluster and an ad as input parameters. For each new cluster, the cluster is matched against all ads in the system. The calculations are fast so the time needed for matching is very small.

The ads are ranked by weight. For each matching ad term in the document the documents local tf-idf value for that ad term is added together to form the ranking score.

6.6 DB Interaction
In this section the database interaction layer will be explained. An ER-diagram showing the structure will be presented as well as important optimizations that have been done to the database layer to support a high throughput.

6.6.1 Purpose
The database interaction part of the system works as a layer between the system and the backend database. The purpose of this part in the system is to speed up the communication with the database by the use of caches and grouping of queries.

6.6.2 Table Diagram
The table diagram illustrates the database structure that is used to store the information generated by the system. It also shows the relations between the different tables.
6.6.3 Optimizations

When MySQL is inserting data into a table there’s a lot of work that has to be done before the data is actually entered into the database. By grouping many inserts together, the preparations before an insert can be reused for all those inserts giving a big performance boost. Therefore all inserts that can be grouped together are grouped in the database interaction part of the system.

All data that are accessed in the database are cached in either the database interaction part of the system or in the queues between the parts of the system.
6.6.3.1 Document storage
The document storage is a big task for the database. There can be thousands of words attached to a document and all those word relations have to be stored into the cont_documents_words table. If this would be done one and one it would take several seconds to store each document.

By inserting multiple values each time into the database the time can reduced significantly. The reason to this is that the database spends a lot of time preparing before the actual insert takes place, if you make a multiple insert these preparations only have to be made once and then it can be reused for the rest of the values that needs to be inserted into the database. Without multiple insert the system wouldn’t be able to have a good throughput.

6.6.3.2 Lexicon
The cont_lexicon table is a generated data structure from the data in the cont_documents_words. The reason to why it’s pre-generated is that it takes quite a bit of time to calculate when the system contains a lot of documents and by generating it into a table the data can easily be fetched from the lexicon. If you are running multiple servers all of them can get the lexicon quickly but it only has to be generated once. This leaves the database more time to do other calculations. It also means when connecting another server into the system it doesn’t lock the whole table making all servers having to wait for it to complete its lexicon calculations.

6.6.3.3 Clusters
The cont_clusters, and cont_clusters_words also utilizes multiple inserts to improve throughput. The number of words attached to each cluster is limited to a maximum of 40 so the performance impact when using multiple inserts will not be as great as with the document storage but it is still expected to make a big difference for the overall system throughput.
7 Test and evaluation

7.1 Overview

In this chapter the system evaluation is presented. The test system was used by three employees at Getupdated and three methods were ranked on a scale from 1-5 with 5 being the best score.

Data is also presented on how the ad matching score that is produced using the cosines-coefficient is affected by the use of stemming.

7.2 Methods tested

The following methods will be considered during the test.

- Standard lexicon matching.
- Stemming.
- Stemming + cluster dendrogram was tested with the test group to see if the combined techniques affected the performance of the ad selection in a positive or a negative way.

7.3 Test group

The test group consisted of 3 people that were working at Getupdated Internet Marketing.

7.4 Test data

The test data used is a database of 10 000 websites that are partners with Getupdated. The site ranges from personal blogs to professionally driven portals.

For each test case generation the whole system was cleared. That means all document data, lexicon data and cluster data was removed from the database. Then a completely new scan of all documents was made to build the lexicon.

After this had been done, all data but the lexicon data was cleared and the process repeated again to rank the documents with an accurate lexicon that reflects the document set. The last step is saving the document to ad data and entering it into the test system.

The internet can almost be considered an unlimited set of documents and by scanning 10 000 websites I tried to build a representative lexicon of the Internet for the Swedish language.

This process was done 3 times, one time for each test case.
The ads used in the test were ads of Getupdated’s 150 advertisers in the affiliate network Affiliator. The ads were tagged with keywords by an employee at Getupdated. The only instructions given were “Write keywords separated by commas that describe what the ad is about”. It was done this way so that the person wouldn’t be influenced by my knowledge of the system and work more like a normal advertiser would do.

7.5 The test system

7.5.1 Purpose

The system build had to be tested so a test interface was built. The purpose of the test system is to give a way to quantify the results of the system built.

It works the following way.

- Test users are created in the system and they are sent their login details.
- The test user login to the system and the system shows the test user a web document and below the website 5 ads are shown that have been matched to the document.
- The user rate the ads on a five graded scale with 5 being a perfect match and 1 being a total unrelated match.
- After the user has rated one site a new one is instantly shown.

The method tested is randomly picked and the user has no knowledge of what method is being tested at the moment.

7.5.2 Platform

The test system is like a digital evaluation form and was built to investigate the different techniques that were decided to be implemented.

The test system is web based and built in PHP and works with a MySQL database. The test users login to the system from their own computers directly over the Internet.
7.5.3 Flow chart

<table>
<thead>
<tr>
<th>Database</th>
<th>System</th>
<th>Test User</th>
<th>Test system design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check user information</td>
<td>Check User login</td>
<td>Login screen</td>
<td>Login screen</td>
</tr>
<tr>
<td>Login attempt</td>
<td>Login screen</td>
<td>Load test interface</td>
<td>Load test interface</td>
</tr>
<tr>
<td>Select document</td>
<td>Select document</td>
<td>Send information</td>
<td>Send information</td>
</tr>
<tr>
<td>Retrieve data</td>
<td>Retrieve data</td>
<td>View, edit, and delete</td>
<td>View, edit, and delete</td>
</tr>
<tr>
<td>Submit information</td>
<td>Submit information</td>
<td>Method to test</td>
<td>Method to test</td>
</tr>
<tr>
<td>Test system design</td>
<td>Test system design</td>
<td>Test system design</td>
<td>Test system design</td>
</tr>
</tbody>
</table>

Figure 10 - Test system flow chart

7.5.4 Flow chart explained

The flow chart explained in words:

1. The user logs in to the system
2. If the login is successful the user is sent to the test interface. If the login is not successful the user is sent back to the login screen.
3. The test system randomly selects which method to test.
4. The test system randomly selects which web document that should be rated.

5. The information is sent back to the test interface and the test user is shown the website and the matching ads.

6. The test user rates the ads according to how well they match the content and press “next”.

7. The information is stored back to the database, then jumps to step 3 again.

The test system keeps doing this process until the user logs out from the system. All the data entries are stored with timestamps and user id so that the data can be investigated.

7.5.5 The interface

The test users were given a login and password to the test system and were instructed to rate the ads according to how well they matched the content with 5 being a perfect match and 1 being a total unrelated ad. Below is a picture of what the test users saw during their evaluation.
Figure 11 - The test interface

The test interface is built to focus on the website and the ads and be intuitive. The only instruction other than the text you see in the image above. “Please rate these ads according to how well they match the content” was that the score 5 is a perfect match and 1 is a total unrelated match.

7.5.6 Extracting test results
The test results were stored in the MySQL database; the statistics for the test section were generated by creating suitable SQL-queries for input to the MySQL Database.

7.6 Test results
Here are the results collected during the test phase presented. The results are just presented and any comments on the results are presented in the next chapter. The user scores are displayed for all three methods. The computed stats from the database are to show the difference between the use of stemming and not.
7.6.1 Ad to document accuracy
These are the results collected from the test framework so this is the data that was entered by the test users. 5 is a perfect match, 3 is an ok match and 1 is totally wrong. Below is the total average score grouped by each user listed.

<table>
<thead>
<tr>
<th></th>
<th>User 1</th>
<th>User 2</th>
<th>User 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Standard</strong></td>
<td>3.1</td>
<td>3.0</td>
<td>3.1</td>
</tr>
<tr>
<td><strong>Stemming</strong></td>
<td>3.4</td>
<td>3.5</td>
<td>3.6</td>
</tr>
<tr>
<td><strong>Stemming + Dendrogram</strong></td>
<td>3.2</td>
<td>3.2</td>
<td>3.2</td>
</tr>
</tbody>
</table>

Table 1 - Document to Ad accuracy

7.6.2 Matching ads per document
These values are the average amount of ads that were able to be matched to a document. The stemming + Dendrogram solution is not listed since the method limits the number of ads to 5 per document cluster. The weight is the ad score calculated with cosines-coefficient, 1.0 means 100% match, 0.1 means 10% match. So the higher the weight the better the ad match the documents content.

<table>
<thead>
<tr>
<th></th>
<th>Any weight</th>
<th>weight &gt; 0,02</th>
<th>Weight &gt; 0,1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Standard</strong></td>
<td>10.5010</td>
<td>5.0319</td>
<td>1.7223</td>
</tr>
<tr>
<td><strong>Stemming</strong></td>
<td>14.5685</td>
<td>6.2959</td>
<td>2.0041</td>
</tr>
</tbody>
</table>

Table 2 - Matching ads per document
7.6.3 Avg. Weight score
The average weight score for the ads attached to documents.

<table>
<thead>
<tr>
<th></th>
<th>Avg. Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>0,0198</td>
</tr>
<tr>
<td>Stemming</td>
<td>0,0211</td>
</tr>
</tbody>
</table>

Table 3 - Average weight score

In the test system the users were shown the top 5 ads attached to a document. To be able to compare the test users scoring with the calculated weight the average weight score for the top 5 ads for each document is presented below.

<table>
<thead>
<tr>
<th></th>
<th>Avg. Weight top 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>0,0285</td>
</tr>
<tr>
<td>Stemming</td>
<td>0,0325</td>
</tr>
</tbody>
</table>

Table 4 - Average weight score top 5

7.6.4 System throughput
When running the system on a regular laptop running windows vista a throughput of 5-6 pages per seconds were reached on average. The number of documents per second is of course dependent on the size of the documents.
8 Test Results discussion

8.1 Overview

The test phase had two parts, one was the human test where 3 test persons got the chance to rank the ads and the second part was the computed stats from the database on the data generated by the system.

8.2 Results discussion

8.2.1 Ad to document accuracy

The data collected from the test group shows that the use of stemming gave the best result. The stemming + dendroid did a little better than the standard version where no stemming where used.

The difference between stemming and non-stemming was not as big as one might have expected. After analyzing the ad keywords, many of the ads contained both singular and plural of many words which explains why the difference wasn’t bigger between the two methods.

Another parameter that affects the results in a negative way is the number of ads in the system. When studying the test user’s score it was noticed that there were a lot of high scores and a lot of really low scores. The main reason to this is that for the low scoring sites the site had a content that did not match the ads available and therefore matching ads to the sites was a challenge for the system. By increasing the width of the advertisers a better result should be achieved.

Stemming should give a even better result over the standard version when fewer keywords are being used and the ads are tagged in a less accurate way. Since the ads will be tagged by the advertisers themselves it’s important give good instructions as well as making it as easy as possible to match the ads to related web documents and stemming is a tool that helps satisfy this.

The issue that was at hand is that there is only a small amount of ads available to match with the content and therefore any action that helps to increase matches should be implemented to improve the systems performance. The trade-off between performance and accuracy was very good since a simple suffix stripping algorithm was used.

Also the stemming + dendroid score should be able to be improved significantly by having a higher threshold on when an ad is considered a match to a web document. It was just in a very few cases where the dendroid was actually used because the matching ads for the web document was already reached because common words had been used when tagging the ads with keywords.
8.2.2 Matching ads per document

<table>
<thead>
<tr>
<th></th>
<th>Any weight</th>
<th>weight &gt; 0.02</th>
<th>Weight &gt; 0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>10.5010</td>
<td>5.0319</td>
<td>1.7223</td>
</tr>
<tr>
<td>Stemming</td>
<td>14.5685</td>
<td>6.2959</td>
<td>2.0041</td>
</tr>
</tbody>
</table>

These results are really showing what I was talking about in the previous section. The average number of matched ads to a document is 10. That is a high number and it’s above the 5 that is the maximum number of ads for the dendroid to be used. The reason to why so many ads are considered as matches is that common words has been used as keywords for the ads.

By just requiring at least a 2% match the number of matching ads is halved. When requiring a 10% match it’s reduced by as much as 5 times for the standard version and more than 7 times for the stemmed version. These results illustrate well that the number of matches versus qualified matches is large. This shows that tweaking the thresholds for ad matching can improve the performance and put the dendroid clustering solution to work more often and that should yield a better result in the end.

8.2.3 Avg. Weight score

The average ad matching weight score shows the difference in matching between the standard version and the use of stemming.

<table>
<thead>
<tr>
<th></th>
<th>Avg. Weight</th>
<th>Avg. Weight top 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>0.0198</td>
<td>0.0285</td>
</tr>
<tr>
<td>Stemming</td>
<td>0.0211</td>
<td>0.0325</td>
</tr>
</tbody>
</table>

By just considering the top 5 matching ads the avg. weight increases by approximately 50%, this is a considerable difference and shows that a higher threshold so that less ads are matched against a web document is needed to improve the accuracy of the matching function.
8.2.4 System throughput

The system handles 5-6 websites per second on average on a laptop. This is a good number and the throughput should be able to be increased considerably running on a dedicated server platform with hardware designed for optimal memory processing and good thread handling. The throughput is considered to be good and satisfy the requirements of the system.


9 Conclusions and future work

9.1 Overview

When first presented to the task, the actual functionality of the system seemed like a quite reasonable amount of work. One problem was underestimating some of the structural parts that have taken a lot of time to get working correctly.

The ad matching proved to be efficient, more work can be put into improving the algorithms and heuristics even more to increase the positive performance from the stemming and the cluster dendroid.

The project can be expanded by introducing feedback into the system and the retrieval phase can be moved closer to the end user for even faster response times.

Working more with finding better threshold for the ad matching can improve the accuracy of the matching.

9.2 Conclusions

When starting this work the complexity of the surrounding structures around the theoretical parts to achieve high throughput was underestimated.

Building a scalable system with high throughput is not an easy task, a lot of time has to be spent on analyzing the data flows in the model and it takes a lot of extra time to optimize these flows. A lot of queues and caches were used in the system to increase throughput which took a lot of time to organize, implement and test.

Working with a threaded system makes it harder to debug to find errors. Another challenge was to find a good value for the amount of threads that are supposed to run. If a thread is given access to the processor and it doesn’t have anything to do, the resource is wasted. This problem was solved by creating a thread handling class that checked to see if the running thread actually performed any work during its last run. If not the thread was put out of commission for a little while. An incremental back-off was used to improve the performance of the thread utilization.

The selected vector space model turned out to be a really suitable method for this type of system, the rankings where good when looking at the test users input and with some tweaking the quality should be able to be improved more.

The suffix stripping stemmer did a good work and the simple implementation gave little impact on the performance so the use of a suffix stripping stemmer was definitely a good design choice.
9.3 **Identified areas of improvement**

9.3.1 **Dendrogram generation**
The dendrogram generation should be optimized to minimize the number of scans over the cluster objects. Introducing an inverted index probably would improve the performance significantly. A considerable slow down was detected as the number of cluster objects grew, to keep a good throughput this has to be redesigned.

9.3.2 **Ad matching thresholds**
The ad matching should use higher thresholds to make the ads more related to the content. Right now the ads might be matching on quite common word added by the advertiser and therefore the Query Expansion clustering doesn’t kick in and the ads relevancy drops.

9.4 **Suggestions for further development**

9.4.1 **Website retrieval**
Right now the system downloads the documents; the problem that might occur is that you have to be logged in to a website to see the content. In those cases this retrieving method will not work. A suggestion here is to move over the document retrieval on to the user by using JavaScript to send the information to the system. JavaScript techniques are available for this, it’s used by Google in their AdWords product and Yahoo in their Yahoo Publisher Network for instance.

A way of performing it is hashing the websites URL into a number with JavaScript and then sending the number to the web server. If there’s a matching number in the database, the ads gets served to the user. If no match for the specific number in the database documents data gets transferred to the website and stored directly into the database. This reduces the complexity of the system and makes it more scalable which is one of the requirements and goals of the system.

The solution also satisfies a good response time for the system since the data is actually transferred to the system on the first impression on the specific web document.

9.4.2 **Lexicon and word matching**
The ad matching worked quite well although only a few of the technique available was used. For instance a thesaurus could be introduced to broaden the term matching alongside the cluster dendrogram query expansion.

A thesaurus could be something to investigate further. The Dendrogram works like a thesaurus in a way by expanding the search terms by looking at similar sites but a more direct approach may give better accuracy.
9.4.3 Introducing feedback

The intended use for this system is for it to run in an online environment with the main task of serving ads to websites. An important aspect here is that the system will record clicks, ad viewings of the ads. A feedback mechanism could be a good thing to introduce to check how well the ads match the content and if the ad doesn’t seem to generate any clicks, the ad’s weight could be reduced to show other ads on the website instead. By doing this the dendroid could also be utilized in a more efficient way.
<table>
<thead>
<tr>
<th><strong>10 Glossary</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Affiliate Marketing</strong></td>
</tr>
<tr>
<td><strong>Cache</strong></td>
</tr>
<tr>
<td><strong>Centroid</strong></td>
</tr>
<tr>
<td><strong>Contextual</strong></td>
</tr>
<tr>
<td><strong>Cluster</strong></td>
</tr>
<tr>
<td><strong>Dendrogram</strong></td>
</tr>
<tr>
<td><strong>Display advertising</strong></td>
</tr>
<tr>
<td><strong>Index Term</strong></td>
</tr>
<tr>
<td><strong>Inverted Index</strong></td>
</tr>
<tr>
<td><strong>Singleton pattern</strong></td>
</tr>
<tr>
<td><strong>Thread</strong></td>
</tr>
<tr>
<td><strong>TF-IDF</strong></td>
</tr>
</tbody>
</table>
11 References

11.1 Literature


<table>
<thead>
<tr>
<th>Reference</th>
<th>Details</th>
</tr>
</thead>
</table>

### 11.2 Websites

<table>
<thead>
<tr>
<th>Reference</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>[snowball]</td>
<td><a href="http://snowball.tartarus.org/algorithms/swedish/stemmer.html">http://snowball.tartarus.org/algorithms/swedish/stemmer.html</a></td>
</tr>
<tr>
<td>[yale]</td>
<td>[<a href="http://graphics.med.yale.edu:5080/TriposBookshelf/sybyl(selector(selector_theory5.html)">http://graphics.med.yale.edu:5080/TriposBookshelf/sybyl(selector(selector_theory5.html)</a>](<a href="http://graphics.med.yale.edu:5080/TriposBookshelf/sybyl(selector(selector_theory5.html))">http://graphics.med.yale.edu:5080/TriposBookshelf/sybyl(selector(selector_theory5.html))</a></td>
</tr>
</tbody>
</table>