VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@mcs.vuw.ac.nz

Improvements to Web Page Clustering Methods

Daniel Wayne Crabtree

Supervisors: Peter Andreae, Xiaoying (Sharon) Gao

October 2004

Submitted in partial fulfilment of the requirements for
Bachelor of Science with Honours in Computer Science.

Abstract

Clustering Search Engines provide users with an improved method of searching the internet by dynamically generating a topic oriented filter list. This project investigates ways of improving the current methods of finding clusters within search results and introduces a new method of evaluating clustering search engines.

Acknowledgments

I would like to thank my supervisors Peter Andreae and Xiaoying (Sharon) Gao for their support, ideas and suggestions for relevant avenues of investigation in this project. Unfortunately, there was not time to investigate them all.

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Why cluster web page search results

With current search engines getting ever more powerful, is there any need for a new type of search engine? The answer is a resounding yes. With current search engines, the largest problem is the ordered list of results coupled with the enormous size of the internet. The first few links in any search may not be the ones desired; in fact, the first several hundred or thousand may be on topics completely unrelated to what the user was searching for. The solution with current search engines is to refine the search. This often involves adding more keywords, altering the keywords or using advanced Boolean features. This can be a time consuming process and even after this process, the user may not find the pages they are looking for, as they may still be thousands down the ordered list. A change is called for, a new way of visualizing the result set is required and web search clustering is one way to do this. By dynamically generating a series of clusters that can be used as filters on the result set, a user can very quickly get an overview of the entire result set, the information it contains and can filter to the topic they require with ease.

1.2 How to cluster web page search results

The clusters are formed by partitioning the result set into clusters of pages, where the pages within each cluster are in some way related. The aim is to generate clusters which contain pages about the same topic, thus dynamically partitioning the result set into topics of potential interest to the user. Ideally, the semantic properties of the content of the pages should be used for clustering, but this is intractable. There are many syntactic properties which may indicate that two pages could be considered related: pages may have common words, common phrases, common in-links, common out-links, or even common words or phrases in the in-linking or out-linked pages. The state of the art web technology for search result clustering and the contributions made by this project consider forming these clusters solely using properties such as these, and little to no attempt is made to understand the semantics of the natural language within the pages or to devise the topics based on understanding this.

As shown in figure 1.1, there are two main tasks for web search clustering. The first is to partition the search results into a set of clusters; the second is to generate an accurate

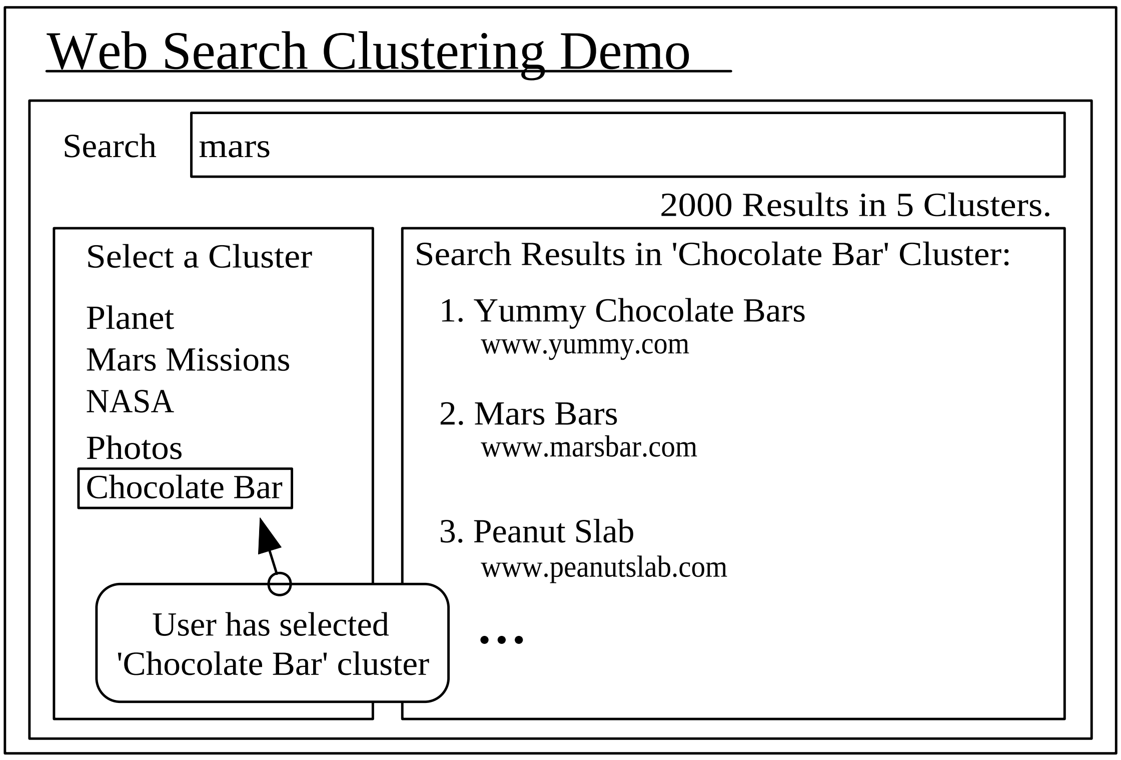


Figure 1.1: Example Clustering Search Engine

description or name for each cluster. Both tasks are important in enabling users to find what they need easily, but this report focuses primarily on the first problem.

1.3 Goals of the Project

Researchers have yet to consider seriously large-scale clustering systems that could be integrated with search systems, so current large-scale clustering implementations use very limited clustering techniques and suffer from very poor quality clusters. Small-scale research systems provide significantly better results, but even these are far from adequate.

The poor results of current techniques stem from a number of problems. Clustering techniques are currently designed to either directly create an appropriate number of clusters or use a rather arbitrary method of choosing which to display to the user. Irrelevant documents from original search results are inadequately identified and persist through to the final clusters. Some clustering methods can join large numbers of closely related clusters in pairwise fashion to form a chain; documents in clusters at one end of the chain may be completely unrelated to those at the other end, thus lowering cluster quality.

One goal of this project is to improve the effectiveness of current clustering techniques. Difficulty in evaluating web clustering results has led to a wide variety of incomparable evaluation methods to be used by researchers. Many of these evaluation methods are flawed or fail to fully evaluate all aspects of the clustering. The subjective nature of evaluating clustering and the human input required has also hindered the evaluation of large data sets and has led to most research considering only small data sets.

A second goal of the project is to provide a unified approach for future research to follow for effectively evaluating web clustering algorithms.

The project has made the following contributions towards the goals:

1. An algorithm that can reduce a large set of clusters to a smaller set containing the clusters of most use to the user.
2. A method and criteria for evaluating and comparing web page clustering algorithms.
3. A reconstruction of one of the standard web page clustering algorithms and an evaluation of its application to clustering both 'short extracts (snippets)' and 'full text' of web pages.
4. Several approaches for improving one of the standard web page clustering techniques.
5. An investigation of an an approach for identifying irrelevant documents that should be excluded from a web clustering.

1.4 Report Organisation

The first section introduces clustering, some of its benefits and other search techniques. Chapter 2 introduces the concept of the clustering search engine and discusses the benefits of a clustering search engine. Chapter 3 considers the result quality and ease of use of existing solutions to the search problem, including directories, search engines and the current generation of clustering search engines.

The second section investigates current methods for clustering and evaluation, identifies problems and provides new algorithms and approaches for solving these problems. Chapter 4 introduces clustering and the different methods available, with a particular focus on the suffix tree clustering method that this project builds on. Several new ideas and approaches for clustering n-gram and suffix tree phrasal information using K-mean and agglomerative clustering algorithms are described, as are several new similarity measures based on the suffix tree base clusters. Chapter 5 introduces the current methods for evaluating web clustering algorithms and identifies problems with them. A set of criteria for web clustering evaluation and a method for carrying out the evaluation are then developed, the method is then proposed as a standard benchmark for future web clustering research.

The third section discusses the implementation of this project. Beginning in chapter 6 the system developed for this project is discussed; chapter 6 differentiates the testing system developed in this project from the production system that would be used in a real application. Chapter 6 also introduces the four main phases of the testing system, the testing interface and the backend database used to facilitate fair testing. Chapter 7 describes the preprocessing phases and discusses how data is obtained and how it is cleaned ready for clustering. Chapter 8 provides details on two of the main contributions of this project: the maximal cluster covering extension to suffix tree clustering and the irrelevant document elimination methods. In addition, the clustering method used in this project is described. Chapter 9 reports of experiments to determine the statistical significance and quality of the new evaluation method by evaluating random data and carrying out an experiment comparing 'snippets' against 'full text' data. The impact of using the maximal cluster coving extension, the effect of each system parameter and the performance of the irrelevant document removal techniques are also evaluated. An extension to the evaluation method was discovered while investigating the effect of early word removal. At the end of chapter 9, the best method from this project is compared against a commercial clustering application.

The final section provides a summary of the work in this project and highlights the main contributions of this work. Chapter 10 discusses directions for future research and builds on the details provided throughout this report to show how these might proceed. Finally, chapter 11 presents the conclusions and key results of this project. An appendix of stop words used in preprocessing, output from clustering, a glossary of terms and references follow.

Chapter 2

Clustering Search Engines

2.1 Background Information

The internet contains a vast amount of information and is growing at an ever increasing rate. Users need a way to quickly search for and find the information they are looking for; currently search engines such as Google [10] provide a means of searching for this information. A typical search provides an ordered list of thousands, if not millions of pages. While the majority of these pages are related to the search keywords, they are predominantly not what the user was looking for. The result set typically contains several distinct topics and the pages the user is looking for often end up deep in or scattered throughout the result set, which due to the ordered nature of the list makes them difficult to find. As an example, when searching for product information, hundreds of pages selling the product appear, but none provide the desired information. Obtaining a result set with just the desired topics requires constructing a query which precisely defines the pages of interest while excluding all others; this can be a time consuming and difficult, if not impossible process.

One solution to this problem is to present the user with a list of the topics included in the search results. This allows the user to immediately narrow the search to only their chosen topic, speeding up their search. To generate the list of topics, we need to separate the pages out into similar groups of documents and this is done by clustering the pages.

Informally, web search clustering is an extension of the ordered list approach currently used by most Internet search engines; web search clustering dynamically generates a set of search filters for a result set by identifying groups of related pages known as clusters. Selecting one of these clusters restricts the result set to only the pages within the selected cluster; with search results often containing thousands of results this enables users to narrow down a search more quickly and provides an overview of the topics within the result set.

2.2 The Benefits of Clustering

Web page search clustering has benefits for all classes of user: beginners, intermediate users and experts will all benefit from the use of a clustering search engine.

Beginners may struggle to get the correct keywords and may not understand all the techniques that can be used to refine their search, they may also often struggle through hundreds of pages of search results looking for pages related to their topic because of this. When using a clustering search engine a beginner even with a relatively simple and typically ineffective query would be able to pull up a large number of resulting documents. The cluster driven filters would then provide the user with an easy way to filter the result set to just the documents they require. Furthermore the cluster names and descriptions help teach the user what type of words to use to search for their chosen topic.

The intermediate user knows how to search effectively and has some limited knowledge of the more advanced search features such as plus, minus and quotes, etc. This type of user can usually find what they are looking for, but it may take a large number of searches to narrow it down and in some cases the user will still not be able to find exactly the pages they are looking for, or they may only find a subset of the appropriate pages. The clustered search saves the intermediate user time, by reducing the number of searches and refinements the user needs to do and by automatically providing the refinements to the user. Also more effectively than by standard techniques, the cluster based filtering filters the result set to only pages related to the topic, which immediately allows for greater coverage of the topic and access to quality information that would otherwise be hidden deep within a result set.

The expert user is similar to the intermediate user, but with much greater skill. The expert user is able to use many of the advanced search features with ease and is able to refine searches reasonably well. The benefits to the expert user follow along the same lines of those of the intermediate user in that the number of refinements needed is hopefully reduced to one saving a considerable amount of time and the end result sets and significantly more topic specific allowing for access to pages that would otherwise be hidden.

There are also other indirect benefits of web search clustering which result from the reduced refinement and data entry. With an ever increasing number of mobile users accessing the internet from cellular phones and PDA's where data entry and typing in particular can be difficult and error prone, it is useful to limit the amount of search refinement required and to provide click-able refinements which can be accessed with a single press. Another indirect benefit is that by reducing the number of pages that need to be accessed, the amount of network traffic required can be reduced; in the case of mobile users, this may save a considerable amount of money in data charges.

2.3 A hypothetical clustering search engine

As an example of a hypothetical search engine with web page search clustering consider figure 2.1. In step one, a user has conducted a search with 'mars' as the search term. The right pane (Search Results) contains the standard web search results, five web pages in this instance. The left pane contains the user selectable clusters which are generated based on the search results. In step two, the user has selected the 'chocolate bar' cluster and the search results on the right have had the documents not belonging to this cluster filtered out.

A feature which is found in table 2.1 is that clusters may be overlapping: the clusters 'Mars Missions' and 'NASA' are overlapping and share document 4 (NASA Missions). Many existing algorithms do not generate overlapping clusters. Although not displayed in this example, it is also possible for clusters to be hierarchical with more detailed sub-clusters available for further refinement once a cluster is selected. Overlapping clusters and hierarchical clusters are not requirements of a clustering search engine, but are quite possibly useful tools

Table 2.1: Ideal Results for each Cluster

Cluster Selected	Displayed Search Results
Planet	1, 3, 4
Mars Missions	3, 4
NASA	4
Photos	1
Chocolate Bar	2, 5

for building clusters of maximal use to the end users.

This project focuses on generating non-hierarchical overlapping clusters. If hierarchical clusters are desired, they can be obtained from the methods of this project by simply applying the methods recursively. Overlapping clusters are desirable since topics and documents naturally overlap; one document may be related to two completely different topics for instance and therefore should feature in at least two different clusters.

2.4 Summary

There is an enormous amount of information on the internet and it is becoming increasingly difficult to provide users with an effective way to search for and find the information the user is looking for. A clustering search engine is one possible solution which can assist users of all skill levels in finding the information desired, can speed up the process of finding the information and can make searching from mobile devices where data entry is hard easier. Clustering search engine results can be characterised by two properties: they can produce hierarchical clusters or flat list clusters, and they can produce overlapping or disjoint clusters.

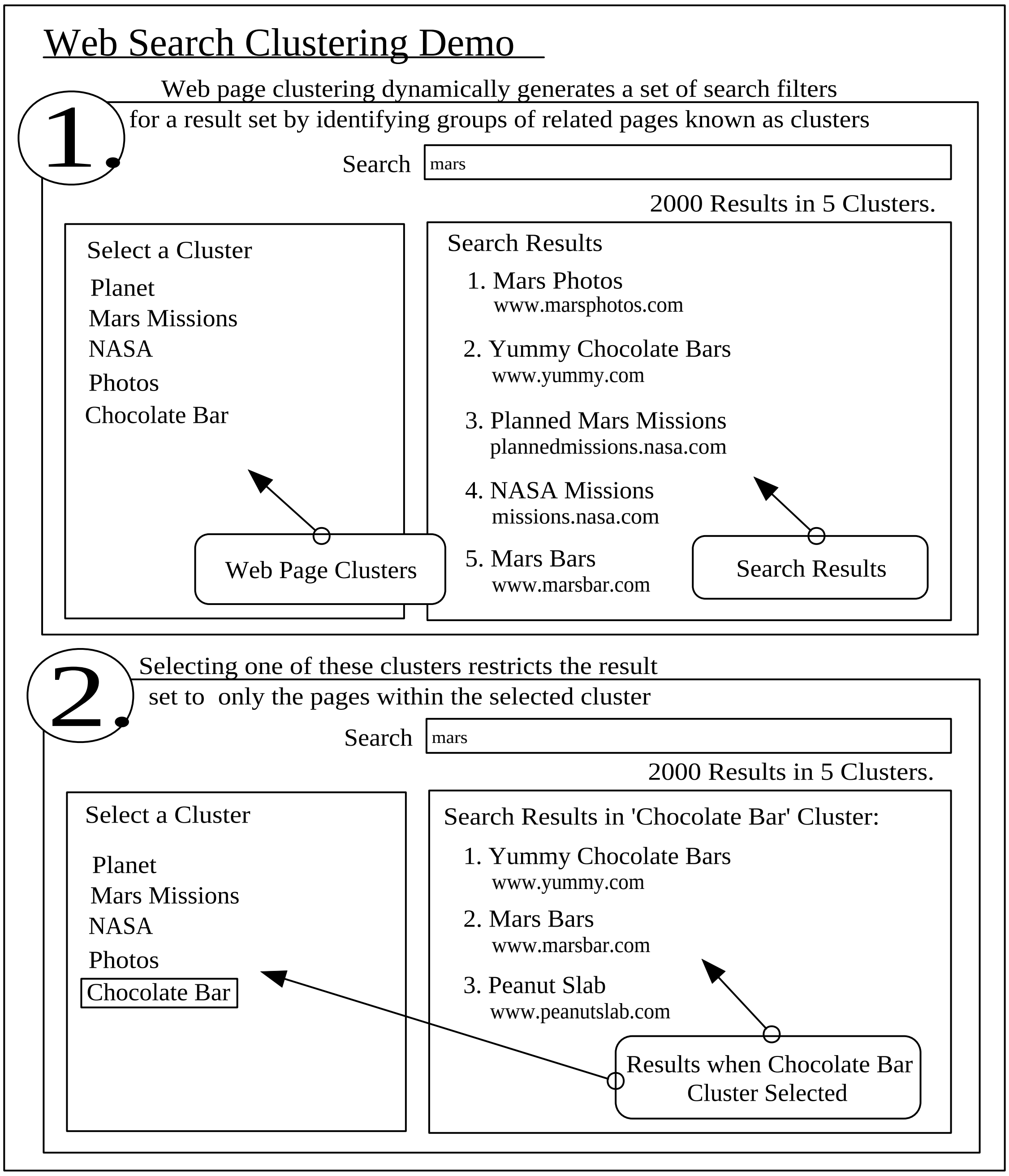


Figure 2.1: Examples Clustering Search Engine - Selecting a cluster

Chapter 3

Existing Web Search Solutions

3.1 Introduction

Chapter 2 discussed web page clustering, the area that this research project focuses on. This chapter steps back and takes a broader look at web search technologies and the current solutions to web search. The different methods (directories, search engines and clustering search engines) are discussed, examples provided and the currently available result quality and ease of use for each are considered.

The primary objective of any search engine is to help the user find the information that they are looking for on the web. Secondary objectives are that the search engine should allow the user to find this information in a time efficient and simple manner, and that users of all experience levels should be able to use the search engine effectively.

3.2 Directories

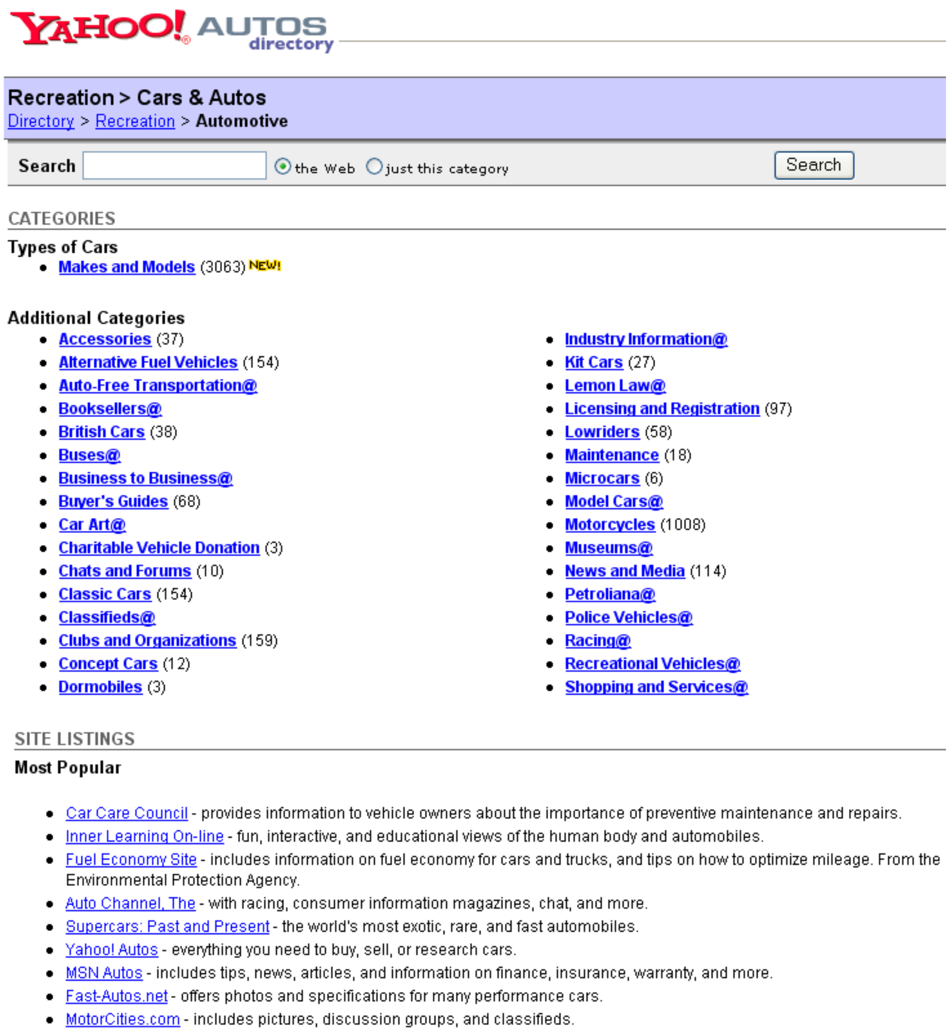
3.2.1 Overview

The most basic category of internet search is the directory. These are human created hierarchies of topics. They typically only include links to the home page of web sites and generally have only a small description, if any, of each site; the category a site is in within the hierarchy indicates what kind of information the website contains. Two of the most well known directories are Yahoo [37] shown in figure 3.1 and dmoz [9] shown in figure 3.2.

Generally, directories contain only a small fraction of the internet: the open directory project (dmoz) contains just over four million links, compared to over four billion in search engines like Google [10].

3.2.2 Result Quality

The web directories are typically of very high precision: because sites are categorised by humans, the websites are placed with near 100% precision. However since only the home pages of sites are listed, the actual category listings may not be a good representation of the



Recreation > Cars & Autos
Directory > Recreation > Automotive

Search [ ] [✓ the Web] just this category Search

CATEGORIES

Types of Cars

- Makes and Models (3063) NEW!

Additional Categories

- Accessories (37)
- Alternative Fuel Vehicles (154)
- Auto-Free Transportation@
- Booksellers@
- British Cars (38)
- Buses@
- Business to Business@
- Buyer's Guides (68)
- Car Art@
- Charitable Vehicle Donation (3)
- Chats and Forums (10)
- Classic Cars (154)
- Classifieds@
- Clubs and Organizations (159)
- Concept Cars (12)
- Dormobiles (3)
- Industry Information@
- Kit Cars (27)
- Lemon Law@
- Licensing and Registration (97)
- Lowriders (58)
- Maintenance (18)
- Microcars (6)
- Model Cars@
- Motorcycles (1008)
- Museums@
- News and Media (114)
- Petroliana@
- Police Vehicles@
- Racing@
- Recreational Vehicles@
- Shopping and Services@

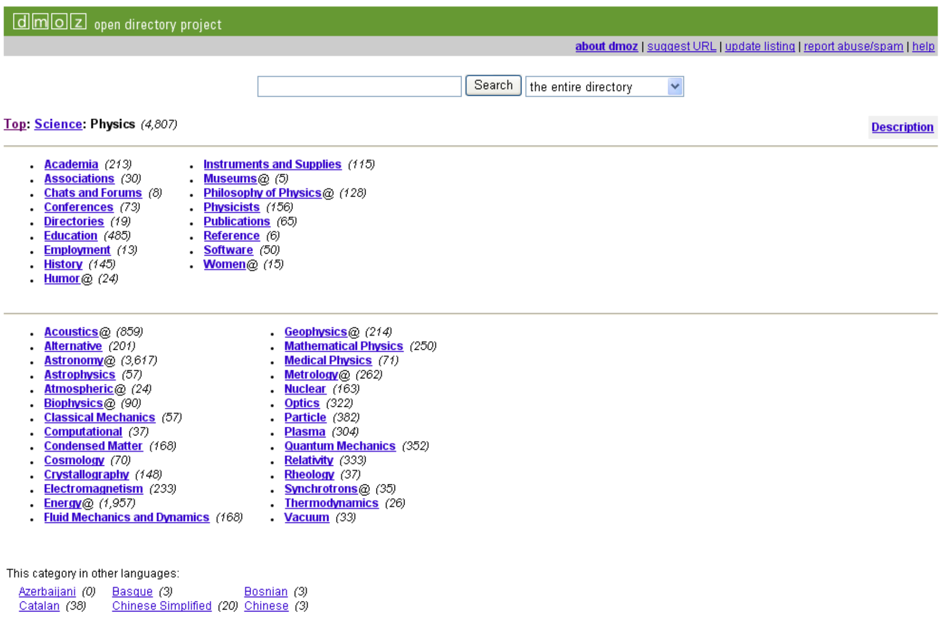
SITE LISTINGS

Most Popular

- Car Care Council - provides information to vehicle owners about the importance of preventive maintenance and repairs.
- Inner Learning On-line - fun, interactive, and educational views of the human body and automobiles.
- Fuel Economy Site - includes information on fuel economy for cars and trucks, and tips on how to optimize mileage. From the Environmental Protection Agency.
- Auto Channel, The - with racing, consumer information magazines, chat, and more.
- Supercars: Past and Present - the world's most exotic, rare, and fast automobiles.
- Yahoo! Autos - everything you need to buy, sell, or research cars.
- MSN Autos - includes tips, news, articles, and information on finance, insurance, warranty, and more.
- Fast-Autos.net - offers photos and specifications for many performance cars.
- MotorCities.com - includes pictures, discussion groups, and classifieds.

Figure 3.1: Yahoo Directory for Cars

site, as sites with a large variety of content may not be placed under all the categories and even if they are, users may not be able to find the content on the target site once they visit it. Directories work best for finding small niche sites which are dedicated to a single topic and they typically allow you to find many of the better sites in these categories. However due to the large amount of human effort required to maintain these directories, their content is lacking. Yahoo, for instance, now charges a significant amount of money per year to be listed; this means that most new sites are not being listed here anymore and the directory is significantly out of date. dmoz is similarly lacking in this regard and is slow to add new sites, if they are ever added at all. When the internet was much smaller, these directories were a good source of information, but as the internet grows, human created directories are becoming obsolete. Perhaps machine generated directories which are updated more regularly and of greater size would make directories more useful for finding the small niche sites in the same way they used to be.


dmoz open directory project
about dmoz | suggest URL | update listing | report abuse/spam | help
Search
the entire directory
Description

Top: Science: Physics (4,807)

- Academia (213)
- Associations (30)
- Chats and Forums (8)
- Conferences (73)
- Directories (19)
- Education (485)
- Employment (13)
- History (145)
- Humor@ (24)
- Instruments and Supplies (115)
- Museums@ (5)
- Philosophy of Physics@ (128)
- Physicists (156)
- Publications (65)
- Reference (6)
- Software (50)
- Women@ (15)

- Acoustics@ (859)
- Alternative (201)
- Astronomy@ (3,617)
- Astrophysics (57)
- Atmospheric@ (24)
- Biophysics@ (90)
- Classical Mechanics (57)
- Computational (37)
- Condensed Matter (168)
- Cosmology (70)
- Crystallography (148)
- Electromagnetism (233)
- Energy@ (1,957)
- Fluid Mechanics and Dynamics (168)
- Geophysics@ (214)
- Mathematical Physics (250)
- Medical Physics (71)
- Metrology@ (262)
- Nuclear (163)
- Optics (322)
- Particle (382)
- Plasma (304)
- Quantum Mechanics (352)
- Relativity (333)
- Rheology (37)
- Synchrotrons@ (35)
- Thermodynamics (26)
- Vacuum (33)

This category in other languages:

Azerbaijani (0) Basque (3) Bosnian (3)
Catalan (38) Chinese Simplified (20) Chinese (3)

Figure 3.2: dmoz Directory for Physics

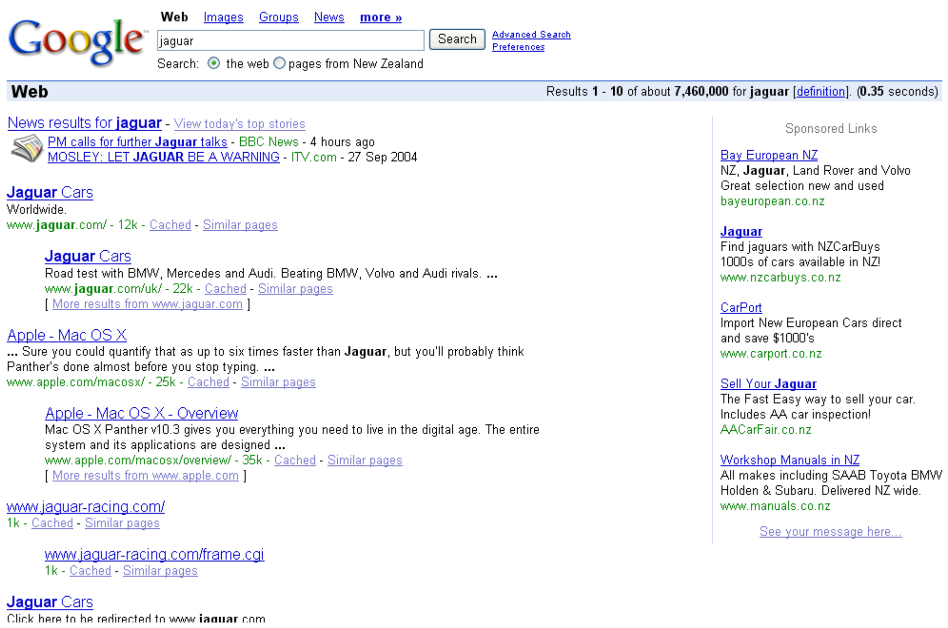
3.2.3 Ease Of Use

Using the directory sites is straightforward: users simply navigate by clicking on links to the topics of their choice. The problem with this is that users have to know where the desired topic is located. Consider the example of finding jaguar car information websites. A user starts at the top level of dmoz. From here, the only category that seems related is the 'shopping/autos' category. But a user may not want to shop for them and so this may seem inappropriate. Clicking on this takes the user to a list of categories related to purchasing car parts and cars. These are not what the user was after and now the user is stuck. It turns out that from the top level the user had to go to 'Recreation', then to 'Autos' from that page. This is probably not a very obvious thing to do, particularly if the user is not interested in cars for recreation or as a hobby. The user can now go to 'Makes and models', then to 'Jaguar' and finally the user has arrived at the sites they desire, although there are not many sites listed – only about twenty in this case. This example shows that directories can make it difficult for a user to find the particular topic they desire. Users need some knowledge about what they are looking for and knowledge about the hierarchy to have any hope of finding what they are looking for. However, it does seem that the worst part is the top level; after delving down deeper, it becomes more obvious where to go next.

3.3 Search Engines

3.3.1 Overview

Search Engines are the most commonly used form of internet search. These respond to user queries with a dynamically generated set of pages which best match the users query. This method allows the user to go directly to pages deep in website page hierarchies. The information presented in search engines also stays relatively current with their content being completely re-indexed typically once every few months. Search Engines are able to keep the information fresh and provide such a detailed level of search by being automated – automatic spiders crawl the internet indexing webpages, which means that there need not be any human involvement with each individual page and that the search indexes can be huge — the larger search engines now include over four billion pages. Two of the most well known search engines are Google [10] shown in figure 3.3 and MSN Search [19] shown in figure 3.4



Web Results **1** - **10** of about **7,460,000** for **jaguar** [definition]. (**0.35** seconds)

News results for **jaguar** - View today's top stories
PM calls for further **Jaguar** talks - BBC News - 4 hours ago
MOSLEY: LET **JAGUAR** BE A WARNING - ITV.com - 27 Sep 2004

Jaguar Cars
Worldwide.
www.**jaguar**.com/ - 12k - Cached - Similar pages

Jaguar Cars
Road test with BMW, Mercedes and Audi. Beating BMW, Volvo and Audi rivals. **...**
www.**jaguar**.com/uk/ - 22k - Cached - Similar pages
[More results from www.jaguar.com]

Apple - Mac OS X
... Sure you could quantify that as up to six times faster than **Jaguar**, but you'll probably think Panther's done almost before you stop typing. **...**
www.apple.com/macosx/ - 25k - Cached - Similar pages

Apple - Mac OS X - Overview
Mac OS X Panther v10.3 gives you everything you need to live in the digital age. The entire system and its applications are designed **...**
www.apple.com/macosx/overview/ - 35k - Cached - Similar pages
[More results from www.apple.com]

www.jaguar-racing.com/
1k - Cached - Similar pages

www.jaguar-racing.com/frame.cgi
1k - Cached - Similar pages

Jaguar Cars
Click here to be redirected to www.**jaguar**.com

Sponsored Links

Bay European NZ
NZ, **Jaguar**, Land Rover and Volvo
Great selection new and used
bayeuropean.co.nz

Jaguar
Find jaguars with NZCarBuys
1000s of cars available in NZ!
www.nzcarbuys.co.nz

CarPort
Import New European Cars direct
and save $1000's
www.carport.co.nz

Sell Your **Jaguar**
The Fast Easy way to sell your car.
Includes AA car inspection!
AACarFair.co.nz

Workshop Manuals in NZ
All makes including SAAB Toyota BMW
Holden & Subaru. Delivered NZ wide.
www.manuals.co.nz

See your message here...

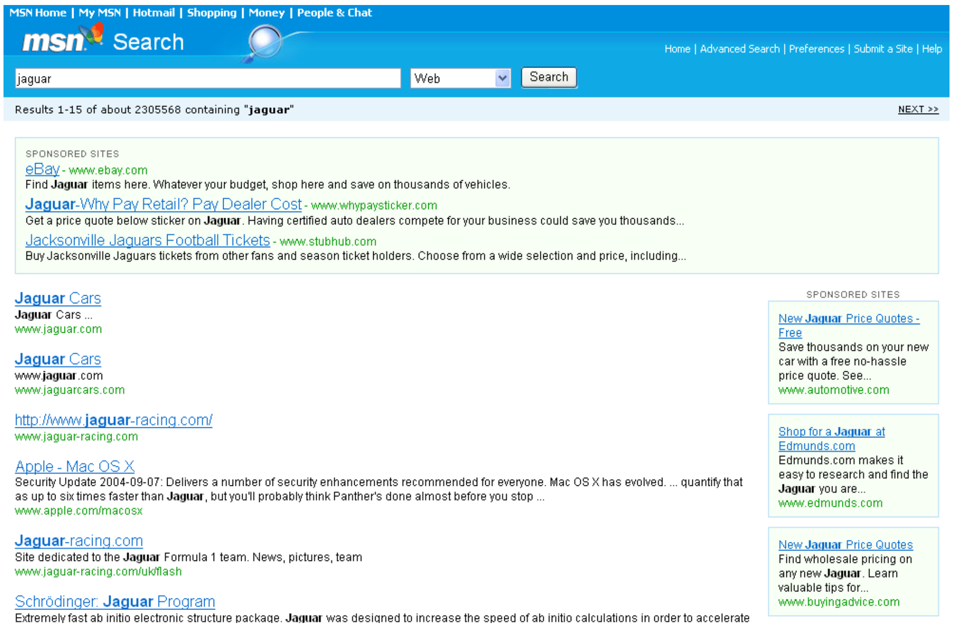
Figure 3.3: Google

3.3.2 Result Quality

Search engine results are often of lower quality than that found in directories. It is common to get quite low precision in the search results with many pages unrelated to the query. This occurs for a number of reasons: broad subject coverage, misrepresented page content and search result ordering.

Broad subject coverage relates to the precision of the query. For instance, 'jaguar' relates to cars, animals, an old games console and a Macintosh operating system. Search Engines are very particular about the search query and keywords used in it; typically, users need to be quite specific. For instance, users looking for information about jaguar cars, would need to enter 'jaguar cars'. Problems arise when users are not sure what they should be searching for or how to classify sub-groupings of a topic. For instance, when searching for information on a product, it is very common to come across a large number of sites selling the product all with very similar information. What keywords must be added to remove these sites and get the useful sites? Because this is difficult, the results are often intermingled with a large number of documents irrelevant to the intended query.

Another source of problems is pages that misrepresent their content: pages may include words that are completely unrelated to their true content in order to get search placement


MSN Home | My MSN | Hotmail | Shopping | Money | People & Chat
msn Search
Home | Advanced Search | Preferences | Submit a Site | Help
jaguar
Web
Search
NEXT >>

Results 1-15 of about 2305568 containing "**jaguar**"

SPONSORED SITES

eBay - www.ebay.com
Find **Jaguar** items here. Whatever your budget, shop here and save on thousands of vehicles.

Jaguar-Why Pay Retail? Pay Dealer Cost - www.whypaysticker.com
Get a price quote below sticker on **Jaguar**. Having certified auto dealers compete for your business could save you thousands...

Jacksonville Jaguars Football Tickets - www.stubhub.com
Buy Jacksonville Jaguars tickets from other fans and season ticket holders. Choose from a wide selection and price, including...

Jaguar Cars
Jaguar Cars ...
www.jaguar.com

Jaguar Cars
www.**jaguar**.com
www.jaguarcars.com

http://www.**jaguar**-racing.com/
www.jaguar-racing.com

Apple - Mac OS X
Security Update 2004-09-07: Delivers a number of security enhancements recommended for everyone. Mac OS X has evolved. ... quantify that as up to six times faster than **Jaguar**, but you'll probably think Panther's done almost before you stop ...
www.apple.com/macosx

Jaguar-racing.com
Site dedicated to the **Jaguar** Formula 1 team. News, pictures, team
www.jaguar-racing.com/uk/flash

Schrödinger: **Jaguar** Program
Extremely fast ab initio electronic structure package. **Jaguar** was designed to increase the speed of ab initio calculations in order to accelerate

SPONSORED SITES

New **Jaguar** Price Quotes - Free
Save thousands on your new car with a free no-hassle price quote. See...
www.automotive.com

Shop for a **Jaguar** at Edmunds.com
Edmunds.com makes it easy to research and find the **Jaguar** you are...
www.edmunds.com

New **Jaguar** Price Quotes
Find wholesale pricing on any new **Jaguar**. Learn valuable tips for...
www.buyingadvice.com

Figure 3.4: MSN Search

under popular searches. For instance, a site relating to Jaguar cars may put other car manufacturer names hidden in its pages so that it will come up in searches for other cars. Modern search engines are attempting to combat these techniques and in particular techniques such as Google's Page Rank which uses in-linking page information to re-order search results. However, there are still erroneous pages occurring in search results, although the number is decreasing as the algorithms improve.

Search result ordering plays a vital role in the quality of the results and intermingling pages of different topics can improve the quality of the results. For example, in a search for 'jaguar', an intermingled result set is better than all the car documents followed by all the animal documents. However, to eliminate unrelated documents and to achieve adequate precision, search refinement is required. While search result re-ordering is useful for many tasks, when the number of pages relating to a query completely outnumbers the pages on the desired topic, finding even one document related to the desired topic is an amazingly difficult task.

There appear to be several flaws and limitations of search engines. Nonetheless, a reasonably skilled user can find the desired pages reasonably quickly and search engines currently provide the best way to find information on the internet.

3.3.3 Ease Of Use

Search Engines are very easy to use but are often hard to use effectively: unlike directories, there is no need to navigate through a complex hierarchy to find the desired categories and pages can be from deep within websites instead of top-level domains. Entering keywords related to the desired pages is straightforward and the results of good queries are typically very good; the difficulty comes in knowing what keywords to use.

Finding the set of keywords that returns only the documents of interest is very difficult and advanced Boolean search features such as word exclusion are often required. Finding words to exclude may require the investigation of unrelated pages. Problems also arise in searching for very specific topics where there are only a few reasonable matches. Finding these is even harder when the user is unsure of their existence and cannot work out how to exclude the irrelevant documents. In some cases, it can be impossible to find the desired pages.

3.4 Clustering Search Engines

3.4.1 Overview

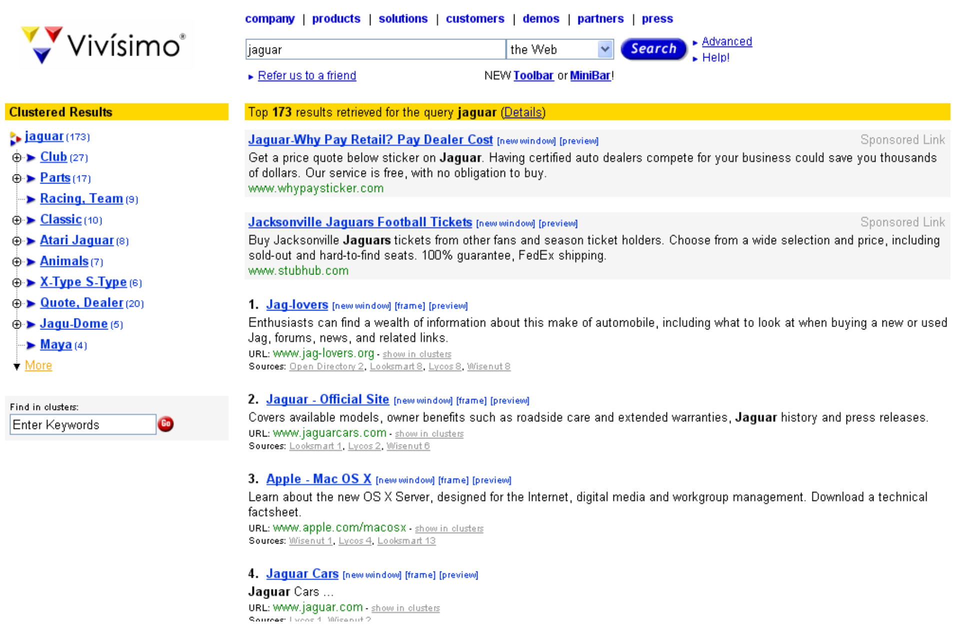
Clustering Search Engines provide a mix between the search engine and the directory. The actual implementations vary, but the idea is that they provide the user with standard search results, as would arrive from a search engine, along with machine created groupings of pages known as clusters which behave in a way similar to directories. These clusters would be oriented around specific topics. In the jaguar search, this may mean clusters such as cars, animals or the macintosh OS, but it could also mean clusters on smaller sub-topics like car parts or car clubs. The user can simply select the specific topic or type of pages that they desire from the list of machine created clusters, which would present the subset of documents in that cluster and possibly a new set of more specific clusters to allow for further refinement. The search may also optionally gather additional pages relating to the more specific search if these were not originally in the clusters.

Almost every search engine other than Google seems to be creating some form of clustering now, whether it is in the form of search refinements, suggestions or clusters. Yahoo [37], Teoma [4], Altavista [2], WiseNut [16], Vivisimo [29], Dogpile [15], Infonetware [14] and Queryserver [21] are some of the internet search engines which provide some form of clustering. Vivisimo and Infonetware (shown in figures 3.5 and 3.6 respectively) are two of the most cluster-centric search engines from that list. Another slightly different variation on the clustering search engine is the cluster oriented search engine, where the user is presented with the clusters as the main object and drilling down is required to get to the results. An example application-based search along these lines is Grokker [11] shown in figure 3.7.

3.4.2 Result Quality

In the ideal situation, the clustering search engine solves most of the problems of the standard search engine. The problem of broad subject coverage is eliminated as the user can easily filter the search results to just the results they desire by selecting the appropriate cluster. The search result ordering problem is fixed once an appropriate cluster has been selected. With a cluster selected, the only documents displayed are those relating to the desired topic. Hence, sorting the results by quality and possibly distinctiveness of the pages is all that is required, which is something the current search engines already do quite well in attempting to solve the result ordering problems. Misrepresented page content is harder to solve and straight clustering of page content will still be misled by fake information and misclassify the document. External sources of information such as link analysis are required to resolve this problem.

The reality is that the cluster quality provided by current solutions is extremely poor. The underlying problem with clustering web search engines in general is that clustering a large



Clustered Results

- jaguar (173)
  - Club (27)
  - Parts (17)
  - Racing, Team (9)
  - Classic (10)
  - Atari Jaguar (8)
  - Animals (7)
  - X-Type S-Type (6)
  - Quote, Dealer (20)
  - Jagu-Dome (5)
  - Maya (4)
- More

Find in clusters:
Enter Keywords

Top **173** results retrieved for the query **jaguar** (Details)

Jaguar-Why Pay Retail? Pay Dealer Cost [new window] [preview] Sponsored Link
Get a price quote below sticker on **Jaguar**. Having certified auto dealers compete for your business could save you thousands of dollars. Our service is free, with no obligation to buy.
www.whypaysticker.com

Jacksonville Jaguars Football Tickets [new window] [preview] Sponsored Link
Buy Jacksonville **Jaguars** tickets from other fans and season ticket holders. Choose from a wide selection and price, including sold-out and hard-to-find seats. 100% guarantee, FedEx shipping.
www.stubhub.com

1. Jag-lovers [new window] [frame] [preview]
Enthusiasts can find a wealth of information about this make of automobile, including what to look at when buying a new or used Jag, forums, news, and related links.
URL: www.jag-lovers.org - show in clusters
Sources: Open Directory 2, Looksmart 8, Lycos 8, Wisenut 8

2. Jaguar - Official Site [new window] [frame] [preview]
Covers available models, owner benefits such as roadside care and extended warranties, **Jaguar** history and press releases.
URL: www.jaguarcars.com - show in clusters
Sources: Looksmart 1, Lycos 2, Wisenut 6

3. Apple - Mac OS X [new window] [frame] [preview]
Learn about the new OS X Server, designed for the Internet, digital media and workgroup management. Download a technical factsheet.
URL: www.apple.com/macosx - show in clusters
Sources: Wisenut 1, Lycos 4, Looksmart 13

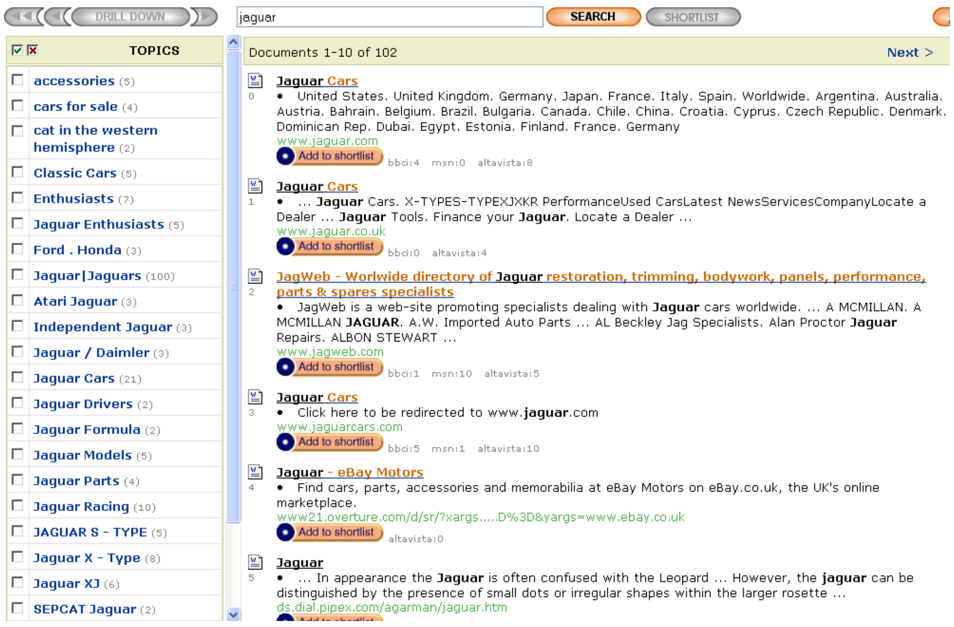
4. Jaguar Cars [new window] [frame] [preview]
Jaguar Cars ...
URL: www.jaguar.com - show in clusters
Sources: Lycos 1, Wisenut 2

Figure 3.5: Vivisimo

number of documents on demand is resource intensive and slow. The strategy to deal with this complexity in current search engines varies: the general idea is to either reduce the problem complexity (Yahoo, Teoma, Altavista) or reduce the problem size (WiseNut, Vivisimo, Dogpile, Infonetware, Query Server).

The approach used by most search engines that reduce the problem complexity appears to be to use heuristics to generate potential cluster names, then provide these as search suggestions. Typically these search suggestions will reduce the number of results, but the results are still just another search and suffer from similar problems to that of the search engine. The clusters can also be of poor quality; Yahoo and Altavista (which both appear to use the same clustering algorithm and present the same clusters) use some form of closeness heuristic. They seem to take the keyword in the original search, together with either the closest noun before or after the keyword in the pages found, then the list of such keywords is ranked and presented as potential new searches. Their technique is quite appalling as the technique is clearly run on only a handful of documents and appears to be heavily biased towards the most frequent documents. It is likely the method uses the frequency of terms to order them before presenting a small subset to the user. Hence, on a search for 'jaguar', only filters relating to cars are listed; filters related to animals do not appear until about position twenty-five on the full search filter list. Teoma appears to do something very similar although the quality of their results is somewhat better: in the jaguar search three of the four main topics had some filters within the 14 filters presented; the only main topic not to appear was the Macintosh OS.

The clustering search engines that reduce the problem size and use more complex clustering algorithms produce better clusters, but instead of returning millions of documents to a typical query, the clusters usually contain a total of one hundred to two hundred documents. This means that the resulting clusters often end up with very small sets of fewer than ten documents in many cases. Without a full analysis, it is hard to know exactly what type of clustering each is performing, but it would seem that these clustering search engines



accessories (5)
cars for sale (4)
cat in the western hemisphere (2)
Classic Cars (5)
Enthusiasts (7)
Jaguar Enthusiasts (5)
Ford . Honda (3)
Jaguar|Jaguars (100)
Atari Jaguar (3)
Independent Jaguar (3)
Jaguar / Daimler (3)
Jaguar Cars (21)
Jaguar Drivers (2)
Jaguar Formula (2)
Jaguar Models (5)
Jaguar Parts (4)
Jaguar Racing (10)
JAGUAR S - TYPE (5)
Jaguar X - Type (8)
Jaguar XJ (6)
SEPCAT Jaguar (2)

Documents 1-10 of 102

0 **Jaguar** Cars
- United States. United Kingdom. Germany. Japan. France. Italy. Spain. Worldwide. Argentina. Australia. Austria. Bahrain. Belgium. Brazil. Bulgaria. Canada. Chile. China. Croatia. Cyprus. Czech Republic. Denmark. Dominican Rep. Dubai. Egypt. Estonia. Finland. France. Germany

www.jaguar.com
Add to shortlist bbci:4 msn:0 altavista:8

1 **Jaguar** Cars
- ... **Jaguar** Cars. X-TYPES-TYPEXJXKR PerformanceUsed CarsLatest NewsServicesCompanyLocate a Dealer ... **Jaguar** Tools. Finance your **Jaguar**. Locate a Dealer ...

www.jaguar.co.uk
Add to shortlist bbci:0 altavista:4

2 JagWeb - Worlwide directory of **Jaguar** restoration, trimming, bodywork, panels, performance, parts & spares specialists
- JagWeb is a web-site promoting specialists dealing with **Jaguar** cars worldwide. ... A MCMILLAN. A MCMILLAN **JAGUAR**. A.W. Imported Auto Parts ... AL Beckley Jag Specialists. Alan Proctor **Jaguar** Repairs. ALBON STEWART ...

www.jagweb.com
Add to shortlist bbci:1 msn:10 altavista:5

3 **Jaguar** Cars
- Click here to be redirected to www.**jaguar**.com

www.jaguarcars.com
Add to shortlist bbci:5 msn:1 altavista:10

4 **Jaguar** - eBay Motors
- Find cars, parts, accessories and memorabilia at eBay Motors on eBay.co.uk, the UK's online marketplace.

www21.overture.com/d/sr/?xargs.....D%3D&yargs=www.ebay.co.uk
Add to shortlist altavista:0

5 **Jaguar**
- ... In appearance the **Jaguar** is often confused with the Leopard ... However, the **jaguar** can be distinguished by the presence of small dots or irregular shapes within the larger rosette ...

ds.dial.pipex.com/agarman/jaguar.htm

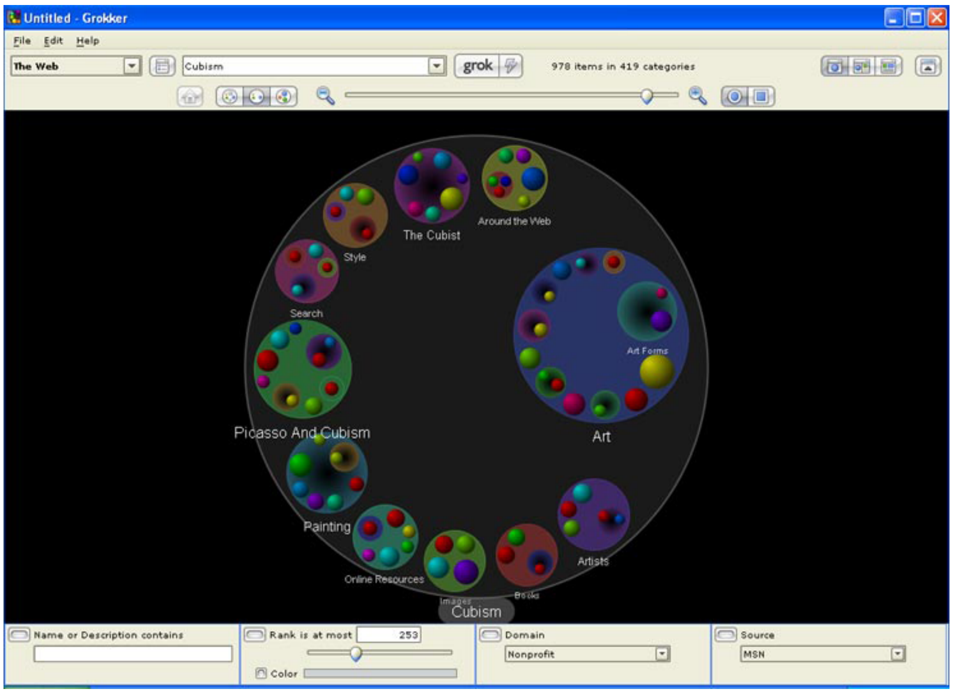
Figure 3.6: Infonetware

are picking out clusters of documents that contain a few frequent words or phrases in common. The small resulting size is particularly prevalent when limited information is used to perform clustering and the resulting clusters often include many clusters that have the same topic, but are listed separately. In the ideal situation, clusters with the same topic but different keywords should be joined together.

The final type of clustering search engine is the cluster oriented search; Grokker is one of these. Instead of providing the user with a listing of search results and a listing of clusters to filter on, a cluster oriented search displays the clusters directly and as the user clicks on these, the clusters are expanded and the documents or sub-clusters within the cluster are displayed. A detailed comparison and study of Grokker was carried out to compare the effectiveness of a real system to the results obtained in our system and the full discussion of this will be provided along with the results in a later chapter. The results showed that Grokker's clusters were of much greater value than those provided on any of the web based clustering search engines. This should be expected given that Grokker is an application that runs on the user's local machine and that searches routinely took fifteen seconds on the test machine. It was observed that the clusters produced appeared to be very useful and the clusters were quite accurate. However, using our evaluation methods, the problem of small clusters not being appropriately joined was evident and reflected in the fact that approximately half the documents ended up in the top level 'other' category when the majority of these should have been joined into other top level categories.

3.4.3 Ease Of Use

The clustering search engine makes search easier for the user, particularly the selection of keywords. The selection of any relevant keywords should result in documents related to the desired topic somewhere in the result set; by selecting a cluster that best matches the desired topic, the results narrow to the desired topic; eliminating the time intensive process



The Web

Cubism

978 items in 419 categories

Name or Description contains

Rank is at most 253

Color

Domain Nonprofit

Source MSN

Figure 3.7: Grokker

of selecting keywords and using advanced search features.

Clustering gives beginning users, who are unsure of keyword selection or search refinement, the ability to find information they would not otherwise have found. Clustering gives advanced users assistance by reducing the time required to narrow searches and also potentially improving the number and quality of the results. Clustering gives all users the benefits of topical filtering not possible with current search engines and requires less typing; which is of great benefit to internet users on mobile devices such as cell phones or PDA's where data entry is difficult.

Current clustering search engines do not work adequately. The major drawback is the quality of the resulting clusters and the scope of the clustering. If the quality and the scope of the clustering could be increased, then the benefits and ease of use associated with clustering search engines could be realized.

3.5 Summary

By combining the benefits of directories with search engines and addressing the limitations of each, clustering search engines can provide a solution to many of the difficulties in getting quality search results. However the current applications fail to achieve this. Some applications cluster on a large scale but use low quality heuristics which provide very poor clusters and then use these to conduct new searches. This search suggestion technique does not appear to be particularly effective. Other clustering search engines use reduced document sets and fare better, but are let down by multiple small clusters on the same topic and by the small size of the clustering result sets.

If the problems of joining the small clusters on the small scale clustering search engines could be addressed and their size scaled up to work with much larger document sets, then the real benefits of clustering search engines could be realized.

Chapter 4

Clustering Techniques

4.1 Introduction

This chapter covers the background information on clustering and the basic ideas, concepts and choices made to select an appropriate clustering algorithm. Firstly, the kinds of information that is potentially available to aid in clustering documents is described. Then the properties that different clustering algorithms can have are discussed. Following that, an explanation of the suffix tree clustering algorithm [39] on which this project builds is given. Then to complete this chapter, several of the clustering techniques considered for use in this project are introduced.

4.2 Information

There are two main kinds of information available for clustering web documents: Textual information and Link Information. Other less frequently used information such as image information [3] can also assist in clustering web pages.

4.2.1 Textual Information

Textual information is the raw data contained in the pages; this may be in the form of individual words, n-grams (phrases up to length n) or phrases of arbitrary length. Textual information can be found in many sources: it may occur in the page directly as plain text or it may occur as hidden text associated with alt text of images, meta tags such as keywords or page description, the page title and it can occur in the URL.

Textual information may come from the page or alternatively from the search engine snippet. A snippet is a small extract of text from the document used as a summary or description of the document by a search engine. The drawback of using only the snippet information is that these may not be representative of the page. In some cases, search engines provide context sensitive snippets that relate to the text surrounding the terms used in the query. In other cases, the Meta description is used. As described in an earlier chapter this can lead to the problem of misrepresentative content as some web pages may use false information. In other cases there may be no snippet provided for certain pages. The obvious trade off is that snippets should be faster to deal with as they are shorter than the full page text, but the resulting clusters should be worse as there is less data.

The decision on whether to use single words, n-grams or phrases is again a trade off: using single words is the fastest and most memory efficient, while using arbitrary length phrases can be expensive in both processing time and memory requirements. [39] shows that using phrase based methods such as suffix tree clustering is more effective than single word based methods, so it may seem that using more information does increase the result quality as expected. But what is surprising is the results in [38], which show that by just using 2-grams (phrases of length one or two), the results are almost as good as using phrases of arbitrary length. What is even more surprising is that 3-grams are actually more effective than using full text in most cases. Therefore, it is clear that using a 2 or 3-gram approach is optimal in terms of the quality / speed trade off.

Another way to extend the textual information is to add related words and semantically equivalent words, particularly higher level descriptive words. For instance, the phrase 'motor vehicle' can be added to documents that contain one of 'car', 'truck', 'van' or 'jeep'. Also in documents where semantically equivalent words exist, the semantically equivalent words could be added, for instance, adding the word 'car' in documents with the word automobile.

4.2.2 Link Information

Link information is information obtained from the hyperlink structure between pages. There are two kinds of direct link information: information from in-linking pages (pages that link to the page to be represented) and information from out-linked pages (pages which are linked to by the page to be represented). The use of more distant information from pages that extend beyond the immediate neighbours (direct in-linking or out-linked pages) is also possible.

Many techniques use link information. As discussed by Wang and Kitsuregawa in [32], [30] and [33] and mentioned in [27], one technique is to use commonalities between shared in-links and shared out-links. Co-citation measures the number of common out-links, coupling measures the number of common in-links. A new technique considered for use in this project is to use the text on the in-links, the text surrounding the in-links, a summary of the in-linking or out-linked pages or the full text of the in-linking or out-linked pages to extend the textual information provided from the page itself. There is a wide range of methods of using link information and further methods are introduced in [12], [13], [34], [35] and [36].

As demonstrated in [32], combining link information with textual information is effective and can provide confirmation of the clusters determined by standard textual information. Link information is also useful for removing outliers or identifying better clusters and can thus improve the overall clustering results.

4.2.3 This Project

In this project, textual information is used: the clustering algorithm uses all phrases of arbitrary length shared by two or more documents. Link information was considered carefully throughout the project, but the overhead of the approximately forty-fold increase in page downloads and processing time proved too large to allow the investigation of this within the limited time constraints of this project. Had the performance of short n-grams been found earlier, these would most likely have been investigated instead.

4.3 Clustering Properties and Proposed Applications

4.3.1 Pre-Retrieval vs Post-Retrieval

Pre-retrieval clustering, also known as off-line clustering, works by clustering the entire document collection before conducting any search. Post-retrieval clustering, also known as on-line clustering, works by clustering the result set returned from a query.

There are two main problems with pre-retrieval clustering: effective clustering algorithms are typically of quadratic or cubic order and thus for enormous collections such as the web with most large search engines currently handling over four billion documents these methods are simply not tractable. The second problem is that the resulting clusters from pre-retrieval clustering are of lower quality than those created using post-retrieval methods, because when clusters are generated post-retrieval, the clusters can take into account the reduced document set and clusters can better segment this set. With pre-retrieval clusters, similar results are obtainable, but an enormous number of overlapping clusters would be required and the set of clusters would need to form an enormous hierarchical clustering.

In spite of the costs, there are still uses for pre-retrieval clustering, such as a directory type service constructed automatically using pre-retrieval clustering. With directories having significantly reduced scope (typically less than ten million pages and covering domains rather than pages), the information from all pages on a site could be combined to summarise a high-level description of the site. These could then be clustered into a hierarchical directory structure dynamically, possibly with some user direction for the top one or two levels of categories.

4.3.2 Overlapping vs Disjoint

Two clusters are overlapping if any document is common to both clusters. Conversely, two clusters are disjoint if there is no document common to both of them. Alternatively, if we view clusters in terms of sets, two clusters are overlapping when their intersection is non-empty and are disjoint when their intersection is empty.

Some clustering methods create overlapping clusters, while other methods create disjoint clusters. With web pages in mind and the topics they may cover, we can see that there must be some overlap in clusters. For instance in a search for 'jaguar', there may be pages comparing the performance of 'jaguar cars' to 'jaguar animals'. These pages have two topics and would need to be in at least two clusters, creating overlapping clusters. Users will thus expect there to be some degree of overlap between clusters.

Although some overlap is required to represent all documents accurately, we want to construct clusters in such a way that overlap is minimised. This is because if two clusters are very similar, then the user will be wasting a lot of their time if they have to look in both clusters, as they will have to consider many documents a second time.

4.3.3 Hierarchical vs Flat

Clusters are hierarchical if they are presented in such a way that one cluster is represented as a subset of another cluster. Hierarchical clustering enables a user to drill down, while flat structures consist of just a list of clusters.

Having hierarchical clusters is not a requirement for a web page search clustering algorithm and many existing algorithms do not generate hierarchical clusters and instead present flat lists of clusters. Take as an example, a search for jaguar. This could have high level categories like 'car' and 'animal'; but it could also have more specific topics like 'race car', 'car parts', and 'car clubs'. These could be placed as subsets of a car cluster and hence a hierarchical clustering is formed. Alternatively they could simply be placed in a flat structure all at the top level. Due to limits placed on the number of clusters displayed to a user at once, this may mean that not as many clusters can be shown. The limits are typically set to about seven to ten clusters, to keep the choice manageable for the user.

Hierarchical clusters could be formed in two ways; the original clustering of the search results may naturally lead to hierarchical clusters, or a flat list of clusters could be recursively clustered to form a hierarchy of arbitrary depth.

4.3.4 This Project

This project uses a post-retrieval, overlapping and non-hierarchical method to cluster the documents. Post-retrieval clustering was essential as the limited time and bandwidth constraints would not allow for the construction of a full search engine. Techniques that generate overlapping clusters were used as internet documents are naturally overlapping and non-hierarchical methods were used as these are easier to evaluate and can always be made hierarchical by recursive application.

4.4 Suffix Tree Clustering

The suffix tree clustering (STC) method [39] [38] is a web page clustering algorithm that uses the phrases shared in common between documents to generate base clusters. Base clusters are sets of documents with at least one phrase in common and each base cluster corresponds to a node on the suffix tree.

The system acts as a multistage clustering pipeline: once the raw search results are obtained, they are pumped towards the final clusters through a number of stages. The first step is to clean the data; the data is then split into phrases ready to be turned into a suffix tree in the next step. The base clusters are then extracted from the suffix tree and combined using a single-link clustering method with a scoring method based on phrase length and the number of common phrases. Finally, a maximal cluster covering is found to reduce the number of clusters to a more user manageable number before returning the results to the user.

4.4.1 Why Suffix Tree Clustering Was Used

Suffix Tree Clustering is the basis for this project as it was the most effective of previous clustering systems using solely text information and it handled phrasal information and overlapping clusters in a suitable way. As mentioned previously, had it been known earlier that short n-grams provide equally good results, a method based on that approach would most likely have been developed. Nonetheless, suffix trees provide a very efficient representation for finding documents with any particular phrase in common. The Suffix Tree clustering method was also selected for implementation as the first few stages of the algorithm were the basis for the newer algorithm combinations discussed later in this chapter, time permitting these other algorithms would have also been implemented.

4.4.2 STC Stage 1 - Document Cleaning

The first stage of the suffix tree clustering algorithm is similar to the approaches used in most web document clustering algorithms which use textual information. The documents have HTML tags, punctuation and similar non-informative text removed. A set of stop words containing very common and uninformative words such as 'the', 'it', 'as', 'on', 'here', ..., are removed. Some stemming is applied to reduce words to their root form, for example, 'computer', 'computing', 'compute', 'computers', which are all words with similar meaning are all changed to the root word 'compute'. Finally, the document is broken up into phrases. There are many places where phrases can be split, including standard sentence terminators such as full stops or exclamation points; the location of surrounding html tags and tag types can also be used for splitting.

4.4.3 STC Stage 2 - Building Suffix Tree

What is a suffix tree

A suffix of a sequence is a subsequence from some element to the end of the sequence. A suffix tree is a trie containing all suffixes of a set of sequences. The type of suffix tree used in the suffix tree clustering algorithm is known as a generalised suffix tree; the 'generalised' means that it may contain multiple strings. There have been many uses for suffix trees including finding the Longest Common Substring, All-Pairs Suffix-Prefix Matching, All Maximal Repeats and others [5]. The main difference between the traditional suffix tree usage and the usage in suffix tree clustering is that traditionally the objects added to the structure were strings and the atoms were individual characters; in suffix tree clustering, the objects to be added are phrases and the atoms are individual words. Suffix Tree's are defined in [23], [38] and [39] among others. Presented here is the definition of a generalised suffix tree used in this project, which is based on [39]. Chosen because it was the only definition available while undertaking this project, it was not until near the very end of this project that the PhD thesis [38] and paper [23] were found.

A generalised suffix tree of a set of documents, each containing a set of phrases, is defined as the following:

1. A sub-phrase of a phrase is a subsequence from one word to another word of the phrase.
2. A suffix-phrase of a phrase is a subsequence from some word to the end of the phrase.

3. A generalised suffix tree is a trie of all the suffix-phrases of phrases in a set of documents.
4. Each edge is labeled with a non-empty sub-phrase — the edge-label.
5. The node-label of a node is defined to be the concatenation of the edge-labels on the path from the root to that node.
6. The base-document-set of a node is the set of documents containing the node's node-label as a suffix-phrase.
7. The document-set of a node is the set of documents containing the node's node-label as a sub-phrase. Alternatively, the document-set of a node is the union of the base-document-sets of the nodes in the subtree rooted at that node.

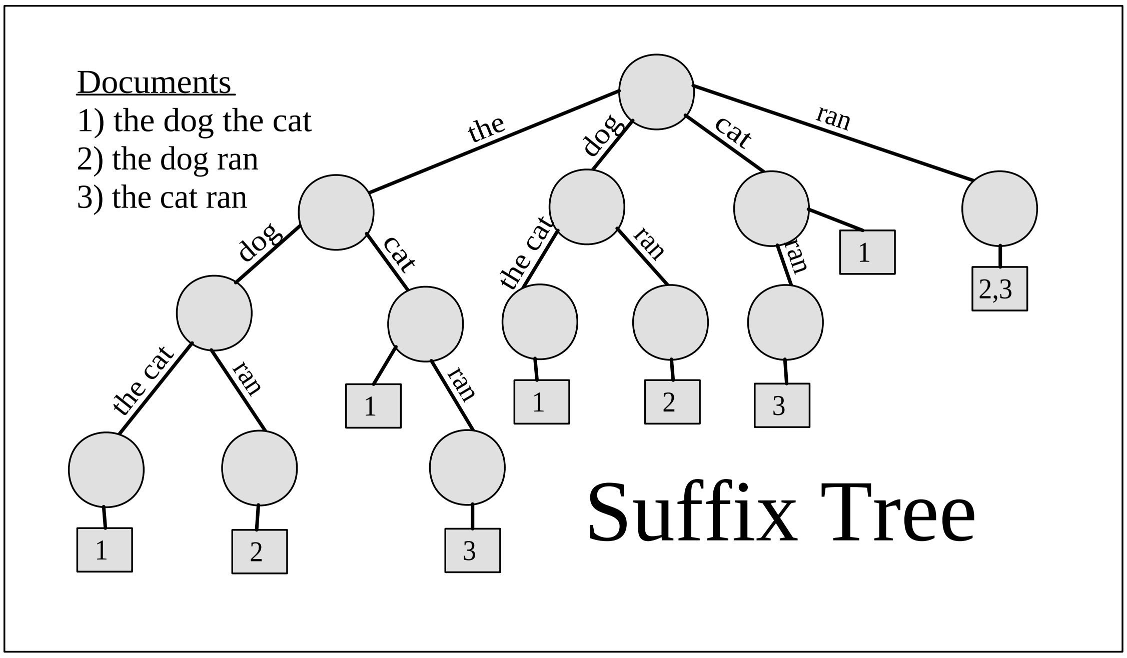


Figure 4.1: Example Suffix Tree

Figure 4.1 shows the suffix tree that would exist if three documents each with a single phrase were added, 'the dog the cat', 'the dog ran' and 'the cat ran'. Figure 4.1 shows edge-labels and base-document-sets, but does not include node-labels or document-sets. Each node on the tree is a base cluster. Consider node A, this node is the base cluster which represents documents which contain 'the dog' and this base cluster contains documents 1 and 2.

Constructing the suffix tree

There are several different suffix tree construction methods, the most simple is the naïve method which simply enters each suffix of the string to be added. The method is quadratic in the length of the string to be added and is described in [38]. There are also several other suffix tree building methods; one of the more recent ones is Ukkonen's algorithm which is described in [23].

As described in [38], Ukkonen's algorithm constructs what are known as implicit suffix trees, these trees do not include the terminating atom nor do they include suffixes that match the prefix of other suffixes. When complete, the final implicit suffix tree is converted into a real suffix tree and the entire algorithm has a time bound of O(m * min(log $|L|$, log m)) where m is the string's length and $|L|$ is the size of the language. The most important trick used in the implementation is the suffix link: essentially this is a link from one internal node to another, possibly distant, node to save traversal time. One problem with these methods is that once memory is used up, the paging to disk is very expensive and the methods slow down considerably.

The approach considered in this project, when the naïve method proved too slow even for testing, was to use a hash table lookup method to jump to the correct branch in constant time, this provides a simple algorithm for adding a suffix-phrase, which is linear in the number of atoms. Compared to other methods however, this is a very space inefficient method and this may cause issues with particularly large situations. However these were not encountered even with full document text and several hundred documents; with larger document collections the problems of paging would surely be run into however.

4.4.4 STC Stage 3 - Clustering

The base clusters of documents corresponding to each node from the suffix tree in stage 2 are identified and each is assigned a score which is based on a combination of the number of documents in the base cluster and the length and words in the associated phrase. These base clusters are then clustered to form the output clusters. The standard method associated with suffix tree clustering and as described in [38] and [39] is to use the single-link clustering algorithm. A link is placed between two base clusters if the number of documents in common between the two base clusters divided by the number in one of the clusters is greater and the number of documents in common divided by the number in the other cluster is also greater than a number, called the similarity constant. This generates the base cluster graph, with base clusters representing nodes and edges between clusters that are sufficiently similar. If two base clusters meet the criteria for having a link between them on the base cluster graph, then they are said to meet the similarity requirement. To find the output clusters, a connected components graph algorithm is run, and each component is a set of base clusters, whose union forms the output clusters. Some method of scoring is then applied, such as the sum of the base cluster scores [6], then the highest scoring clusters are returned.

4.4.5 Variations on STC Stage 3

Several original variations on the third stage of the suffix tree clustering algorithm were considered in this project, but were not implemented and tested due to lack of time. They are discussed briefly here to provide insight into their possible use in future research in this area.

Instead of computing the similarity between every pair of clusters, only the higher scoring clusters need be placed in the similarity graph. This is similar to the method described in [39] where only the 500 best scoring clusters are kept at any time. There are more advanced variations of this which could allow for the number of clusters to vary depending on the total number of cluster and the scores of the clusters.

Another variation is to store the similarity measures on the graph edges, so that the clusters could be changed dynamically at a later stage. This could possibly be used to create hierarchical clusters: setting a high similarity constant reduces the number of joins and gives lower level clusters, while a lower similarity constant created more joins giving high lever clusters. The other possibility is to alter the connected component stage, to consider the strength (base cluster similarity) and structure of the connected components. Loosely or weakly connected components could be identified and split using some form of graph algorithm. Perhaps this would allow outliers, chained clusters (clusters linked in pairwise fashion) or unrelated clusters to be split or identified.

The third stage can also be affected by changing the amount of information available. Consider using a dynamic stop word list, expanding the list corresponds to shrinking the suffix tree and hence results in the removal and combination of base clusters. This could be useful for forming a hierarchical clustering: the size and scope of the base clusters affects the level at which clusters form, for instance, having fewer base clusters leads to higher level clusters. Words that occur in too few or too many documents could be added to the stop list, if the bounds on the stop list were set using percentages, then when clustering recursively the document sets to be clustered on the second level will be smaller and hence, less strict bounds will be used, reducing the size of the stop word list and hence, cause more specific clusters to form. Which is exactly the property required of hierarchical clusterings.

Changing how clusters are linked could fix the chaining problem of single-link clustering. Instead of linking together clusters individually which can create chaining effects, when two clusters are joined they could be merged immediately and the merged cluster could be considered against others. This of course means that the order in which pairings are considered is important. In fact this is simply an agglomerative method; the next section discusses this approach and a K-means approach to the base cluster clustering problem.

Another method for modifying how clusters are linked could provide a better solution to the chaining problem. A component is either a cluster or a set of linked clusters. In addition to the similarity requirement holding, two components may only link if each cluster in each component meets the similarity requirement for joining with a large enough fraction of the clusters in the other component. Doing this means the resulting clusters depend on the order in which the components are considered. A possible solution is to add copies of the original components after each join, thus each and every possible join will be considered. However, there are 2^n possible joins making this intractable.

This section ends with a possible approach for identifying outliers and breaking cluster chains. If the similarity requirement holds between a base cluster and too many other clusters, then the phrase used is probably not a good determinant of topic and the base cluster should be eliminated as it is a potential cause of outliers and unwanted chaining. The challenge is in determining what fraction of the clusters is too many and this is search dependent.

4.5 Other Clustering Techniques

4.5.1 K-means

K-means is a very commonly used and well documented iterative partitional clustering technique. Descriptions can be found in [17] and [25]. The algorithm starts with K clusters and modifies them all simultaneously. The idea is that there is some distance measure between documents. Each cluster is represented by its centroid, and each document is allocated to its nearest cluster. After each iteration, each centroid is recomputed based on the documents in its cluster and the process repeats.

As in [25] the algorithm is:

1. Select K points as the initial centroids
2. Assign all points to the closest centroid.
3. Recompute the centroid of each cluster.
4. Repeat steps 2 and 3 until the centroids do not change.

K-means does not find overlapping clusters and is non hierarchical. When it comes to clustering documents such as web pages, there is no obvious method for a distance or similarity measure. One word based method is to represent each document by a vector containing the term frequencies of each word weighted by the inverse document frequency of the word. To compute the distance between two documents, the vector cosine of the two vectors is computed and the centroid is defined as the average of the vectors of the documents it represents. This allows the standard K-means algorithm to be directly applied. A description of this method is found in [25].

4.5.2 New Similarity Measures

As an alternative to the standard K-means, this project considered a new use of K-means as a replacement third stage for the suffix tree clustering algorithm. There are two methods for making this integration: one is to cluster the base clusters, the other is to cluster the documents.

Applying K-means in the third stage to clustering the base clusters requires a new similarity measure and centroid definition. The similarity measure can be defined as the number of documents in common between the base clusters divided by the number of unique documents in both clusters. While the centroids can be the phrases in common between all documents belonging to the centroid.

Applying K-means in the third stage to clustering the documents requires a new similarity measure and centroid definition. Using base cluster membership of the documents, the similarity measure can be defined as the number of base clusters both documents belong to, divided by the total number of unique clusters they belong to. This application of K-means to the third stage allows the centroids to be defined in two ways; the first is to use the phrases common to all documents in the centroid, the second is to use the set of base clusters common to all documents in the centroid.

Unfortunately there was not time to implement either of these alternate stage 3 clustering algorithms, although some further observations were made.

The K-means method based on base clusters finds overlapping clusters as multiple base clusters may contain the same documents. But as with tradition K-means the version based on documents does not find overlapping clusters. These K-mean methods can also easily be extended to a hierarchical methods by recursively applying the clustering algorithm to each cluster.

One potential problem with any K-means derived approach is that the resulting clustering depends on the random start points chosen for the centroid locations. If some heuristic evaluation is defined on the quality of a cluster set, then K-means could be run several times and the cluster set with the highest heuristic evaluation could be used instead. This could help avoid the pitfall of ending up in the many local optimums that exist in the search space.

4.5.3 Agglomerative

As described in [25] and [1], an agglomerative clustering algorithm starts with a selection of documents to be clustered, then at each step either, the two most similar documents, clusters, or document and cluster are merged. This requires some definition for the similarity of clusters. As described above, the standard cosine vector similarity measure can be used for this, or either of the two base cluster directed methods considered in this project could be used with an agglomerative clustering algorithm in the third stage of the suffix tree algorithm. Another variation of the base cluster method considered in this project would be to use an agglomerative method along with the similarity being defined as the difference of the number of documents in common and the number of documents not in common.

As in [1] a simple agglomerative algorithm is:

```
1. Find the 2 closest objects and merge them into a cluster
2. While more than one cluster remains:
        Find and merge the next two closest points, where a point is either
        an individual object or a cluster of objects.
```

For this project, the objects could be either a document or a base cluster, if using the suffix tree clustering method.

The agglomerative algorithm is naturally hierarchical and the hierarchy is determined by the order of merges. A cut through the tree can be taken to split the single remaining cluster into an arbitrary number of sub-clusters. These can be further split by performing cuts to form hierarchies of clusters from the subtrees. As with K-means, the agglomerative method described here does not create overlapping clusters, but if the base clusters are clustered, then the documents will naturally overlap.

Again, the problem with the hierarchical method is that only one possible tree is created, and the similarities between objects (documents or clusters) may be very close or even identical. In these situations it is hard to know which to choose. One possibility is to do the same as could be done with K-means and define a heuristic evaluation method. Then by creating several hierarchical trees by running the agglomerative algorithm several times these could be evaluated and the best selected. Different cut points could also be used and evaluated to choose the best cut location for each tree. A potential question is, how should the agglomerative algorithm be changed to handle the non-determinism between selecting the clusters.

This project proposes several alternatives. One would be to randomly select from the closest few alternatives; another would be to use some variation of the metropolis-hastings method. This could be done by assigning the probability of choosing to join two objects as one minus the similarity of the two objects divided by the total similarity between all object pairs. Then a pair could be randomly selected according to its probability. The first drawback of this method is that with a large number of clusters, there would be n^2 pairings and hence the probabilities would be very low, leading to a large number of samples being needed before an acceptance was reached. The other drawback is that very poor pairings could be selected. A solution to both of these is to only allow the best few pairs to be considered.

4.6 Summary

This chapter has introduced two kinds of information that facilitate web document clustering, textual information and link information. Several properties of clustering algorithms including overlapping and hierarchical properties were introduced and their desirability for web clustering was discussed.

The basis of this project, suffix tree clustering was then introduced. The three main stages, document cleaning, building the suffix tree and clustering were identified and a particular focus was paid to the last two stages. In particular a variety of proposals for modifying the third stage are given.

Finally, two alternative clustering techniques K-means and agglomerative clustering were discussed. Extensions to these methods that allow them to replace the clustering stage of the suffix tree clustering algorithm were outlined, new similarity measures were constructed and new methods for avoiding the local minima these methods may enter were discussed.

Chapter 5

Evaluating Clustering Methods

5.1 Introduction

Evaluating cluster sets returned by a web clustering algorithm is not an easy task: providing an objective measure of the quality of a cluster set is difficult as any definition is dependent on subjective definitions of what a good cluster set is.

Due to the difficulties of evaluating cluster sets, research in this area has thus far used a wide variety of evaluation methods each of which is specific to the authors own subjective definition of what a good cluster set is. The result is that evaluation results cannot be compared and it is difficult to tell which of the current methods works best. The difficulty of evaluation is further increased since different methods and different authors use different raw data and the methods are also specific to the goal of the research. Methods tested on small data sets may be too inefficient to perform on large data sets and methods that run on larger data sets may not produce good results compared with the small data methods. All these factors and others need to be considered when choosing a cluster evaluation metric.

This chapter considers the approaches and measures which can be used to evaluate clustering results, discusses the implementations of these used in current research and introduces a new method developed in this project to measure performance using precision and recall which has the aims of minimising bias, while allowing cluster quality to be evaluated. The method introduced is only a starting point for evaluation and ideas for future development of this method are discussed.

5.2 Approaches and Measures

There are two main approaches to evaluation in [28]: the gold standard approach and the task oriented approach. The gold standard approach requires the construction of an ideal or reference solution to the problem. With respect to clustering, this would mean creating an ideal clustering of the set of documents to give several test cases. These clusters would have to be generated manually by human experts. The task oriented approach evaluates directly how well the solution solves the task. For clustering this would mean a human would manually compare the clusters and consider whether the documents within them were sensibly placed and so forth. Task oriented methods are typically more prone to the effects of subjective evaluation and so this needs to be monitored carefully especially when using a task oriented approach.

Various numeric evaluations of the quality of clusters with respect to a gold standard or task oriented approach are available. The most common of these are precision, recall, entropy and F-measure. How these are actually mathematically defined depend on which approach is used for evaluating the quality — the gold standard or task oriented approach. Since the definitions change from paper to paper, no definite description of any of these measures can be given. But the general sense of precision and recall that is somewhat consistent across papers is that precision is the accuracy of the clusters, this relates to how many outliers or irrelevant documents are included in the cluster — fewer outlier or irrelevant documents will typically lead to larger precision. Recall, on the other hand, is typically related to coverage — how well do the clusters cover all the documents provided originally.

5.3 Methods used in current research

To get a feeling for the different approaches, the evaluation methods of the better web clustering research papers were evaluated and the pitfalls and strong points considered. Several of these will be discussed in this section.

5.3.1 Zamir - Suffix Tree Clustering

Zamir uses a task based approach in [38]. A search was defined, such as 'Jaguar' and a topic of interest, such as 'information about the animal' was also defined. Precision-Recall, average precision and expected search length are the measures used to evaluate the results. Here we describe how precision-recall and average precision were determined by Zamir. The approach used to determine these is based on reordering the ranked list of documents using the clusters. The clusters are used to reorder the ranked list by selecting the clusters in order of those which best match the topic of interest. The results were based on the assumption that users could correctly choose the best clusters in order 100% of the time; their user testing showed this was more like 80%. However, the results were fairly similar using both measures and the average was usually taken. Each document returned is marked as either relevant or irrelevant to the query, repeated documents are either ignored or marked as irrelevant. The definitions of precision and recall are thus quite different than one would usually expect: to determine precision, it is assumed that a user would consider the first 10% of the documents in the reordered list and precision is then defined to be the number of relevant documents seen divided by the number of documents seen. To determine the average precision, the sum of the precision calculated after each document in the first 10% is divided by the total number of relevant documents. Precision-Recall graphs are constructed by determining the average precision with a number of different percentages of the documents included, until 100% recall is achieved. Recall is the number of relevant documents seen, divided by the total number of relevant documents.

The reordering process is heavily subjective since humans must determine the ordering and whether a document is relevant or not to a query after the clustering is performed. It is also noted that users may only want to look at a single cluster, however as noted in [38] different clustering methods produce different numbers of clusters. If there were a limit on the number of clusters to be returned, say 10, the resulting clusters would have to cover the documents more fully, although a single cluster need not cover a users topic of interest entirely. It is indicated in [38] that average precision methods are influenced by the size distribution and that the results often over-penalise small clusters. While the methods used for calculating precision in this project were different to those of Zamir, it was found that smaller clusters tended to have significantly higher precision that larger clusters. Also smaller clusters tend to be of less use to the end user and as such these should actually be evaluated as lower quality clusters. However, precision is a measure of accuracy, not size; recall is used to determine whether the clusters are large enough, since if they are too small, a fixed set of ten clusters would not be able to cover very many documents. The indication that the measures of average precision are affected by cluster size would indicate that the precision measure used by Zamir is not particularly good. Probably the largest issue with Zamir's approach is that since no limit is placed on the number of clusters, a series of very small clusters, one document in each, would produce perfect results.

5.3.2 Wang and Kitsuregawa - Combining Link and Contents Information

In [31] and [32], Wang and Kitsuregawa take a mix between a task oriented and gold standard approach. Following a task oriented approach, each document is marked as either relevant or irrelevant to the query. Typically around 75% of documents were marked as relevant. Precision is then defined as the number of relevant documents in clusters, divided by the number of documents in clusters. Recall is defined as the number of relevant documents in clusters, divided by the number of relevant documents. These measures seem to be evaluating how effectively the clustering method is able to remove irrelevant documents rather than the quality of the clusters. For instance, in a search for 'jaguar', as long as documents relating to cars, animals, macintosh os and games console were included in clusters, even if they were all in one cluster or mixed up in a random fashion, perfect results would be achieved. This method would provide a good way to evaluate the irrelevant reduction mechanisms as discussed at the end of chapter 8, but for evaluating cluster quality it is of no use. Following a gold standard approach, clusters were manually created and the average entropy of a set of clusters was computed by somehow comparing the manually created clusters and actual clusters to provide probabilities of documents being members of each cluster. The entropies were then calculated from these probabilities and averaged. The standard entropy formula is used. $E(j) = \sum_{i=1}^{n} p_{ij} log(p_{ij})$, where E(j) is the entropy of the j^{th} cluster, n is the number of classes, and p_{ij} is the probability that cluster j belongs in the i^{th} class. But no definition of how the probabilities were calculated is given, making the results of no use to the reader. With precision and recall not considering cluster quality and entropy not being properly defined, no real conclusions about cluster quality can be drawn from their work. This is quite disappointing as their methods are quite interesting. The importance of an effective evaluation method is thus crucial to evaluating cluster work and a simple and effective measure of evaluating actual cluster quality is essential.

5.3.3 Wong - Incremental Document Clustering

In [7] and [8], Wong evaluates an incremental document clustering technique using a task oriented method based on F-measure. F-measure is a combination of precision and recall defined as: $F(T) = \frac{2PR}{P+R}$. Each document is assigned a topic T by a human expert. The precision and recall for a cluster X are then defined in [8] as:
N_1 = number of documents of topic T in cluster X

$N_2 =$ number of documents in cluster X
$N_3 =$ total number of documents of topic T
$P = Precision(X,T) = \frac{N_1}{N_2}$
$R = Recall(X,T) = \frac{N_1}{N_3}$
The cluster with the highest F-Measure is selected to be the cluster for T. A weighted average of the F-measures for each topic T is used to evaluate the overall clustering.

This style of evaluation effectively computes the quality of each cluster returned and provides a single measure of the quality. However one drawback of this approach is that the results are highly dependent on the manually selected clusters. For instance, in the search for 'jaguar', one topic may be car, but there may be several clusters which relate to subtopics of car such as 'parts' and 'car clubs'. Since only one of the clusters will be selected to represent the car cluster, this method is thus targeted towards evaluating how well classes reconstruct the user specified clustering, rather than how good the clustering is. There is no way to guarantee that the user clustering is optimal and so other variations which are equally good should be accepted with equal probability. The evaluation method developed as part of this project is a variation of Wong's method which addresses this problem.

5.3.4 Zamir - Grouper

A completely different approach to evaluation is to conduct a user evaluation and to see how effective the methods are for assisting actual users. Zamir [40] conducted such an evaluation with his Grouper system. However, Zamir concluded that conducting a user evaluation is very difficult and by just observing search logs and usage no real conclusions can be drawn by this method.

5.4 A Better way to Evaluate

This project introduces a new method to evaluate clustering results that is in some ways similar to the approach taken by Wong [7], but with the pitfalls of that method avoided. This section starts by introducing the criteria of a good web clustering evaluation method. Following this, the evaluation method is described, and finally the new evaluation method is analysed, its flaws identified and possible future improvements suggested.

5.4.1 Evaluation Criteria

To be effective at evaluating a clustering algorithm requires the evaluation technique to meet a number of criteria. We have decided on six such criteria: Evaluates Cluster Quality, Evaluates Document Coverage, No Bad Solutions, Minimal User Bias, Minimal Method Bias and Statistically Significant. None of the evaluation methods used by the current research described meet all six of these criteria and hence all fail to provide a good evaluation methods due to at least one of these reasons.

Evaluate Cluster Quality

Any good clustering should have high individual cluster quality. This means each cluster should be made up of only documents which have the same or highly related topics. Putting documents about 'cars' and documents about 'animals' in the same cluster from a 'jaguar' search would mean low cluster quality and the evaluation method should identify this. Hence the evaluation method must distinguish between clusterings of different qualities.

Evaluate Document Coverage

Document coverage is the fraction of the documents that are placed within a cluster that represents their topic. Individual category coverage measures the same fraction, but is unbiased by the size of the categories. For instance, methods that increase the coverage of large clusters at the expensive of small clusters will do poorly at individual category coverage, while methods that increase coverage of all clusters equally will do better at individual category coverage; all methods will do equally well at overall document coverage. Having a higher fraction for overall document coverage could mean missing out other categories all together, so the evaluation method should consider this and the document coverage metric should be based on individual category coverage rather than overall document coverage.

No Bad Solutions

All too frequently, an evaluation method can be broken by a clearly bad solution. A bad solution in the case of web clustering is a set of clusters that is clearly not very useful to the user, but which score highly under the evaluation method. An example is the evaluation technique implemented in [38] where a large number of very small but high quality clusters would produce excellent results. Clearly having a low ratio of documents to clusters is a bad thing for usability and does not do anything to simplify the task at hand for the user. A criteria for an evaluation method is that there should be no 'bad solution' that gives a high value to a useless clustering.

Minimal User Bias and Method Bias

Any sort of subjective task such as cluster evaluation requires user input as to what the desired results are. But there is no single way to do this and the evaluation method should minimise the amount of bias that could be introduced towards any particular clustering. One clustering method could be targeted towards a flat clustering of lower level clusters, whereas another could be targeted towards a hierarchical clustering of fewer higher level clusters. The evaluation method should not be biased towards any particular cluster configuration due to user bias or method bias, which means that the degrees of freedom in user selection should be minimised.

Statistically Significant

The final part of any evaluation is to determine the significance of the results. By clustering on data with the labels randomly rearranged, a benchmark can be established; the benchmark can be used to determine the significance of other tests performed using the same evaluation method and data. Without this type of benchmark, one evaluation could achieve 90% and another 95%, it may be the case that the one which achieves 95% is significantly slower than the other and so the other is selected for use. However, consider the case where on random data, 89% is achieved. This indicates that on a transformed space the results to be compared should really be 9% against 55%. The difference between results depends on the significance of the measure, so any evaluation method should determine the significance of the results before drawing any conclusions.

5.4.2 Evaluation Method

This project uses a gold standard method by assigning documents to categories and defines new precision and recall functions for evaluating web page clustering results. Precision measures how accurately the clusters represent a single topic, while recall measures how well the actual clusters cover the documents.

Each document must be assigned a category by a human before clustering; these are termed user assigned categories. The category assigned should be the highest level category assignable. For example, in a search for jaguar' with documents for 'car clubs', 'car parts', 'car dealers' and 'animals', documents should be classified as car or animal.

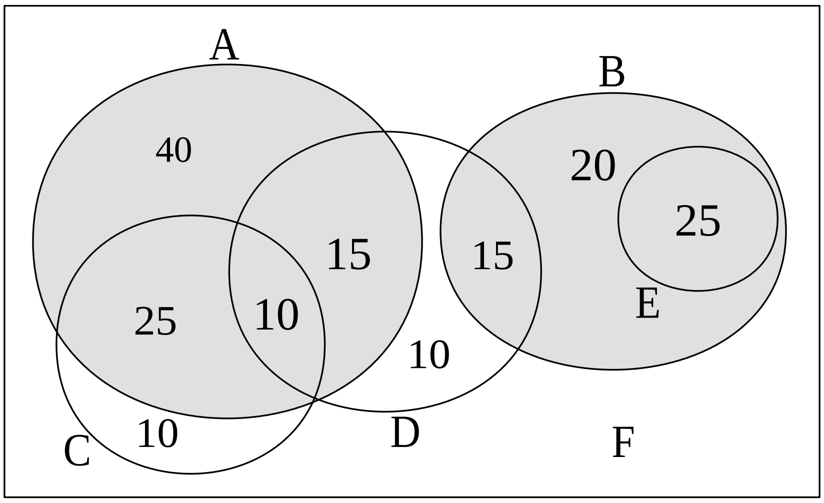


Figure 5.1: Cluster Evaluation Example

Figure 5.1 gives an example situation, in which the shaded ovals, A and B, are the user assigned categories, and the other ovals, C,D and E, are resulting clusters to be evaluated. F contains documents that are in sufficiently small user assigned categories that these documents would ideally be grouped in an 'other' category. The numbers in figure 5.1 specify the number of documents in each region.

Calculating precision and recall requires knowledge of what class each cluster is representing. Each resulting cluster is assigned to the user assigned category which it has the largest number of documents from. In the example, C and D represent A, while E represents B.

Precision is defined as the total number of documents from each cluster in its represented category divided by the total number of documents in the clusters.

Example: $\frac{(25+10+10+15+25)}{(25+10+10+10+15+10+15+25)} = \frac{85}{120} = 71\%$

Recall is defined as the total number of distinct documents from each cluster in its represented category divided by the total number of documents in the user assigned categories.
Example: $\frac{(25+10+15+25)}{(40+25+10+15+15+20+25)} = \frac{75}{150} = 50\%$

Overall Precision and recall are determined by taking a non weighted average of each over all sufficiently large user assigned categories, where sufficiently large is defined as categories that include at least four percent of the result set. Note that in the actual implementation and evaluation later in this report, we actually use the weighted variant of these; it was only recently realised that the non weighted average is actually a better method to use.

Several clustering methods can then be compared by comparing the precision and recall of each method. To maintain statistical significance of the results, for each test, each method should also be used to cluster randomly labeled documents. Actual results can them be compared to these statistical indicators to indicate statistical significance of the results.

5.4.3 Evaluation Method Analysis

The evaluation method described should be very effective and meets all the stated criteria. Cluster quality is evaluated by way of the precision metric, which measures how pure each cluster is and identifies any complete outliers with ease. The use of very high level topics is useful in many respects as will be discussed shortly, but with respect to cluster quality, this approach of using high level topics may leave something to be desired. Consider a search for 'jaguar' where the clustering contained several smaller sub-clusters of car, any permutation (generated by a sequence of 1 for 1 swaps) of those car documents between the car clusters will give identical results. Clearly some permutations will be bad, for instance, two smaller clusters with each cluster being made up half of 'car clubs' and half of 'car parts'. This clustering would have the same results as one which split the 'car clubs' and 'car parts' perfectly. This is an example of a bad solution being allowed to exist. If this method of evaluation is to become the standard for web clustering evaluation, then this issue will need to be addressed. One possible solution would be to form a hierarchy of topics and assign each document to somewhere on this hierarchy. This would create more work in the creation of test sets and a more complex method of determining the category a cluster is representing would be required. It would be possible to use something along the lines of the entropy measure described by Wong in [7] to select the best matching category. The difficulty comes in selecting between a match against a child category and a parent category; clearly if a cluster is trying to represent the child category, then choosing to consider it as being the parent category would only serve to reintroduce the bad solution problem. The simplest approach and the approach used in this project is to simply assume that the quality of the sub-clusters is equal to the quality of the matching against the parent categories. This is not correct, but it provides a simple approximation to the correct answer.

Document coverage, and in particular individual document coverage, is measured effectively by recall. The non weighted average is taken to find the overall precision and recall to avoid bias towards algorithms that do a good job on a single large cluster, but poorly on a number of smaller clusters. For instance, consider a case with one large cluster with 80% of the documents and four smaller clusters with 5% of the documents each. Suppose that precision and recall are both 100% for the large class and 20% for the smaller classes. In this case, with a weighted system there would be 84% precision and recall overall, but with a non weighted approach there would only be 32% precision and recall overall.

As already mentioned there is at least one type of bad solution that can occur using this evaluation method, although we have suggested a solution to this problem. To our knowledge there are no other bad solutions which can exist as a result of using this evaluation method.

By only needing high level categories to be assigned to each document, there is very little user bias and it is typically fairly easy to determine the high level category a document within a result set should belong to. To deal with the problem with the cluster quality evaluation method, it is likely that more information from the user / expert will be required and this will increase the amount of user bias as classifying documents into lower level categories is a more difficult and more subjective task. It may be necessary for some form of fuzzy set membership to be used to ease the burden and reduce user bias when no clear distinction between categories can be made. It may be better to ignore the bad solution which can result and use a user evaluation of the results to detect these. The method of comparing with high level categories removes the bias against any particular method and both high level and low level clusterings will evaluate well. However, the trade off for this is that it works too well and the bad solutions as already mentioned can result.

To detect the statistical significance of the results, the random test is performed. This allows a baseline or benchmark result to be established to which other results can be compared.

A final drawback of this method and all other evaluation methods discussed excluding the user evaluation is that they do not work with particularly large data sets. For instance consider clustering a standard result set from Google [10] which may contain millions of results. Going through and making a manual assignment and determination of each of these pages would take an enormous effort and unsuitably large amounts of time, leaving only usability and user based evaluations as possible. As already mentioned these type of studies are fairly unreliable and can really only provide a guide to the effectiveness of the results. One possible method of evaluation is to pick a subset of documents, say the first few hundred that are to be clustered, since these should be fairly representative of the clusters. The result set could be shrunk to just these few hundred documents and evaluated using the method described above. As these documents should be representative, the results should be fairly accurate and the standard deviation could be used to set the margin of error. It may turn out that more than several hundred results are needed to give a small enough error bound, but the number required would hopefully not be prohibitively large.

5.5 Summary

This chapter discussed the approaches that are often taken to evaluating web clustering and the measures used to perform the evaluations. Several of the more popular methods used to evaluate existing web clustering research were introduced, discussed and any pitfalls or strong points highlighted. A set of evaluation criteria have been established which specify the minimum qualities that any web clustering evaluation method must have. The method developed in this project and proposed as a standard for future web clustering evaluations was described. The evaluation method was then analysed and the strong points highlighted. Two problems with this method and most other methods were identified and potential solutions explained.

Chapter 6

System Framework

6.1 Introduction

The previous chapters of this report have given some background on web search clustering, described some of the existing solutions, considered past clustering research, analysed several clustering methods and introduced a new evaluation method. This chapter serves as an introduction to the system developed for this project to test and evaluate different clustering methods. It starts out by giving an overview of the system and then it outlines the main clustering framework used in the system. Following this, it provides details on the language choice, user interface and database backend.

6.2 System Overview

There are two different systems with different goals: the production system and the testing system. The production system aims to be efficient and has a minimal configuration, while the testing system is designed to facilitate efficient testing and fair evaluation of different configurations. Figure 6.1 provides an overview of the production system framework and the basics of how the phases interact. The system starts with the user giving a request to the frontend in the form of search keywords. This is forwarded to the clusterer which goes to Google and obtains the search results for the user's query. Once the search results are obtained, they are cleaned and phrasified ready for clustering. The documents are then clustered by building the suffix tree, combining the base clusters and applying the maximal covering algorithm to find the clusters to be output to the user. The clusters are then output to the user.

Figure 6.2 provides an overview of the testing system framework and the basics of how the phases interact. The two main differences between the production system and the testing system is the inclusion of a testing database and a more configurable user interface frontend, henceforth referred to as the testing interface. In the testing system the database fits in after the phrasifying step. The data generated up to the phrasifying is stored in the database to facilitate more efficient testing by eliminating the need to redo the first few steps for each test. Having the data stored also ensures fair testing in that each configuration and algorithm is evaluated using the same data. The testing interface implemented is designed to allow tests to be carried out, results evaluated and parameters altered. This is in contrast

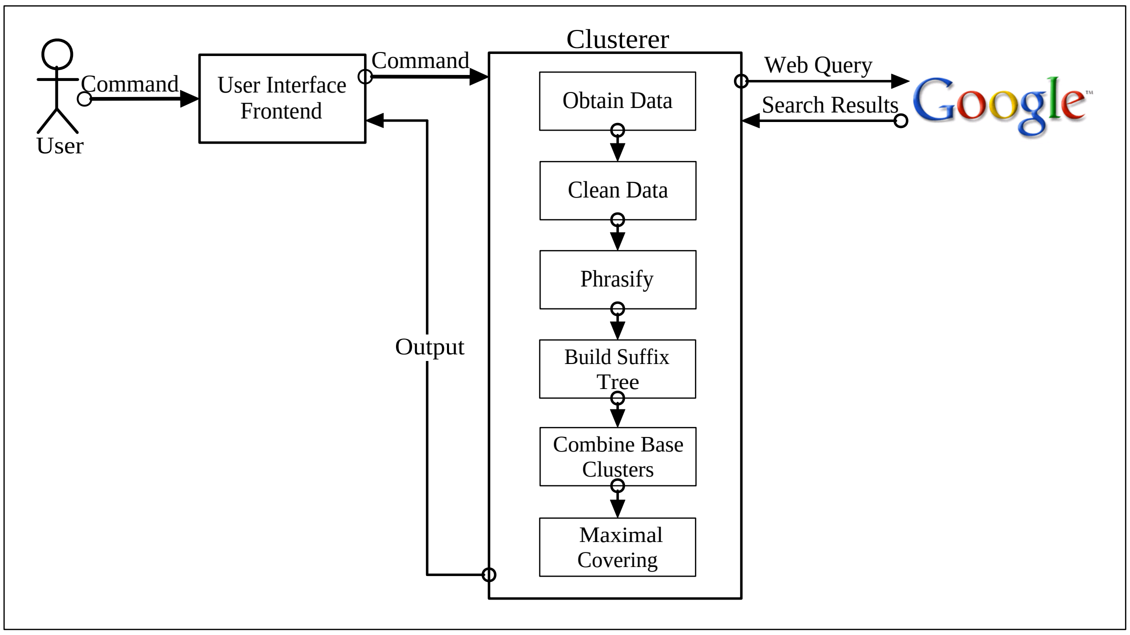


Figure 6.1: Production System Framework

to the production system where the user interface has none of these abilities and the clusterer would be altered to exclude the extra steps required for carrying out these modifications and evaluations. In this project only the testing system has been implemented. From this point forward, unless otherwise stated, all discussions relate to the testing system implemented in this project. More details on the individual components and their interactions will be dealt with in the following sections.

The clustering tester is the backbone of the testing system.

Phase 1: Obtaining Data Phase 1 is initiated through the user interface. In phase 1 the raw data is acquired and stored in the database ready for cleaning. Phase 1 is discussed in chapter 7.

Phase 2: Preparing Data Phase 2 is initiated through the user interface. Phase 2 transforms the data gathered in phase 1 into clean and phrasified data. Phase 2 is discussed in chapter 7.

Phase 3: Clustering Data Phase 3 is initiated through the user interface by selecting what test to run — algorithms, data and parameters. Phase 3 is the most important step — where the actual clustering takes place. The chosen data is loaded into the system from the database and the chosen clustering method is performed. This constructs a set of clusters ready for phase 4. Phase 3 is discussed in chapter 8.

Phase 4: Evaluating Clusters Phase 4 is initiated automatically after phase 3. The clusters are evaluated using the method introduced in section 5.4 and the results and the clusters are output according to the options set by the user before phase 3 began. Phase 4 is discussed in chapter 9.

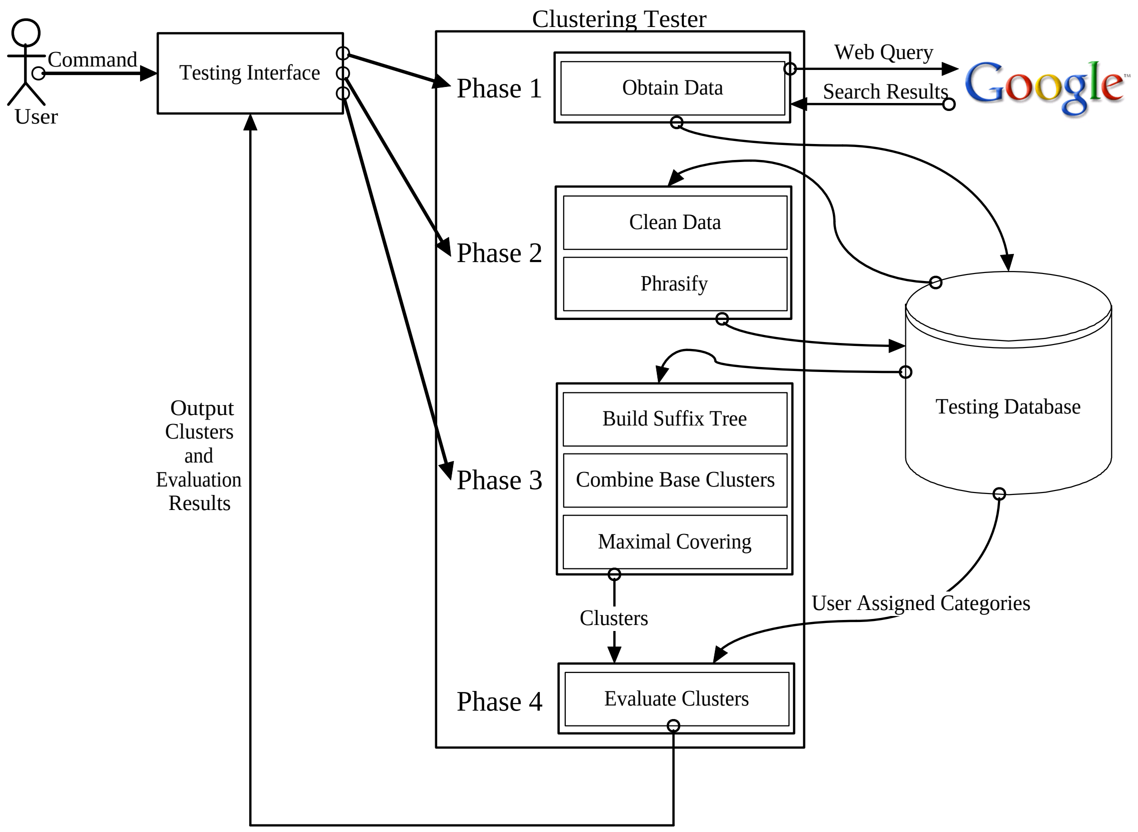


Figure 6.2: Testing System Framework

6.3 Language Choice

For this project, Java 1.4.2 [26] was selected as the language of choice. Java was chosen due to its simplicity and extensive class library of basic packages allowing for basic data structures and functions to be used out of the box reducing the programming effort. The language is also cross platform compatible, allowing the system to be run on most common operating systems including unix and windows.

Java is not as memory efficient as C++, due to the overhead of the garbage collector. With the large number of allocations and de-allocations required in this system C++ may have been more efficient, although for the most part Java can be as efficient as C++ [18]. For this research the Java implementation is sufficiently efficient and the simplicity and cross platform functionality win out.

6.4 The Testing Interface

The main part of the testing interface for interacting with the system is a standard Java Swing interface and is thus able to run on multiple platforms including both Unix and Windows.

There are three main menus in the testing interface: the main operations menu, the output options menu and the test options menu. These menus allow full control of the parameters of the testing system. Batch testing is also provided to reduce burden on the tester.

6.4.1 Main Operations Menu

This menu allows the tester to execute any part of the testing system. On the main operations menu there are options which correspond to each of the four main phases outlined — options for obtaining raw data, cleaning and phrasifying and running clustering tests. These actions generally cause a sequence of dialog boxes requesting information such as search keywords, number of documents to obtain and test to run. As an example, figure 6.3 shows a dialog that occurs at some point after selecting to run a test. This dialog is asking the user to select the clustering method; once selected, the test will proceed using the currently loaded test data, parameters and the selected clustering method.

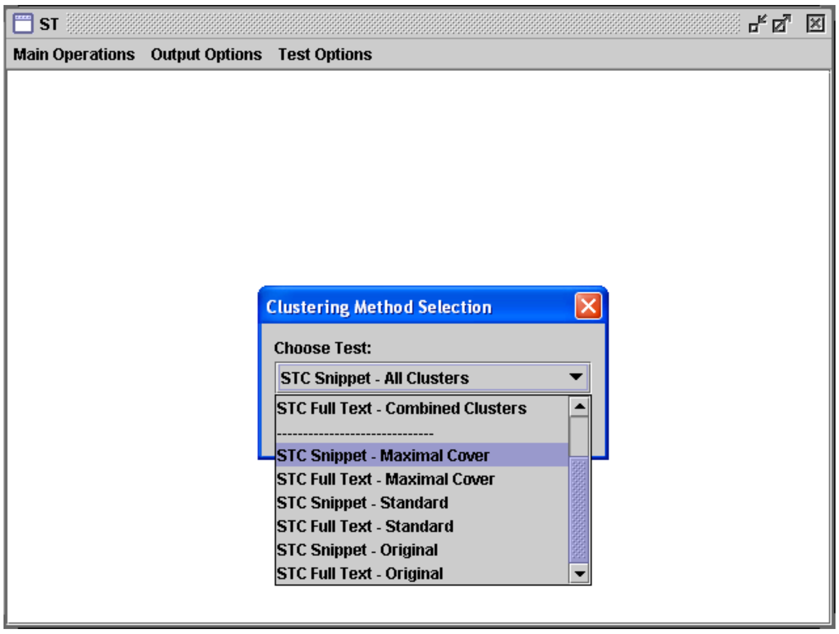


Figure 6.3: Clustering Method Selection

6.4.2 Output Options Menu

The second menu is the Output Options menu. This has six options and allows for the following test parameters to be set: Show Clusters, Show Cluster Document Lists, Show Cluster Flags, Show Cluster Phrases, Show Cluster Types, Run Batch Test. Table 6.1 provides details on what these parameters configure.

6.4.3 Test Options Menu

The final menu is the test options menu. This menu allows for the clusterer parameters to be altered at run time. From here the similarity constant can be altered, the maximal cluster covering lookahead set, the maximal cluster covering beta changed and whether non scorable words should be removed before building the suffix tree.

Table 6.1: Output Option Effects

Option	Effect
Show Clusters	Display Clusters in Output
Show Cluster Document Lists∗	Display list of documents with id, user assigned category and URL for each document in each cluster
Show Cluster Flags∗	Display number used in identifying cluster group membership during clustering
Show Cluster Phrases∗	Display the phrases that were used to generate the cluster
Show Cluster Types∗	Displays a 0 base clusters, 1 for joined clusters
Run Batch Test+	Puts the system into batch test mode, which means the next test run will be a batch of tests with different similarity constants ranging between the minimum given and 1 with step size given

(∗ Only applicable if Show Clusters is selected)
(+ minimum and step size are set using dialog boxes after selecting this option)

6.4.4 Sample Clustering Output

On a number of occasions, output is displayed in the main window to show the current progress of the system, for example, displaying which pages are currently being downloaded when obtaining the raw data. The more interesting output comes in the form of the results and evaluations of the different tests. Figure 6.4 demonstrates sample output from running a clustering algorithm on a search for 'jaguar' with Show Cluster Document Lists and Run Batch Test disabled and all other output option effects enabled. This example shows the following features:

Phrase Indicates the phrases used in forming the cluster.

Type 1 indicates this is a joined cluster, a 0 indicates this is a base cluster.

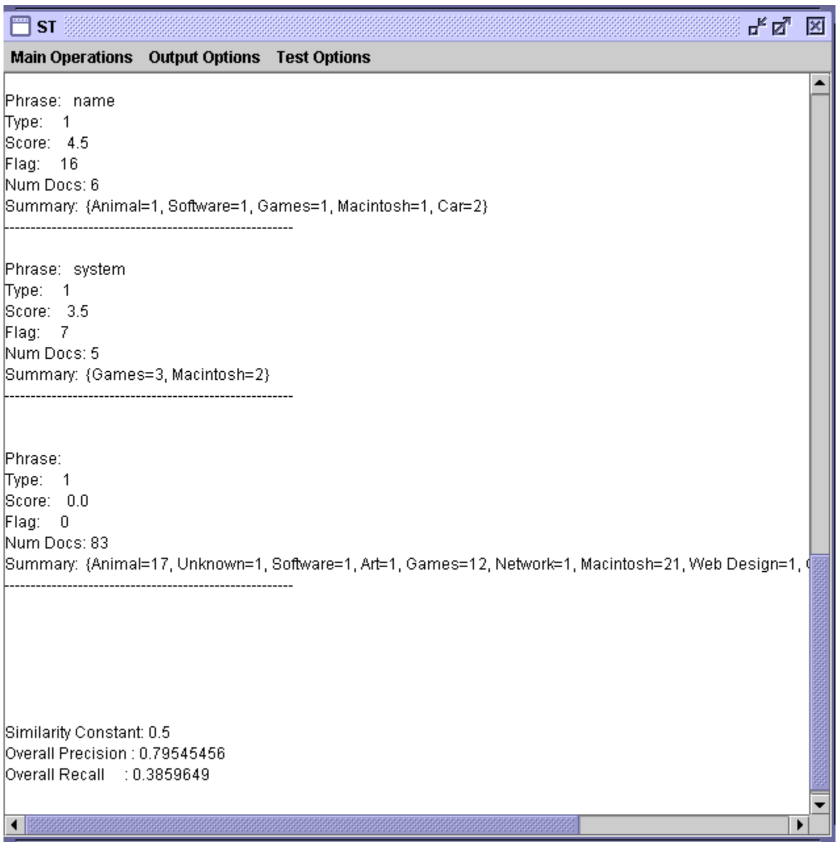
Score The score assigned to the cluster using the scoring algorithm.

Flag A number used in the identify group membership during clustering.

Num Docs The total number of documents in the cluster.

Summary The user assigned categories and the number of documents from each represented in this cluster.

The output ends with the overall evaluation of the current test where the overall precision and recall are displayed. Figure 6.5 shows the same set with Show Cluster Document Lists enabled in addition, which displays document lists for each cluster — showing the id, user assigned category and URL for each document in the cluster.



```
Phrase:  name
Type:    1
Score:   4.5
Flag:    16
Num Docs: 6
Summary: {Animal=1, Software=1, Games=1, Macintosh=1, Car=2}
------------------------------------------------------

Phrase:  system
Type:    1
Score:   3.5
Flag:    7
Num Docs: 5
Summary: {Games=3, Macintosh=2}
------------------------------------------------------


Phrase:
Type:    1
Score:   0.0
Flag:    0
Num Docs: 83
Summary: {Animal=17, Unknown=1, Software=1, Art=1, Games=12, Network=1, Macintosh=21, Web Design=1,
------------------------------------------------------





Similarity Constant: 0.5
Overall Precision : 0.79545456
Overall Recall    : 0.3859649
```

Figure 6.4: Short Sample Output

6.4.5 Missing Interfaces

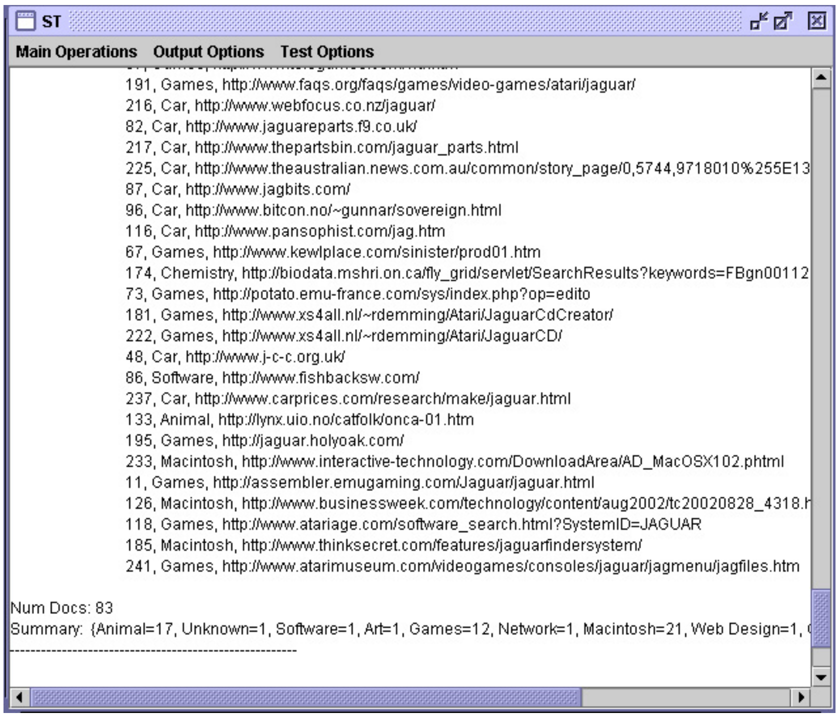
The other parts of the user interface are more rudimentary — changes to output options and algorithm options not listed in this section requires modifications to be made to the source code and user assigned categories must be entered directly into the database. These parts could have been incorporated into the user interface, but it was decided that the research aims of the project had a higher priority than the usability of the interface.

6.5 Testing Database

6.5.1 Database over File System

There is a need to store the raw data and semi processed data from test to test to maintain persistence across tests allowing for a fair comparison of the clustering methods. Also, storing this data, it only needs to be gathered from the internet once reducing bandwidth costs and reducing the length of processing time by storing semi-processed data. This allows for the duplication of efforts that would otherwise arise to be eliminated.

The two main choices for storing this data are either in a database or directly using the file system. For this project, it was decided to store the data in a database. This selection was made since using a database required no additional programming and many nice fea-



```
                191, Games, http://www.faqs.org/faqs/games/video-games/atari/jaguar/
                216, Car, http://www.webfocus.co.nz/jaguar/
                82, Car, http://www.jaguareparts.f9.co.uk/
                217, Car, http://www.thepartsbin.com/jaguar_parts.html
                225, Car, http://www.theaustralian.news.com.au/common/story_page/0,5744,9718010%255E13
                87, Car, http://www.jagbits.com/
                96, Car, http://www.bitcon.no/~gunnar/sovereign.html
                116, Car, http://www.pansophist.com/jag.htm
                67, Games, http://www.kewlplace.com/sinister/prod01.htm
                174, Chemistry, http://biodata.mshri.on.ca/fly_grid/servlet/SearchResults?keywords=FBgn00112
                73, Games, http://potato.emu-france.com/sys/index.php?op=edito
                181, Games, http://www.xs4all.nl/~rdemming/Atari/JaguarCdCreator/
                222, Games, http://www.xs4all.nl/~rdemming/Atari/JaguarCD/
                48, Car, http://www.j-c-c.org.uk/
                86, Software, http://www.fishbacksw.com/
                237, Car, http://www.carprices.com/research/make/jaguar.html
                133, Animal, http://lynx.uio.no/catfolk/onca-01.htm
                195, Games, http://jaguar.holyoak.com/
                233, Macintosh, http://www.interactive-technology.com/DownloadArea/AD_MacOSX102.phtml
                11, Games, http://assembler.emugaming.com/Jaguar/jaguar.html
                126, Macintosh, http://www.businessweek.com/technology/content/aug2002/tc20020828_4318.h
                118, Games, http://www.atariage.com/software_search.html?SystemID=JAGUAR
                185, Macintosh, http://www.thinksecret.com/features/jaguarfindersystem/
                241, Games, http://www.atarimuseum.com/videogames/consoles/jaguar/jagmenu/jagfiles.htm

Num Docs: 83
Summary:  {Animal=17, Unknown=1, Software=1, Art=1, Games=12, Network=1, Macintosh=21, Web Design=1,
-------------------------------------------------------
```

Figure 6.5: Full Sample Output

tures are already built in to the query engine. This trades off against a speed improvement which could potentially be obtained by using a custom solution. However, the benefit of a marginal improvement in data load times was minimal against the time of actually running the clustering and hence the database was selected.

6.5.2 Database Selection

MySQL Server 4.0.xx [20] was selected as the database of choice. Due to its price, cross platform availability and speed. MySQL is an open source and freely available database, which runs on most common operating systems and is known to be one of the most efficient (if not the most efficient) databases when using the MyISAM table format. The MyISAM table format lacks many transactional features such as transaction support, foreign key support and ACID compliance but these are not required in this system.

6.5.3 Database Design

The database is used for storing all the information gathered from the web pages and search engine such as raw page text and snippets. The database is also used for keeping track of each test search, the user assigned categories for pages and the cleaned text. Table 6.2 shows the important data stored in each table. The database is very simple, with only two tables. The option of splitting the document table and storing data at the word, phrase level was explored, but splitting at a finer granularity proved to be too slow and the benefits of splitting were not great.

Table 6.2: Key Database Table Fields

Table	Field
Search	
	Query
	NumberOfResults
Document	
	URL
	RawDocument
	RawSnippet
	CleanedDocument
	CleanedSnippet
	UserAssignedCategory

6.6 Summary

This chapter, distinguished the testing framework implemented in this project from the production framework. Each part of the testing framework was described, including details on why Java is used, why and how the MySQL database is used and what the testing interface allows the tester to accomplish.

The next few chapters delve deeper into the system and fully examine each of the 4 phases of the main clustering framework which were only outlined in this chapter. Phases 1 and 2, Obtaining and Cleaning Data are discussed in chapter 7, the main clustering algorithms used in phase 3 are detailed in chapter 8 and the evaluation and results are provided in chapter 9.

Chapter 7

Phases 1 & 2 - Preprocessing

7.1 Introduction

This chapter describes each step of the preprocessing phases (obtaining data and preparing data) which obtain the search results and prepare the raw data for use in the clustering phase. It also discusses issues in their design and implementation, and any advantages and disadvantages of using each step of the preprocessing phase.

7.2 Phase 1 - Obtaining Data

Phase 1 is all about obtaining the data used for clustering. It first obtains the search results (URL's and document snippets) to be clustered, and then obtains the actual web pages and content of these for use in the clustering.

7.2.1 Obtain Search Results

This preprocessing step involves obtaining the results of a search query. These results may be just a list of URL's, or may also include titles, snippets, full page text, or other information about the pages.

There are two main ways of tackling the problem of obtaining a set of search results for a query. The simplest is to use an existing search engine by querying the engine and using the result set it returns; the other is to develop a search engine from scratch.

The benefits of using an existing search engine are that development time is reduced, indexing time is eliminated and the implementation is very easy. However, because we have only external access to the search engine, there are drawbacks:

- There are limitations on the type of queries that can be performed. For instance, Google does not allow some combinations of site specific and link specific operators.
- There are limitations on how much information is available. For instance, Google will display at most 1000 results.

- The scope of the information is limited. For instance, Google does not display all in-linking documents even when this information is available.

Developing a new search engine to power this phase would get around these issues and allow full control over the type of queries that can be performed. Another advantage would be that preprocessing steps such as those discussed in the sections that follow could already have been performed at the time of indexing and hence improve the running times of the clustered search significantly. However, as this is to be used for research, and only off-line testing and comparison will be used, the real time response usually required of a search engine is not essential. There are also drawbacks to developing a search engine. There would be a significant amount of development time required to build a search engine that is remotely comparable to the existing enormous internet search engines. There would also be a considerable amount of time spent indexing the internet and this would also require a great amount of hardware and resources such as internet bandwidth which would cost a prohibitive amount.

For this project, an existing search engine will be used remotely. This decision is made easy by the fact that there is limited time and resources, making the development of a large scale search engine infeasible.

Another issue with using existing search engines is that snippets often fail to provide an adequate description. Sometimes snippets are dependent on the form of the query — context sensitive snippets. Context sensitive snippets represent only a small segment of the page, which can lead to misclassification. In other cases, there may be no snippet at all. Using these snippets as the basis of a clustering algorithm could be a significant disadvantage and make perfect results unobtainable. However, we cannot really consider this a drawback, as snippets from the search engine do not have to be used and the full-page text can be used instead to generate our own snippets for the purposes of this research. Alternatively, the combination of snippets from multiple search engines could provide a more detailed snippet, which would remove some of the problems with using snippets from a single search engine.

Implementation

For this projects we used the popular internet search engine Google [10]. In particular, we used the Google Web API `http://www.google.com/apis/` which provides a license to conduct up to 1000 automated queries per day.

The method for obtaining the search results is shown in figure 7.1. First the query is passed onto Google and the first x results are obtained. Then, for each of these results, the URL of the result and the context sensitive Google snippet is gathered.

It is usually considered that Google gives the best search results of any internet search engine and so selecting Google over other search engines was an easy decision. Also, Google's Web API provides for easy implementation and it is possible to immediately start getting the search results without writing any wrapper to extract the results. In addition to the problems of using an external search engine, a disadvantage of using Google's Web API is that it is slower than using Google directly. However, as it is against Google's terms of service to query the search engine automatically using any other method this is the way the queries must be made. However, before we were aware of Google's terms of service, we

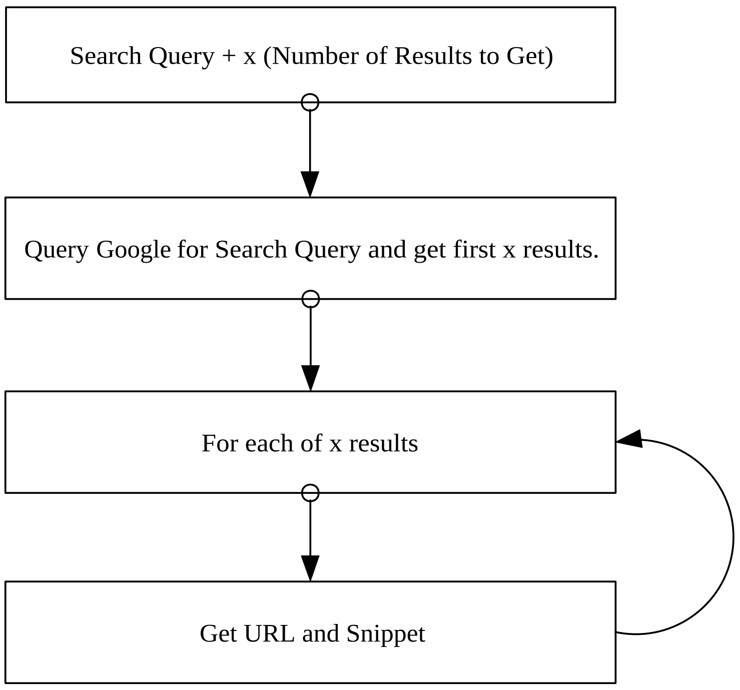

Search Query + x (Number of Results to Get)
Query Google for Search Query and get first x results.
For each of x results
Get URL and Snippet

Figure 7.1: Obtaining Search Result information from Google

wrote an information extraction program to extract search results from Google. This method was able to extract the first 300 search results from a search for 'jaguar' in less than a second compared to the approximately forty five seconds using the API. Clearly the Google Web API has lower priority than the standard web searches. This is not a significant issue as this is to be used for research only and a real web clustering search engine would likely be built on top of its own search engine.

There are further limitations of Google and the Google Web API. Firstly, when using the Google Web API, there is a limit of 1000 on any single result set. This is not a problem as the type of tests run for this research use only small test sets typically less than 500. The Google Web API also imposes a limit of 1000 queries per day. This is enough for this project and if link information were collected, it would be enough to get approximately one result set. Also the raw results are stored in the database backend, so we do not need to re-access Google to rerun queries later. The Google Web API also limits some kinds of searches, such as searching within a site using keywords. There are also the inherent limitations with Google itself. Google only deep indexes popular sites and so lesser known sites are unlikely to appear as frequently. Similarly a significant number of in-links will not be found using Google's in-link search. Also Google's page-ranking system means that sites with the most and best in-links end up at the top of the search results and so new sites with good content are missed out.

Future Extension

Even with all its limitations, Google is still the best search engine available for this research and the only real alternative would be to either develop a new search engine or to use multiple search engines.

In the ideal situation the search engine and the web page search clustering would be an integrated package since full access to the search engine would be beneficial. By being integrated with the search engine, all preprocessing for clustering could be done on the pages when indexing occurs. Also, the number of pages and type of page information stored could be tailored towards the requirements of the clustering algorithm. This would allow for both increased speed of clustering (as no preprocessing would be required) and clustering results could be improved by the tailoring. Clearly there are a number of benefits to building a search engine along with the clustering algorithm and this would be essential in a real world clustering search engine.

The other method which could be used more simply would be to use snippets from multiple search engines instead of just a single search engine. This would provide multiple sources of information and avoid some of the problems discussed with using snippets from a single source. Implementing this approach would not really require any significant change to the methods discussed in this report, but improvements would only be possible in methods that use the snippet information as their primary information source. Both this and the new search engine alternative have the potential to provide better results and both are worthy of future investigation.

Another future extension mentioned in the design considerations section was being able to generate a good snippet or description of a page. This would be useful for use in web clustering algorithms and such a function could also be used to improve on the methods used by the current search engines. However this is a separate research question and will not be considered further in this project, it would make a good future extension to this work and could either form a module in itself to extend on this project's implementation or it could be used to improve a standard search engine.

7.2.2 Page Grabbing

Page grabbing is the process of downloading pages over the internet, to obtain the actual page contents of the pages to be clustered.

Design

The first tradeoff to be made is between using a single-threaded HTTP client and a multi-threaded client. The trade off between the two is quite simple: the single-threaded version has a simpler implementation whereas the multi-threaded version is likely to be much faster as it can make use of all of the available bandwidth, avoids having to wait for timeouts and can use the bandwidth on other pages while waiting for the timeout on another connection. Timeouts are one reason why the multi-threaded version can use all the available bandwidth; others are that while saving documents there is time wasted, where the connection is idle when using the single-threaded version, and some servers may be slower than the available bandwidth.

Since the single-threaded version forms the basis of the multi-threaded version, it was decided to develop a single-threaded version and then extend it to a multi-threaded version if it was too slow.

The other considerations in the design of the page grabbing step are what to do about pages that do not exist, pages which had no snippet in the search engine, pages which use frames or redirects, and foreign language pages. Processing such pages should be automatic and the clustering results can be improved by removing pages which do not have the required information for clustering accurately. For instance, redirect pages do not contain the content of the pages they redirect to and hence do not accurately reflect the content an end user would see if they visited the page; the same is true for frames. Pages with either no snippet, no full text, or that return an error should removed as these pages could bias the results of evaluating the clustering methods by giving an unfair advantage to methods which use certain information. For example, link information. It would have been fairer if foreign language pages were removed; however, as this affects all methods equally, it was not seen as having a significant effect on the end results. Also, manually altering the result sets could be seen as introducing a human bias, which this project wants to avoid.

Implementation

The implementation used in this project is a basic multi-threaded HTTP client with a time-out of 10 seconds on any single page. The original single-threaded HTTP client was found to be incredibly slow — as suspected, bandwidth was simply not being used while waiting for page timeouts, saving to disk and using slow servers. The multi-threaded HTTP client solved all these problems and made full use of the available bandwidth. By having multiple pages simultaneously downloading; bandwidth can be used by active page downloads while other processes are waiting for page timeouts, bandwidth not used in connections to slow servers can be given to other connections and downloads can continue while another is storing data.

The pure speed advantage of the multi-threaded HTTP client over the single-threaded HTTP client is shown in table 7.1. For the comparisons, tests of the same size use the same test pages, there is no caching and single-threaded tests were run second to ensure that any end web server level caching benefits those tests. The connection used was a paradise.net.nz 10Mbit/1Mbit (download/upload), the timeout for a page not responding is set to 10 seconds and the test sets used are the first test size pages from the Google [10] search results for 'jaguar'. Also in the multi-threaded search case, a maximum of twenty-five threads were allowed. Note that the timeout will have the greatest impact on the multi-threaded case due to the significantly shorter download times and that at the end there may be a wait time for the remaining pages to timeout. This explains the non linear growth of the multi-threaded results — over larger test runs the maximum 10 seconds wait at the end for timeouts would become negligible and the results would tend towards a linear function as is evident with the single-threaded case. What is clear from the tests is that the slope of the single-threaded curve is significantly steeper and this was to be expected. Hence the multi-threaded version being used for the rest of this project.

The implementation uses filters to remove pages that do not exist, pages with either no snippet or no full text, and pages which use frames or redirects. Pages in foreign languages are left in, unless they are removed for one of the other reasons. Ideally the foreign language pages would be removed and the framed or redirected pages would be followed and used in place of the deleted pages. The method here was used as it was easy to implement, leaves the results unbiased towards any method, and leaves the results untouched by any human which may have introduced bias.

Table 7.1: Single-threaded vs Multi-threaded HTTP Client

Client	Test Size[a]	Time Taken[b]
Single-threaded	50	178.4
Multi-threaded	50	10.4
Single-threaded	100	341.5
Multi-threaded	100	13.9
Single-threaded	250	915.7
Multi-threaded	250	26.2

[a]Test Size is the number of pages downloaded
[b]Time Taken measured in seconds

Future Extension

To extend the filters to handle foreign language pages, there are two approaches: one would be to convert the pages into English which is currently intractable, the other is to simply exclude such pages, which could be done at the search engine level, or in a filter by using the domain name or by analysing the textual content. It was observed that from the data used in this project, that simply eliminating documents from certain top level domains (TLDs) would be sufficient and that filtering based on character models would work reasonably well as well. More sophisticated methods would be required for foreign language pages within typically English TLDs with languages that use standard English characters. Removing these pages is likely to improve the clusters as these pages are not classifiable using the information provided and without knowledge of the foreign languages. It is also likely that in an English clustered search engine that foreign language pages are not desired unless they can be translated.

Redirects and framed pages could easily be included by following the frames or following the redirects and using the end pages instead. Missing snippets can be acquired by building a new search engine with snippet generation, by using snippet generation directly on the full text or by combining the results of multiple search engines as discussed in the previous section. While including these pages would be of benefit to the end user and would make more of the internet includable in the clusters, the quality of the clusters and hence the evaluation metrics would not be affected. Hence, from a research standpoint, including these pages is not of great importance, though in a real world implementation these would need to be implemented.

7.3 Phase 2 - Preparing Data

Phase 2 involves preparing the data obtained in phase 1 for use in the clustering algorithms. This phase essentially tidies up the documents by removing HTML tags and similar non-content data in the cleaning step, by removing very common and unimportant words known

Table 7.2: Phase 2 Cleaning Breakdown

Step	Activity	Task
1.	Replace Line Breaks with Spaces	Cleaning
2.	Remove Comments	Cleaning
3.	Remove Scripts	Cleaning
4.	Extract Meta Keywords, Descriptions	Phrasifying
5.	Replace some HTML tags with Spaces	Cleaning
6.	Replace some HTML end tags with New Lines	Phrasifying
7.	Replace remaining tags with Spaces	Cleaning
8.	Remove Special HTML characters	Cleaning
9.	Replace some punctuation with New Lines	Phrasifying
10.	Replace some characters with Spaces	Cleaning
11.	Remove Adjacent New Lines	Phrasifying
12.	Remove non alpha-numeric characters	Cleaning
13.	Remove Stop Words	Removing Stop Words
14.	Remove leading and trailing whitespace	Phrasifying
15.	Remove plain number phrases	Cleaning
16.	Apply Porter Stemming Algorithm to each word	Stemming

as stop words in the second step, by stemming the remaining words to find their root form and hence allowing words with similar meaning to be identified as the same, and finally by splitting the strings into phrases using the locations of the HTML tags removed and the locations of punctuation. Table 7.2 shows a breakdown of the steps performed in this phase to clean the data and the following sections consider these as to how they relate to each task. This phase has a large effect on the quality of the clustering and the ramifications of the design decisions associated with each task are discussed in the sections below.

7.3.1 Cleaning and Phrasifying

The cleaning and phrasifying share a lot in common and are thus combined together into this single section.

Cleaning is the process of removing elements from the HTML documents that are of no use to the clustering. These include elements such as comments, scripts, some HTML tags, special HTML characters and plain number phrases. The Cleaning step corresponds to steps 1, 2, 3, 5, 7, 8, 10, 12 and 15 from table 7.2.

Phrasifying is the process of splitting the sequence of words up into a sequence of phrases of words. This is for use in algorithms which work on phrases, such as the suffix tree clustering algorithm discussed in the next chapter. Phrasifying corresponds to five steps in the overall data cleaning algorithm: steps 4, 6, 9, 11 and 14, which extract meta keywords and descriptions, replace some HTML end tags with new lines, replace some punctuation with new lines, remove adjacent new lines and remove leading and trailing whitespace.

Design

The hardest part of the cleaning and phrasifying steps is determining which HTML tags and characters should be left for splitting into new phrases as part of the phrasifying step and which should simply be removed part of the cleaning step.

The obvious choices for the cleaning step are to remove comments and scripts which are not visible to the user. However, it may be possible to extract some information from scripts, such as information from dialogs that may be displayed to the user. Similarly image tags can be removed, though a more advanced solution would be to determine a description for what the image represents using advanced image processing techniques.

For phrasifying, the hardest decision is what to do with certain font tags. It may be that sections in the middle of a phrase are highlighted in bold but they still form part of the phrase; if phrases are split at font tags, these end up being split into separate phrases. An investigation on where these should be split would be required to determine the optimal splitting point.

Implementation

The current implementation removes line breaks, comments and scripts, a variety of HTML tags such as image tags, tags not removed after some of the phrasifying steps, special HTML characters, non alpha-numeric characters, and any plain number phrases left at the end.

For phrasifying, a variety of table tags, meta tags, image tags, form tags, list tags and font tags are used for splitting into new phrases; other HTML tags are simply removed as part of the cleaning step discussed earlier. Some punctuation characters such as full stop, comma, exclamation point,colon, etc are used for splitting. Other non alpha-numeric characters are simply removed as part of the cleaning step.

Future Extension

There is significant room for investigation into what is an appropriate level of splitting, on what items splitting should occur, and how the choice of where to split affects the end clustering results. In particular, further investigation on the effect of splitting on font tags may be desirable.

A more interesting research area which could be investigated would be that of an intelligent cleaning and splitting agent which would clean and split on a document by document basis. This would be quite a challenge and could make an entire research project in itself, and hence is outside the scope of this project, but would definitely be worthy of future investigation.

7.3.2 Removing Stop Words

Stop words are words which frequently occur in text on any topic, and have no meaning by themselves. Typically stop words are conjunctions, determiners or prepositions such as ‘also’, ‘each’, ‘the’ and ‘through’, but this is not always the case — common web terminology and words frequently found on the web are often classified as stop words as well. This means that words like copyright, home and click may be considered stop words. Removing stop words corresponds to step 13 from table 7.2.

Design

The main benefit of removing stop words before clustering, is to speed up the clustering step. For suffix tree clustering this is achieved by reduced phrase lengths leading to shorter build times for the suffix tree. Also, memory requirements are reduced since with stop words removed there will be fewer distinct phrases. This also means that documents with semantically equivalent but different phrases in the original are more likely to be matched in the transformed text and so improve the clustering results. Other clustering methods which consider page content or matching between page content will have similar benefits arising from the shorter documents and phrases. An indirect benefit of shorter documents is that database queries will be faster in cases where there is no index.

The advantages and disadvantages of removing stop words closely parallel the advantages and disadvantages of stemming, described in the next section.

The main disadvantage with removing stop words is the loss of information. For instance the well known phrase 'To be or not to be' [24] would be completely eliminated under this project's implementation of stop words. Individually the words have little to no meaning, but when combined as a phrase they are meaningful. Since there are certainly situations where information will be lost, there is a tradeoff between the value of the information and the value of the speed and improvements gained by removing the words. This tradeoff is adjustable by adding or removing words from the stop word list.

One such adjustment is whether numbers should be on the stop word list. If they are, then movies and their sequels would likely end up in the same cluster. So perhaps numbers should not be on the stop list. On the other hand, many numbers are irrelevant and a great speedup would be given in those cases. Such decisions are difficult to make without testing and comparison and a full investigation would be required to determine whether numbers should be removed or not. The effectiveness of the stop word preprocessing would also depend to some extent on the clustering algorithm and so no single stop word list will be ideal in all situations.

Another method that can be used in addition to stop words, is to remove words that occur in too many, or too few documents. The idea is that if a word occurs in too many documents then that word does not help distinguish document topics and if the word appears too few times, then it will lead to very specific clusters being found. Potentially, statistics from the entire document set could be used to perform this step as part of preprocessing. Another possibility is to perform this type of adjustment at a later stage, for example as part of the clustering in phase 3. This would allow for the thresholds on too frequent and too infrequent to be set dynamically and the word frequencies could be based on just the documents to be clustered rather than the whole document set. Doing this step as part of the clustering would also allow for fine tuning. For instance, if there were a large number of documents, then different degrees of stop words could be removed; if the threshold for infrequent words was set high, then only very high level topics would emerge, and if the threshold was set lower, then more lower level topics would emerge. This would allow for some kind of hierarchical clustering to be performed by applying the clustering recursively and adjusting the cutoff parameters at each level, or perhaps automatically by using percentages of the total number of documents and thus allowing the reduced document set to change the cutoffs automatically as the document sets shrink when applying the clustering to sub-clusters.

Implementation

The stop words preprocessing step removes all stop words from any text, this includes snippets and full web page text of original. Both common words and web specific words have been included in the stop word list, the full list of stop words can be seen in appendix A.

Some limited investigation was performed into removing common and infrequent words in the preprocessing phase based on thresholds of around 0 - 5% for infrequent words and 30 - 60% for common words.

Results

It was found that removing frequent and infrequent words based on frequency in the preprocessing phase caused issues with the phrases becoming too general and too many completely different phrases ended up being represented by the same sequence of words and hence the resulting clusters were of poorer quality than they would be otherwise. One possible solution would be to replace these removed words by a series of stars, so that at least phrases of the same lengths would match and thus eliminate matching between phrases of different lengths which only infrequent and common words removed. Some investigation was also conducted into whether applying these steps at the start of clustering and only on the single result set rather than the whole set of searches would be useful. It was found that it did decrease the quality of the clusters, but that also the size of the clusters were also affected. This suggests that this method could be used as part of a hierarchical implementation of clustering to allow for control on the granularity of the clusters found at different levels. However, since the precision was also reduced, further controls such as the star-method or further integration with the clustering would be required to make sure the precision doesn't degrade.

Future Extension

There is the possibility for research to be conducted on how the stop list affects the quality of the clustering. Possible avenues for investigation include: altering the set of words and seeing how adding more or removing words affects the clustering, removing the stop word removal step altogether, and further research into removing words based on their frequency in the document set in the preprocessing phase or the clustering phase.

7.3.3 Stemming

Many words are formed by extending a common root word, or stem. Take the example from [22], 'connect' is the root word or stem for 'connect', 'connected', 'connecting', 'connection' and 'connections'. Stemming is the process of transforming a word into its root form or stem. For instance, all of the words 'connect', 'connected', 'connecting', 'connection' and ‘connections’, should be transformed to ‘connect’ after applying stemming. Stemming corresponds to step 16 from table 7.2.

Design

Stemming is useful as a preprocessing step in web page search clustering since words with similar meaning will all be described in the transformed document by a single word. This means that pages with original phrases that do not match, but which have similar meaning will now have an improved chance of matching and will thus increase the number of matched phrases between pages with similar content. This provides the benefit of increasing the likelihood that two pages with the same topic will end up in the same cluster.

There is one point of adjustability to consider with regard to stemming and its application to web page search clustering. The level of stemming provided will directly affect the quality of the clusters. With too much stemming, information will be lost and pages which are not related may be clustered together. In contrast, if there is too little stemming then pages with similar content having semantically equivalent but different wording may not end up in the same cluster. Also, there will be more distinct words and phrases which will slow down clustering and increase the memory requirements for clustering algorithms.

Another impact of the amount of stemming is related to the ideas for hierarchical clustering discussed in the stop word removal step in section 7.3.2: the level of stemming could perhaps affect the granularity of the cluster topics in a similar way to the cutoff levels for the frequent and infrequent stop words.

Implementation

In this project the Java version of the Porter stemmer [22] from `http://www.tartarus.org/~martin/PorterStemmer/` has been used.

> ‘The Porter stemming algorithm (or Porter stemmer) is a process for removing the commoner morphological and inflexional endings from words in English.’ [22]

Using this code provides the benefit of reduced development time by reducing the code that needs to be written, tested and debugged in order to implement stemming, on the downside there is less flexibility. Of course if more flexibility is required, a different stemming algorithm or a modified Porter stemmer could be substituted.

Future Extension

To test the effect of the level of stemming on resulting clusters would require having greater control over the level of stemming than is provided by simply implementing the Porter stemmer. Currently the only level of control is to either have the stemmer on or off. With a finer level of granularity, better results may be obtainable and future research could investigate the effect of the level of stemming on cluster results. Future research could also investigate whether the amount of the stemming has a similar effect on the granularity of the clusters as changing the amount of infrequent and common stop words does.

7.4 Summary

In this chapter the two preprocessing phases — obtaining data and preparing data — have been broken down further and the design and implementation considerations and details have been discussed, the impact of these decisions on the later phases of the system has also been considered and future improvements which could be made have been suggested. In the following chapter the most important phase — clustering — is considered, this phase uses the preprocessed data obtained from running the data through the phases discussed in this chapter.

Chapter 8

Phase 3 - Clustering Methods

8.1 Introduction

Building on the discussion and analysis of clustering methods in chapter 4, this chapter describes the algorithms implemented as part of phase 3 of the overall framework. The second and third stages of the suffix tree clustering algorithm, building the suffix tree and clustering are described and the implemented algorithm and some analysis is given for each. One of the main contributions of this project, the fourth stage extension to the suffix tree clustering algorithm, is detailed, analysed, and explained. Finally this project's work on identifying irrelevant documents is described.

8.2 Clustering Implementation

8.2.1 STC Stage 2 - Building the Suffix Tree

As introduced in section 4.4, this project uses an adapted version of the naïve suffix tree construction method.

Algorithm - Adding documents to Suffix Tree

```
1. For each document
2.      For each phrase
3.           For each progressively shorter word level suffix of the current phrase
4.                Add phrase to root of suffix tree
```

Algorithm - Adding a phrase to node of Suffix Tree

1. If the phrase has word length of zero, add the current document to the base-document-set of the current node and return to processing phrases.
2. Find the edge from the node whose first word prefix matches the first word prefix of the phrase. (If such an edge exists, there is only one such edge.) If no such edge exist,

create an edge from the current node to a new node, label the edge with the current phrase, add the document to the base-document-set of the new node, and return to processing phrases.

3. Find the maximal word length phrase that prefixes both the edge and the phrase.

4. If the common prefix, phrase and edge all have equal word length, add the current document to the child node.

5. Else if the common prefix has word length equal to the edge, add the suffix of the phrase that excludes the common prefix to the child node.

6. Else, remove the edge, add a new edge labeled with the common prefix from the current node to a new node, add an edge labeled with old tree edge minus common prefix from the new node to the child node and add the suffix of the phrase excluding the common prefix to the new node.

The adapted basic algorithm used in this project is almost identical to the naïve one, but to alleviate the $O(b)$ time of finding a matching edge (where b is the branching factor of the node), each node has a hash index on the first word prefixes of each edge from the node which allows this step to be performed in $O(1)$ time. This optimisation reduces the cost of inserting a phrase from the naïve algorithms $O(ndb)$ to $O(nd)$, where n is the word length of the phrase, d is the maximum depth of the tree and b is the maximum branching factor of the tree. Note, b is generally much larger than n or d. As a speed improvement for the third stage of the algorithm, all documents attached to any descendant node of a node are also attached to the node.

8.2.2 STC Stage 3 - Clustering

The third stage clustering implementation used in this project is based on the discussion of suffix tree clustering in [39].

Algorithm - Single Link Base Cluster Clustering

1. Each node of the suffix tree is turned into a base cluster consisting of the documents in its document set (i.e., all the documents that contain the sub-phrase in the label of the node).

2. Each base cluster is assigned a score based on the length of the sub-phrase and the number of documents in the base cluster. The score is defined as:

$$score = \begin{cases} 0.5 & \text{if } |\textit{sub-phrase}| = 1 \\ 6 & \text{if } |\textit{sub-phrase}| > 5 \\ |\textit{sub-phrase}| & otherwise \end{cases}$$

3. The clustering similarity graph is constructed by placing a link between a pair of base clusters when the number of documents in common between the two base clusters divided by the number in one of the clusters is more than the similarity constant and the number of documents in common divided by the number in the other cluster is also more than the similarity constant.

4. The connected components algorithm is run on this graph. Each connected component is then concatenated to represent one combined output cluster.

5. The score of each output cluster is computed.

6. The clusters are then ordered by score and returned

Scoring and Outputting Combined Clusters

In this project clusters are scored using one of two methods. The first is the simple method used in [39], [38] and [6], where the score of a combined cluster is defined as the sum of the scores of the base clusters contained in the combined cluster. There is one major problem with this simple method. Consider the case where a large set of clusters with very similar document-sets is joined, the score will quickly grow very large, but this joined cluster is not significantly better than any one of the clusters.

The new scoring method developed in this project fixes this problem by re-weighting the scores based on the number of documents added by a cluster. In this project the scoring method is defined as:

S is the score of the currently picked clusters, D is the set of distinct documents in the currently picked clusters.

While there are unpicked base clusters

1. Pick a base cluster B.
2. Let b be the number of documents in B that are not in D.
3. Update the score S by adding $\frac{b}{b+|D|} \cdot score(B)$.

An observation is that the new scoring method is inconsistent, in that scores depend on the order in which base clusters are considered. This is a problem with the currently implemented new scoring method. However, there is an easy fix for this: a better and consistent way of implementing a method that achieves the goal of the new scoring method. Such a method is:

1. For each base cluster c with score s in the combined cluster, assign the cluster-document-score $\frac{s}{|c|}$ to each document in c.

2. Assign each distinct combined cluster document a document-score equal to the sum of the document's cluster-document-scores divided by the number of base clusters the document is a member of.

3. The score for the combined cluster is then the sum of the document-scores for each distinct document in the combined cluster.

In spite of the problems with the implemented version of new score, the method outperforms the simple scoring function as shown in chapter . This does not mean that the consistent scoring method will have the same results, but due to the similarities between the two, the flawed method could be considered an approximation to the consistent version and so the results found regarding the implemented version will most likely mirror the results that would be found if the consistent version were to be implemented.

Once scored there are two possibilities. One is that the top n results are returned, which allows the highest scoring clusters to appear at the top. This is the method used in [39] and [38]. The other method developed in this project is to carry out a fourth stage and compute the maximal cluster covering of n clusters to find the n clusters that should be returned. The combination of simple scoring method with returning the top n is referred to as the original method and emulates the method used in [39] and [38]. The combination of the new scoring method with returning the top n is referred to as the standard method. The maximal cluster covering method can be combined with both of these approaches, but unless otherwise stated it is combined with the new scoring method.

8.3 STCE Stage 4 - Maximal Cluster Covering

The Maximal Cluster Covering stage is an extension of the Suffix Tree Clustering algorithm, forming an algorithm this project terms STCE for Suffix Tree Clustering Extended. It reduces a large set of clusters down to a more user manageable size of no more than ten. This entails searching the solution space for the set of clusters of most use to the user, based on two heuristics: Minimal Overlap — minimising the number of times a document will be considered and Maximal Coverage — maximising the number of topics represented, which reduces the amount of searching required.

Picking a limit of ten clusters is rather arbitrary; the method could just as easily return K clusters. Note that it is at most ten that are returned — the method may return four or five if that seems like a better set to give back. Ten was selected as it is not so large that a user cannot comprehend and select an appropriate cluster, yet it is large enough that it can represent most higher level topics contained in the result set.

These heuristics were designed to provide the most useable clustering. If there is too much overlap between the clusters then the documents have not been distinguished enough and the user has to sift through the same documents multiple times. For obvious reasons we would like to successfully cluster all documents and hence the goal of clustering most of the documents by having maximal coverage.

8.3.1 Algorithm

The maximal cluster covering algorithm is formulated as a search problem and standard search techniques are applied and solve the problem in an efficient manner. The following search space, search method, and heuristic scoring function are defined for this problem:

Search Space The set of all sets of clusters with ten or fewer elements.

Search Method A best first method using branch and bound pruning to keep the search efficient and a small look-ahead to improve the results.

Heuristic Function $(\beta + 1)D - \beta C$, where C is the number of documents in the solution, D is the number of distinct documents in the solution and β is the constant controlling the trade off between overlap and coverage.

Table 8.1: Maximal Cluster Selection

Number of Clusters Clusters	Minimum Score (% of highest score)	Maximum Fraction Deleted (% of clusters)
< 20	-	-
< 50	20	25
< 100	20	40
< 250	25	50
< 500	30	80
< 1000	30	90
≥ 1000	30	95

Algorithm Details

The implementation starts by removing clusters whose score is not considered of high enough quality. This may mean dropping the bottom x% or dropping those with a score less than some fraction of the highest score. It may also require that a minimum number of resulting clustering still exist. The thresholds used depend on the total number of clusters. Table 8.1 shows the minimum score required (as a fraction of the highest score) and the maximum fraction of the clusters that can be deleted.

Since the search space will typically be very large, an efficient search algorithm is required. Furthermore, since there is no correct answer to the problem we are trying to solve, we do not mind if the solution found does not maximise the heuristic function. The heuristic function is merely to guide the search in the right direction; once in the general vicinity of the optimal answer, the heuristic function does little to effectively distinguish states.

The goal of the search is to find the set of at most ten clusters which maximises the heuristic function. To do this search a best first search method is used that builds up the solution one cluster at a time, each time picking the cluster that increases the heuristic function value the most. The only problem with this is that there are situations where quite poor fittings will be found. For instance if there is one fairly large cluster then this will be picked first and then it may be the case that no other clusters can be picked, so the search ends here, even though there were a set of three clusters that covered more documents with little overlap. Without any other information this second cluster set is generally the preferred choice. To allow the search method to find these type of solutions some look-ahead is implemented and an arbitrary amount of look-ahead n can be set.

To further optimise the method, a branch and bound technique is used to prune the search tree. Also various other speed optimizations were implemented, such as removing documents that are in the solution set and adding the number removed to an overlap variable in each cluster, and removing clusters which would not improve the current solution if added.

Table 8.2: Effects of Beta Constant in Heuristic Function

Beta	Formulae	% of distinct documents required for a cluster to be selected
0.00	D	No penalty for overlap
0.25	$1.25D - 0.25C$	At least 20%
1.00	$2D - C$	At least 50%
2.00	$3D - 2C$	At least 66.67%
10.00	$11D - 10C$	At least 90.91%

Heuristic Function

The selection of the clusters is determined by the use of the heuristic cluster covering score. This is determined by the heuristic function which takes a set of clusters and returns the score for the clustering. The function needed to incorporate the goals of the covering, namely that the overlap be kept small and the number of documents missed by kept small, combined with the limit of ten clusters. This means there had to be a trade off between the number of documents missed and the amount of overlap. The heuristic function also had to allow for this trade off curve to be fairly smooth.

If C is the number of documents in the solution, D is the number of distinct documents in the solution and β is the constant controlling the trade off between overlap and coverage, then the heuristic function is $D-(\beta(C-D))$, which is the number of distinct documents less β multiplied by the number of duplicate documents. This can be rewritten as $(\beta+1)D-\beta C$.

Table 8.2 shows examples of the heuristic function for different β values and what it says about the type of resulting cluster sets it will result in. The % of distinct documents required for a cluster to be selected refers to the % of documents in a cluster that are not already in the solution. If this minimum % of documents new to the solution do not exist in a cluster, the cluster does not need to be considered. As $\beta \rightarrow \infty$, the percentage of documents that must be distinct approaches 100%.

8.3.2 Example Application of Maximal Cluster Covering

Consider the case with four clusters shown in the figure 8.1, a two step look-ahead and a beta of one. In this case the grey circle will not be in the solution as it overlaps with the other three clusters too much, leading to less than 50% of it's documents being distinct. It cannot be selected on the first step either due to the look-ahead.

8.4 Irrelevance Identification

One of the largest problems with the single link clustering method is that unwanted chaining effects can occur. Several possible solutions to the chaining problem were suggested in section 4.4.4. A close variant of the chaining problem is the irrelevant document problem, which has less impact on the results of the clustering than the chaining problem can have,

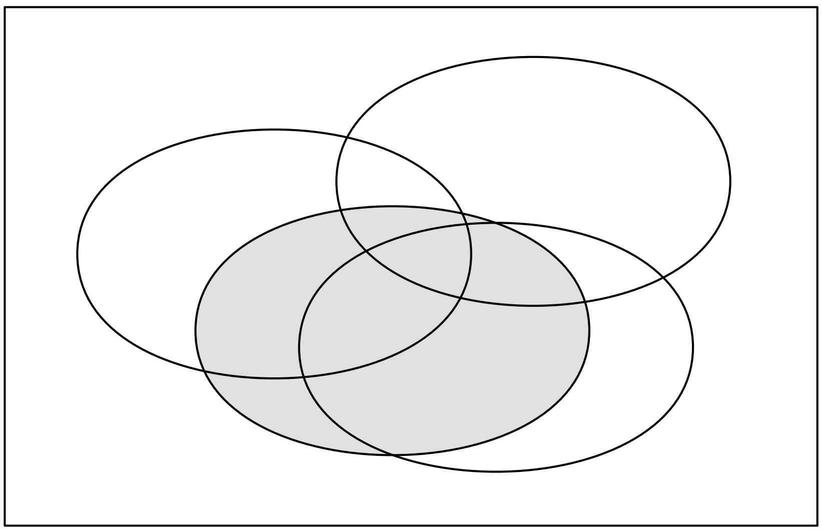


Figure 8.1: Maximal Covering Example

but is potentially harder to identify and fix. The irrelevant document problem occurs when totally unrelated documents occur in the search results due to matching the key words. These usually match one or two other documents and often end up in base clusters with other documents. The problem arises in that once a document is in a base cluster, the later clustering method, in our case, the single-link clustering, assumes that the base clusters solely consist of related documents. This means that once unrelated documents are put together in a base cluster, they end up together in the final clusters too.

Two solutions to this problem were investigated. The first, which was not implemented, was to replace the third stage of the suffix tree clustering algorithm with a radically different method. Several such methods were introduced in chapter 4. A further method that was investigated was to consider implementing a clustering algorithm such as single link clustering on the documents themselves rather than the base clusters and to use the number or fraction of base clusters two documents share in common to form the similarity measure.

The other method considered was to try to eliminate the irrelevant documents earlier in the clustering, back when the suffix tree was being constructed. The type of documents that cause the irrelevance problem are typically quite distinct from the other documents and the topic to which they actually relate is often represented in the document set by typically only the one document and in some cases just two or three. The methods considered for identifying these irrelevant documents are based on what we term base cluster analysis. Two different methods were implemented: (1) number of base cluster occurrences, (2) number of base cluster occurrences weighted by base cluster size multiplied by base cluster score. Several other methods were also considered, but not implemented, including (3) number of base cluster occurrences weighted by base cluster score, and (4) average size of base clusters in which document occurs.

8.5 Summary

This chapter discussed two of the main contributions of this project, the maximal cluster covering extension to suffix tree clustering and the irrelevant document identification methods. Also presented were implementation details of the second and third stages of the suffix tree clustering algorithm.

The building of suffix trees was outlined as a modification of the naïve method using a hash index to achieve order n^2 time complexity. The third stage clustering was outlined as the standard single-link clustering method.

The maximal cluster covering extension to the suffix tree clustering algorithm was introduced as a search algorithm for finding a reasonable set of clusters to display to the user that best represent the result set. Minimal Overlap and Maximal Coverage were introduced as heuristics to determine cluster set quality and an enhanced best first search was described to find good solutions in reasonable time.

Finally, irrelevance identification was discussed and the several methods based on base cluster analysis were described as potential methods of identification.

Chapter 9

Phase 4 - Testing and Results

9.1 Introduction

This chapter applies the evaluation method introduced in chapter 5 to evaluate the effectiveness of the evaluation method, the maximal cluster covering extension and the irrelevant document identification methods. The test data, test setup, results and analysis are provided and explained for each test and any conclusions are drawn.

In the closing stages of this chapter the best variation of the system developed in this project is compared against Grokker — a commercial clustering search engine which appears to provide very nice results from a users point of view. An evaluation of Grokker is given along with details on how Grokker compared to the method developed in this project.

9.2 Types of Evaluation

Two types of evaluations were performed as part of this project. Evaluations of the evaluation method and evaluations of the clustering contributions — maximal cluster covering and irrelevance identification.

The evaluation method was evaluated by duplicating experiments performed in past research and determining whether the same conclusions could be drawn. This further confirms the findings of past research and provides room for evaluating the evaluation method and the system in general.

The two main clustering contributions — maximal cluster covering and the irrelevance identification implementations — were tested to determine the benefit, if any, of these methods.

9.3 Data Sets

Testing was performed using two data sets, corresponding to searches for 'jaguar' and 'salsa'. Both raw data sets and both randomly relabeled data sets are available online at `http://www.danielcrabtree.com/clustering/rawdata.zip` in the form of an SQL table dump. Each data set contains the raw data and the preprocessed data and either the manually assigned user assigned categories or the randomly assigned user assigned categories as used by the two statistical significance data sets. One important detail about the data sets is that the snippet data includes only the snippets returned by Google [10] — the page titles are not included. This is in contrast to the snippet methods in other research which include the page title. The page title is a valuable source of information and should have been included in the snippet data. This oversight occurred early on when data was captured and this information was unfortunately forgotten.

9.4 Results

In this section, the results and evaluation of this project are presented. Firstly, the statistical significance of each method is established for use as a basis in evaluating and testing. Next, a test performed in prior work, showing that clustering using full text performs better than using only snippets, was reproduced. Results are presented for both data sets, and is provided to evaluate the evaluation method itself. The maximal cluster covering contribution is then evaluated on the jaguar data set, firstly this added stage is compared with the standard cluster top n scoring method and for comparison to past work it is also compared against the original top n suffix tree cluster scoring method — the scoring methods are as outlined in chapter 8. The effect of varying the similarity constant and β is then investigated, again using the results from the jaguar data set. Next, the effect of different depths of lookahead in the maximal cluster covering lookahead is shown and discussed, and finally the irrelevance identification techniques are tested on the jaguar data set. In all tests the same conclusions could be drawn from both 'jaguar' and 'salsa' data sets, hence 'salsa' tests have been excluded from many of the results displayed below.

In the following results, 'maximal clustering covering method' refers to cases where the maximal clustering covering stage is used to select the clusters to be output; the 'standard method' refers to the case where the standard top n scoring method introduced in chapter 8 is used, the 'original approach' refers to the case where Zamir's [38] original top n scoring method introduced in chapter 8 is used.

In some tests F-measure has been used to simplify the display and reduce the comparison of results from two variables (precision and recall) to one variable (F-measure). The standard F-measure of: $F(T) = \frac{2PR}{P+R}$ is used, where P is the precision and R is the recall.

9.4.1 Statistical Significance

The user assigned categories to each document were randomized and each combination of settings (similarity constant and maximal cluster covering beta) and methods (maximal clustering covering, standard or original) were applied to each data set (jaguar and salsa).

In the case of the jaguar data set there were a total of 210 documents. After randomization, 172 of these were assigned to categories other than their true category.

Table 9.1 shows the 95th percentile measurements across test cases for many parameter values for both randomised data sets. As shown in table 9.1 testing on both result sets and using both snippet or full text methods can give quite high precision and recall on the random data. This means that when using snippet data, up to approximately 45% precision and 25% recall can be achieved by a random method and in the case of full text data, up to

Table 9.1: Statistical Significance - Performance achieved on random data

Data Set	Data	Precision / Recall (%)
jaguar	Snippet	43 / 25
jaguar	Full Text	40 / 45
salsa	Snippet	44 / 25
salsa	Full Text	40 / 45

approximately 40% precision and 45% recall can be achieved. Hence, any method must be able to produce a higher level of precision and recall to be achieving anything better than random. Note that the full text data results in a higher recall, even on random data, because more documents end up in the resulting ten clusters when using the full text data compared with the snippet method.

9.4.2 Snippet vs Full Text

To help evaluate the evaluation method and to reevaluate a conclusion of Zamir [39], the clustering method with the maximal cluster covering extension and two step lookahead was tested on both the snippet and full text versions of the 'jaguar' and 'salsa' data. Different levels of the similarity constant were used to provide a fair comparison and a beta of one was used for the maximal cluster covering component.

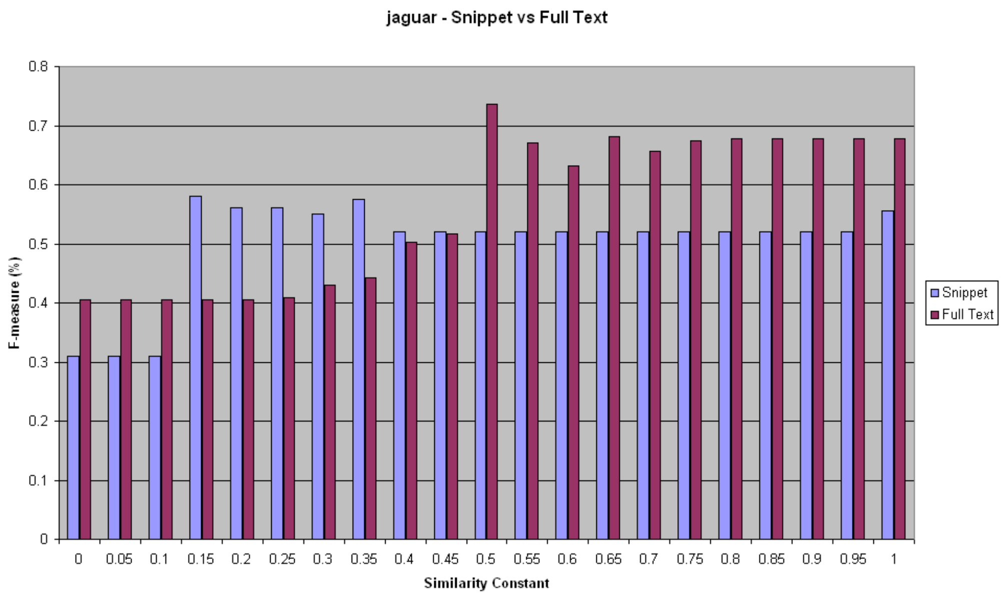


Figure 9.1: Jaguar - Snippet vs Full Text - F-measure

Figures 9.1, 9.2 and table 9.2 display the results of comparing the full text method against the snippet method on the 'jaguar' data. From the F-measure it can be seen that quite clearly the full text method can outperform the snippet method: with a similarity constant of between 0.5 and 1, the full text results are better than the snippet results at any similarity constant.

Table 9.2: Jaguar - Snippet vs Full Text

Similarity	Snippet (%)		Full Text (%)	
Constant	Precision/Recall	F-measure	Precision/Recall	F-measure
0.00	41/25	31	37/45	41
0.05	41/25	31	37/45	41
0.10	41/25	31	37/45	41
0.15	80/46	58	37/45	41
0.20	80/43	56	37/45	41
0.25	80/43	56	37/46	41
0.30	79/42	55	39/48	43
0.35	83/44	57	38/53	44
0.40	80/39	52	43/60	50
0.45	80/39	52	45/60	52
0.50	80/39	52	67/82	74
0.55	80/39	52	59/77	67
0.60	80/39	52	61/65	63
0.65	80/39	52	67/70	68
0.70	80/39	52	65/67	66
0.75	80/39	52	68/67	67
0.80	80/39	52	67/68	68
0.85	80/39	52	67/68	68
0.90	80/39	52	67/68	68
0.95	80/39	52	67/68	68
1.00	86/41	56	67/68	68

Table 9.3: Salsa - Snippet vs Full Text

Similarity	Snippet (%)		Full Text (%)	
Constant	Precision/Recall	F-measure	Precision/Recall	F-measure
0.00	35/40	37	36/41	38
0.05	35/40	37	36/41	38
0.10	36/43	39	36/41	38
0.15	43/53	48	36/41	38
0.20	45/58	51	36/41	38
0.25	47/56	51	36/41	38
0.30	48/56	52	36/51	42
0.35	41/44	43	39/57	46
0.40	42/45	43	42/47	44
0.45	42/45	43	39/61	48
0.50	41/43	42	50/68	58
0.55	41/43	42	51/71	59
0.60	41/44	42	56/53	54
0.65	41/44	42	56/64	60
0.70	41/44	43	67/63	65
0.75	41/44	43	57/58	58
0.80	41/44	43	73/67	69
0.85	41/44	43	73/67	69
0.90	41/44	43	69/64	67
0.95	41/44	43	69/64	67
1.00	42/41	41	70/64	67

The best results are with a similarity constant of 0.5 giving an F-measure of 74%. The best results with the snippet method come with a similarity constant in the range 0.15 to 0.35, and peaks at 58%. Turning to the precision and recall (figure 9.2), the picture becomes somewhat different. Snippets uniformly have better precision than the full text methods, but the recall of snippets is not much better than the baseline of 25% found from the statistical significance tests, with the best recall for the snippets being 46%. The best performance of the full text is with a similarity constant of 0.5, and both precision and recall are well above the baseline.

Figures 9.3, 9.4 and table 9.3 display the results of comparing the full text method against the snippet method on the 'salsa' data. With respect to the combined F-measure the results are very similar to those on the 'jaguar' data, but for full text the optimal similarity constant has shifted to 0.8 - 0.85. With the precision and recall, the results are fairly similar for the full text method although the recall is slightly lower across the board. However, the snippet method has performed considerably worse on precision compared to the results achieved on the 'jaguar' data, with the full text method outperforming in both measures at some similarity constants.

There are several conclusions to be drawn.

1. The results found in this project agree with those found by Zamir, that having access to full text data allows for better clusters to be found.

2. This project's new evaluation method is effective at distinguishing between good and

bad clusterings of web pages.

3. Having less data (snippets) can sometimes provide higher precision at the cost of low recall. Having more data often picks up irrelevant documents, lowering precision, but increasing recall significantly.

One drawback to using full text data is that the method is slowed significantly by the requirements of processing the increased amount of data: using the full text method takes thirty seconds to cluster compared with the one or two seconds typically required to cluster using just the snippet data. Given this, while the full text method can produce far more effective results, a significant speed up is required for this method to be of use in the real world. The most expensive operations are the pre-processing and construction of the suffix tree. The feasibility of using full data hence depends on whether the clustering component is integrated with the search allowing these expensive steps to be performed prior to searches. If integrated, the pre-processing and some adapted form of suffix tree building could be pre-performed to allow this to be computed only the once offline, rather than every time a search is conducted. This would allow the time complexity to be reduced significantly and allow the method to be used in real search applications.

A second point of note is that the salsa data set is more difficult to classify. There is significant overlap between a number of the topics contained within the data set, and some form of fuzzy set membership in the evaluation method would be required to alleviate these difficulties. This explains the slightly lower quality results obtained from the 'salsa' data compared to the results obtained from the 'jaguar' data, but the evaluation method has successfully evaluated the methods and the results agree with those found on the more distinctively classifiable 'jaguar' data. This is a testament to the power of the evaluation method.

9.4.3 Maximal Cluster Covering vs Standard vs Original

In this section the effectiveness of the maximal cluster covering stage developed in this project is evaluated. The method is compared against the straight top ten clusters from the standard weighted scoring method developed in this project and also against the top ten clusters from the original un-weighted sum scoring method presented by Zamir [39]. The maximal cluster covering is used with a beta of 1 and a 2 step lookahead. Testing has been conducted using the full text 'jaguar' data set with similarity constants ranging between 0 and 1.

Figures 9.5, 9.6 and table 9.4 display the results of comparing the three methods using the full text 'jaguar' data. The combined F-measure quite clearly highlights the effectiveness of each method, with the maximal cluster covering method winning by a significant margin. The standard scoring method comes in second place and the original scoring method is not far behind the standard method at most similarity constants, although it drops off quite significantly between similarity constants of 0.7 and 0.95.

The precision and recall measures show the maximal cluster covering approach consistently beating the standard approach, but the original scoring method throws up some rather different results. The precision of the original scoring method for similarity between 0.5 and 0.95 is significantly better than any other method and between 0.75 and 0.95, 100% precision is achieved. However this comes at the cost of very low recall, down to 11% in the 0.75 to 0.95 similarity constant range. In these cases the top scoring clusters all related to car

Table 9.4: Jaguar Full Text - Maximal Cluster Covering vs Standard vs Original

Similarity	MCC∗ (%)		Standard (%)		Original (%)	
Constant	P/R	F-measure	P/R	F-measure	P/R	F-measure
0.00	37/45	41	37/45	41	37/45	41
0.05	37/45	41	37/45	41	37/45	41
0.10	37/45	41	37/45	41	37/45	41
0.15	37/45	41	37/45	41	37/45	41
0.20	37/45	41	37/45	41	37/45	41
0.25	37/46	41	37/45	41	37/45	41
0.30	39/47	43	37/45	41	37/45	41
0.35	38/53	44	37/45	41	40/53	45
0.40	43/60	50	45/45	45	41/52	46
0.45	45/60	52	40/44	42	40/52	45
0.50	67/82	74	47/61	53	78/36	49
0.55	59/77	67	49/60	54	78/36	49
0.60	61/65	63	49/60	54	81/31	45
0.65	67/70	68	45/52	48	81/31	45
0.70	65/67	66	48/56	52	87/14	24
0.75	68/67	67	49/55	52	100/11	20
0.80	67/68	68	49/55	52	100/11	20
0.85	67/68	68	49/55	52	100/11	20
0.90	67/68	68	49/55	52	100/11	20
0.95	67/68	68	49/55	52	100/11	20
1.00	67/68	68	49/55	52	49/55	52

(∗ Maximal Cluster Covering) (P/R = Precision/Recall)

Table 9.5: Tuning Similarity Constant and Maximal Cluster Covering Beta

Similarity Constant	Beta 2.00	1.50	1.00	0.75	0.50	0.25
0.40	43/49	43/50	44/60	46/65	45/68	43/69
0.45	43/57	43/62	44/60	45/62	45/61	38/50
0.50	61/60	60/74	61/85	62/85	66/84	67/84
0.55	64/77	67/77	68/74	64/75	63/75	67/76
0.60	72/74	61/70	61/70	64/73	60/59	60/65
0.65	69/74	70/76	68/67	67/70	60/58	64/60
0.70	71/80	70/70	65/78	60/66	62/69	58/58
0.75	72/80	60/56	68/67	62/67	71/68	65/68
0.80	69/80	60/56	69/68	60/57	73/70	65/67
0.85	69/80	60/56	69/68	60/57	73/70	65/67
0.90	69/80	60/56	69/68	60/57	73/70	65/67
0.95	69/80	60/56	69/68	60/57	73/70	65/67
1.00	75/73	58/56	65/58	58/60	72/76	66/63

(Precision/Recall (%))

pages which were easily identifiable and pure clusters were easily found. However, animal, Macintosh OS and Atari game console pages were completely left out from the resulting clusters, as were many of the less easily identifiable car pages. So in this case the 100% precision achieved is not of great importance due to the worse than random performance on the recall. This correlates with the F-measure results and quite clearly the ordering of the three methods is that maximal cluster covering is the clear winner, the standard method is second and the original method third.

The clear advantage of the maximal cluster covering method over the other methods and the new scoring method over the original scoring method shows that these contributions were worthwhile and advance the state of the art in web clustering. Whether it is feasible to perform maximal cluster covering will be considered later as part of the investigation into the amount of lookahead required.

9.4.4 Similarity Constant vs Maximal Cluster Covering Beta

This section seeks to determine the best settings for the similarity constant and the maximal cluster covering beta. The relationship between the two and their impact on precision and recall is also considered. Results are obtained by using suffix tree clustering with maximal cluster covering (STCE) on the full text 'jaguar' result set with 2 steps of maximal cluster covering lookahead.

Figure 9.7 shows a three dimensional surface plot of the F-measure combination of the precision and recall shown in table 9.5. It is noted that due to the way Excel generated the graph, there is a run-off to 0 at one extreme, this does not happen in the data and should be ignored.

From the graph and the data it can be seen that with similarity constant of between 0.5 and 0.6 that the results are fairly independent of the beta values between 0.25 and 2. But as the

Table 9.6: Lookahead Level Results

Steps of Lookahead	Precision / Recall (%)
0	54.5 / 68.4
1	65.9 / 81.9
2	61.3 / 84.8
3∗	63.7 / 83.0

(∗ slowed maximal covering stage from under a second to over an hour)

similarity constant increases above 0.6, betas at the extremes of 0.25 and 2 are better than at the points in between. While the F-measures at each of these points are fairly similar, the trade off between the precision and recall at these points is not always the same, some points such as those with larger similarity constants can achieve better precision at the cost of a little recall, while those closer to the 0.5 - 0.5 range of similarity constant produce slightly higher recall at the cost of precision.

The following effect of similarity constant can be observed: as similarity decreases, precision decreases. This is because as similarity decreases more clusters are joined and more mistakes are made and unrelated clusters joined, leading to decreased precision. For similar reasons, as similarity decreases, we expect recall to increase. This does occur as the similarity constant starts decreasing from 1, but at some point recall starts to decrease. This is unexpected, but occurs due to the way recall is determined. Since a cluster represents just a single category, if two clusters are joined, then a poorer quality cluster will be selected to represent the other class's documents or possibly none if all of the documents from that class have ended up in the single large cluster. The recall on the other class is thus decreased significantly and the overall recall too will drop.

Choosing an optimal beta and similarity constant is no easy task and depends on the requirements and the particular problem distribution. Due to the distribution it would appear that selecting the similarity constant independently of the beta would be wise, as the actual performance depends on the data set as much as anything else. Across all betas and in the 'salsa' data set as well, a similarity constant of 0.5 through 0.7 provides consistently good results, whereas higher similarity constants can suffer or behave sporadically on different data sets. Under some conditions low beta can provide good results, but higher beta values produce better results more frequently. Hence choosing a beta between one and two is preferable. Higher beta values in fact reduce the amount of overlap allowed, so it would seem that by controlling the amount of overlap, more consistent results can be produced.

9.4.5 Maximal Cluster Covering Lookahead

This section compares the performance and cost of using different amounts of lookahead in the maximal cluster covering stage. Results are obtained by using suffix tree clustering with maximal cluster covering (STCE) on the 'jaguar' full text result set with a similarity constant of 0.5, maximal cluster covering beta of one and different amounts of lookahead.

As shown in figure 9.8 and table 9.6, some lookahead is clearly worthwhile with a large improvement in both precision and recall with just one step of lookahead. But beyond that there is not that much benefit: going to two steps is a possibility, but beyond the first step the method is essentially trading off precision for recall, something that is not necessarily a good idea. Two step lookahead also comes at a slight additional computational cost, though with the techniques implemented, two step lookahead is still very efficient and there is not a significant difference between one and two step lookahead in terms of running time. However, three steps is just far too slow and takes the overall run time from under thirty seconds to over an hour, since there is just a slight shift in the precision and recall and no real improvement, there is no point in using three step lookahead.

Hence, a single step is all that is required and two is possible, but offers no real benefit. The results found from investigating the run time of the maximal cluster covering stage found that there is not a significant burden placed on the run time of the algorithm by implementing this additional stage if just one or two steps of lookahead is used. Furthermore, the added run time is insignificant when compared to the computational cost of conducting the pre-processing and building the suffix tree in the case of the full text data.

9.4.6 Irrelevance Identification

From looking at the results typical of the suffix tree clustering method with the maximal covering extension, such as those shown in Appendix B, it was found that the single biggest cause of loss of precision was documents irrelevant to the search being picked up and then clustered due to freak occurrences of matching words or terminology. With such documents generating around 25% of the loss of precision, a method to identify and either eliminate or shift these documents into an 'other' category could significantly increase the performance of the clustering methods in terms of precision.

As previously discussed, a variety of techniques were considered and two based on base cluster analysis were implemented, one based solely on base cluster frequency, the second based on the base cluster frequency, but weighted by the score and size of the base clusters.

Figures 9.9 and 9.10 compare the number of documents removed against the number of irrelevant documents removed for the two techniques. In this situation 'removed' can also be interpreted to mean 'identified'. The graphs compare the values at a number of different thresholds and any location on these curves can be achieved by altering that threshold.

As can be seen from carefully examining figures 9.9 and 9.10, both methods do reasonably well at identifying the irrelevant documents to start with, but before too long, too many real documents are identified. Ideally the graphs should go straight up the y axis, then along the x axis. This would allow for perfect identification of the irrelevant documents. The only problem is the two methods considered provide a fairly diagonal line, which shows that these methods are not particularly good at identifying irrelevant documents. Once the fraction of irrelevant documents removed versus real documents removed drops below 25% there is no benefit to either precision or recall, and as the number of real documents drops from the beginning, the recall is similarly dropping. While this fraction does not drop below the 25% mark until quite extreme values, there is no point in going this far, as the drop in recall will be very severe compared to the marginal small increase in precision. Smaller cutoffs where the fraction is closer to 50% such as those near the left of the graph provide better results and in this region of the graph the straight base cluster frequency method works best. The only problem with this is that with the small gains that are experienced in precision a fairly equal trade off is seen in recall and the gains are very small.

From this it can be seen that while the methods investigated here indicate there is potential for identifying irrelevant documents, the actual use of the methods implemented here is not great. Apart from allowing a slightly increased ability to trade off precision and recall there is no real advantage to using these methods. It is thus concluded that the methods implemented are not effective at this task, but perhaps one of the other methods discussed could achieve this task and provide the accompanying benefits to precision while not sacrificing much recall.

9.4.7 The Precision-Recall Approach

A final test that was performed late in the development and thus not discussed earlier or in detail, was to investigate whether removing words that would be scored as 0 before building the suffix tree would have a significant effect on the clustering results. The words that are scored as 0 are the words that occur in fewer than some percentage or the documents or that appear in more than some percentage of the documents. It was found that the entire clustering process was sped up substantially, while the resulting precision increased slightly and recall decreased slightly. On the 'jaguar' data set for instance, in the majority of the cases within the suitable ranges of similarity constant and beta identified earlier around a 5% increase in precision and a 6 - 8% drop in recall were identified. Combined with the speed increase, these results appear very reasonable.

A further observation is that these figures depend on the percentages used for pruning of non-scorable words. With tighter bounds on what words are acceptable, there is generally a greater increase in precision and a greater drop in recall. An observation from this is that changing the amount of data available for clustering allows precision and recall to be traded off against one another. Of course, no more recall can be achieved than there is initially and there is also a limit on how much precision can be achieved. However this is identified as providing a better method for comparing and evaluating clustering methods and is termed the precision-recall approach. By evaluating clustering methods with a variety of different percentages, precision-recall curves would shown the traditional trade off between precision and recall. This would allow methods to be compared based on the amount of data provided. For instance, one algorithm may have a very steep curve and with full information it may be better than any other algorithm, but due to the very steep curve, it is not nearly as good with less information. On the other hand, another method with an almost flat curve may be slightly worse with full data, but with substantially less data it may be almost as well and do so in much reduced time. This evaluation method is certainly a good extension to the evaluation method suggested in this report and the theories developed about it here should be properly analysed to see if they hold true in general.

9.5 A Comparison with Grokker

Introduced in chapter 3 and shown in figure 3.7, Grokker appears to be a fairly effective clustering search engine which appears to produce rather good results. In this section Grokker is evaluated and compared against the suffix tree clustering extended system that includes maximal cluster covering.

Table 9.7: Grokker Result

Correct / Total	Category	Cluster Category Break Down
12 / 15	animal	12 animal, 1 car, 1 c++, 1 unknown
18 / 22	animal	18 animal, 2 car, 1 clothes, 1 poem
20 / 21	car	1 animal, 20 car
15 / 21	car	1 animal, 15 car, 1 food, 2 macintosh, 1 movie, 1 star trek
4 / 13	car	1 bats, 4 car, 2 game, ... 1 search, 1 unknown
7 / 9	car	7 car, 2 game
14 / 21	car	14 car, 2 game, 3 macintosh, 1 music, 1 power supply
19 / 20	car	19 car, 1 star trek
15 / 18	games	15 games, 1 macintosh, 1 music, 1 web design

9.5.1 Evaluating Grokker

To evaluate Grokker a search for 'jaguar' was performed with Grokker set up to obtain its results solely from Google. The top level 'other' category and other generic branches were removed as these are considered to be documents that were not clustered and are ones that Grokker could not place in with the main clusters. The remaining clusters were of a hierarchical nature; these hierarchies were flattened into their top level clusters. Duplicates within clusters were also removed. In total, Grokker clustered 208 items; after the deletions there were 9 clusters remaining, comparable to the 10 generated using this projects method. The number returned by Grokker varies and 8 and 10 were also seen in other test runs.

Of the 9 clusters, 2 related to animals, 1 to the Atari games console, 0 to Macintosh OS (these were all left in the other cluster for some reason and there was a reasonably large Macintosh OS cluster in there) and the remaining 6 related to cars. To calculate precision we took the sum of the number of documents correct in each cluster and divided by the sum of the number of documents in each cluster. To calculate recall, we merged all clusters with the same category, then took the sum of the number of distinct correct documents divided by the sum of the number of documents that could possibly have been in each category.

In total there were 171 potential documents which could have been clustered and 88 distinct correct documents remain in the merged clusters. Along with the sums of the results from table 9.7 this gives a precision of $\frac{124}{160} = 77.5\%$ and a recall of $\frac{88}{171} = 51.5\%$.

It is noted that in some situations Grokker picks different clusters to form the top level clusters. The algorithm was run several times and in some cases the Macintosh category was represented by a top level cluster, instead of as a sub-cluster in the other category. When a run with a top level Macintosh cluster was analysed, the precision was very similar at 76.0% and recall was marginally improved to 56.0%. It seems that in bringing in the Macintosh cluster, other clusters were degraded and so the recall did not improve significantly.

9.5.2 Comparing Grokker to STCE

The results of Grokker are very interesting as they follow closely the results of the snippets based STCE method used in this project, which achieved 80% precision and 46% recall compared to Grokker's 76% precision and 56% recall. From inspection of the data available to Grokker, it has access to the page titles in addition to the snippets that the snippet approach in this project has. This additional data can explain the small difference between the precision and recall, as lower precision and higher recall are associated with increased data and that is what is observed here. From the investigation and the examination of the clusters produced by Grokker, it would appear that their implementation uses a method rather similar to suffix tree clustering. Since Grokker is a commercial application, this type of information is not publicly available and so this cannot be confirmed. One claim about their clustering method from [11] is that the method uses linguistic techniques to improve the quality of the results. Since similar results can be obtained using the same data without such techniques, it would appear that either the techniques implemented are ineffective or that the techniques were not well targeted towards the 'jaguar' search. If the linguistic techniques were hand crafted that may explain why. Another observation is that their process was significantly slower than the snippet based suffix tree method of this project — for result sets of the same size, their method took approximately 15 seconds, compared to less than a second with this projects implementation. This is probably explained however by the overheads of the graphical interface and of the hierarchical application of clustering in the Grokker application.

What is most surprising about the results is that they look very good. From a pure user observation of their resulting clusters it appears to be of high quality and very usable; it is only when it comes to evaluation that the flaws are detected. This bodes well for the methods developed in this project which can produce results of similar quality in the case of snippets and of better quality with full text information. It also shows that particularly high recall and precision are not the be-all and end-all of effective clustering and that one of the biggest limitations is really the number of documents that can be clustered efficiently. If clustering of equivalent quality could be conducted on a large scale efficiently, then the methods would be suitable for real world use. Of course, even on the large scale, the benefits of further increasing precision and recall measures would not go unnoticed and further research into these areas is certainly warranted.

9.6 Summary

This chapter has shown that the evaluation method developed in this project can effectively evaluate web cluster results. It was also demonstrated that it can reach the same conclusions as other evaluation methods used in the past, such as those used by Zamir [38] to evaluate whether full text or snippets are better for clustering. As it turns out the obvious answer that more information and thus full text is better, is again proved true by this projects new evaluation method.

The maximal cluster covering extension is also evaluated and it is found that this stage is both very effective at improving the results over previous methods and that only one step of lookahead is required to achieve this improvement. This showed that the maximal cluster covering extension is worthwhile and is a great contribution to the web search clustering problem.

The irrelevance identification methods were evaluated and it was found that the methods developed in this project do not work well at this task.

The precision-recall approach was identified as an improved method for evaluating and comparing different web clustering algorithms.

Finally Grokker, a commercial web search clustering application, was evaluated and it was found that the results of the snippet based STCE algorithm from this project with 80% precision and 46% recall is comparable to Grokker's 76% precision and 56% recall. The full text based STCE algorithm from this project is substantially better with 72% precision and 80% recall.

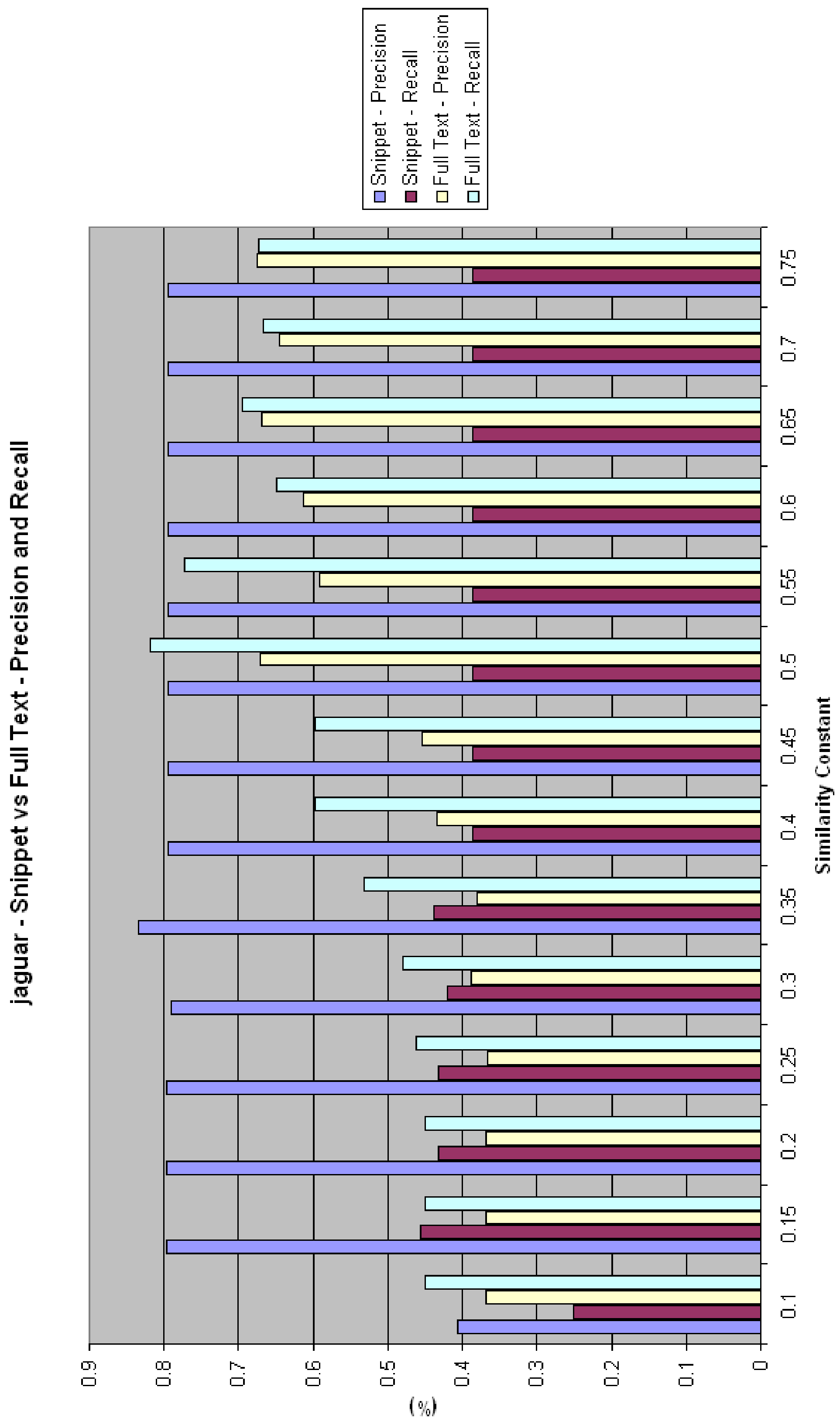


Figure 9.2: Jaguar - Snippet vs Full Text - Precision and Recall

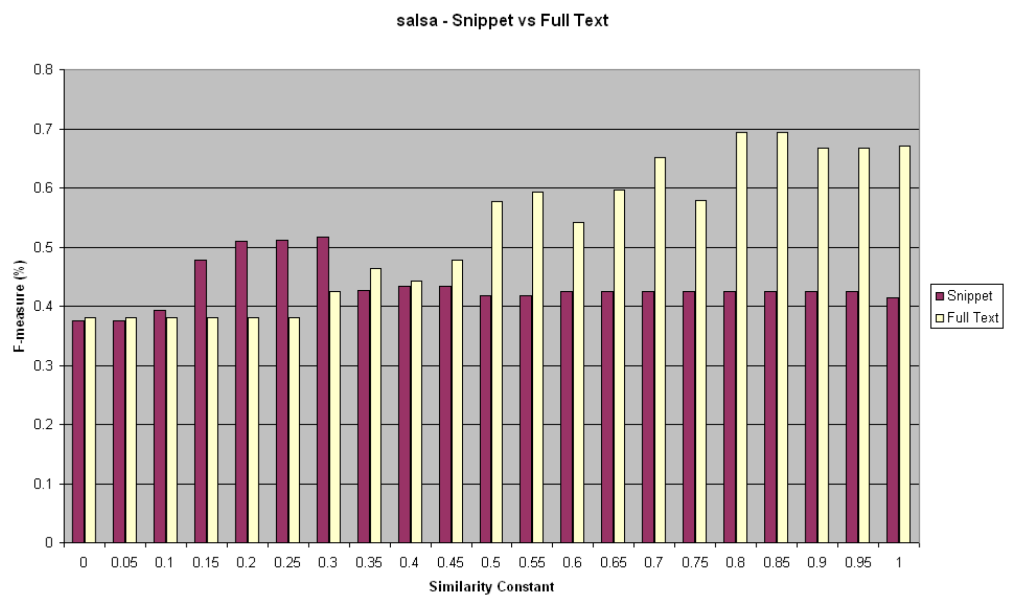


Figure 9.3: Salsa - Snippet vs Full Text - F-measure

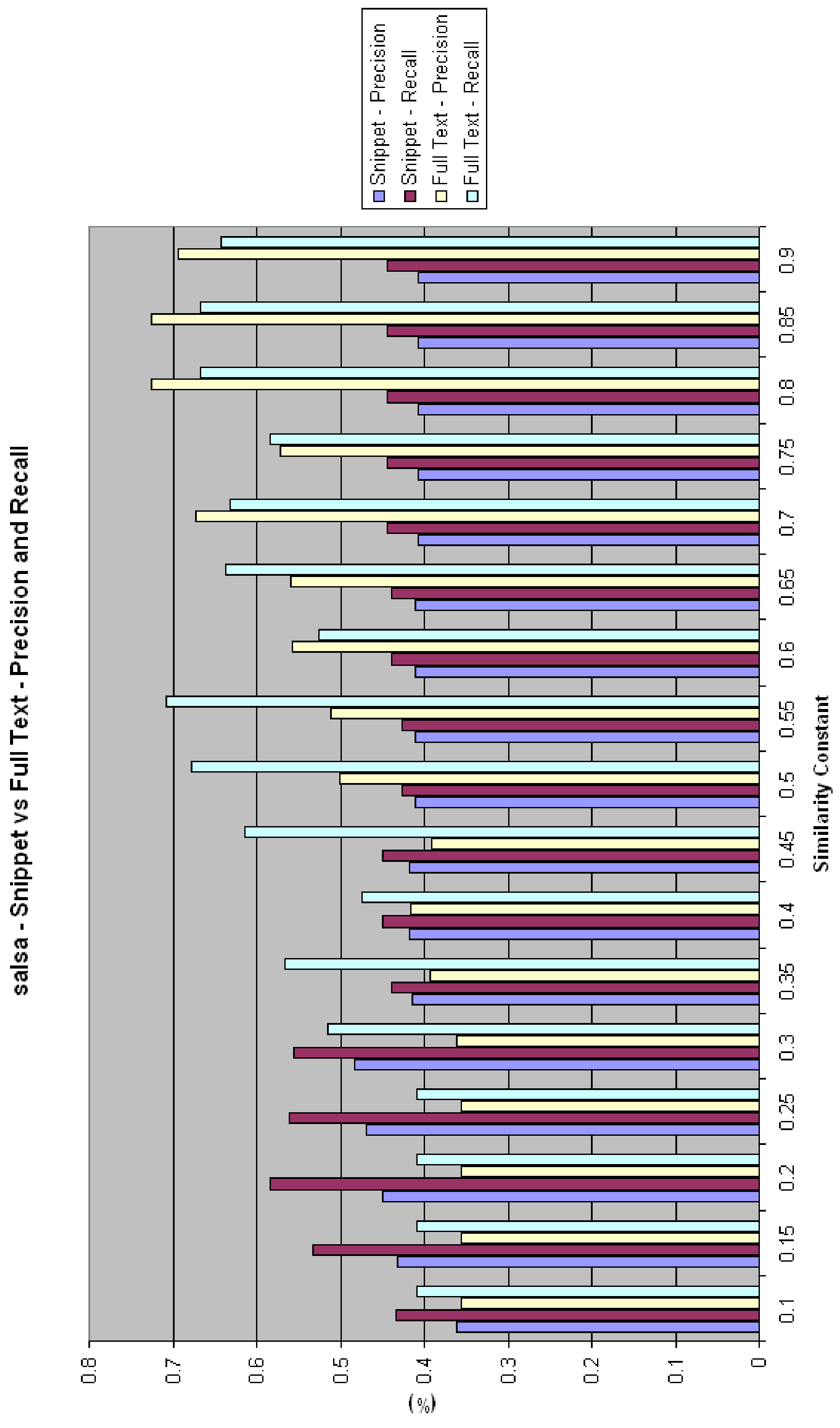


Figure 9.4: Salsa - Snippet vs Full Text - Precision and Recall

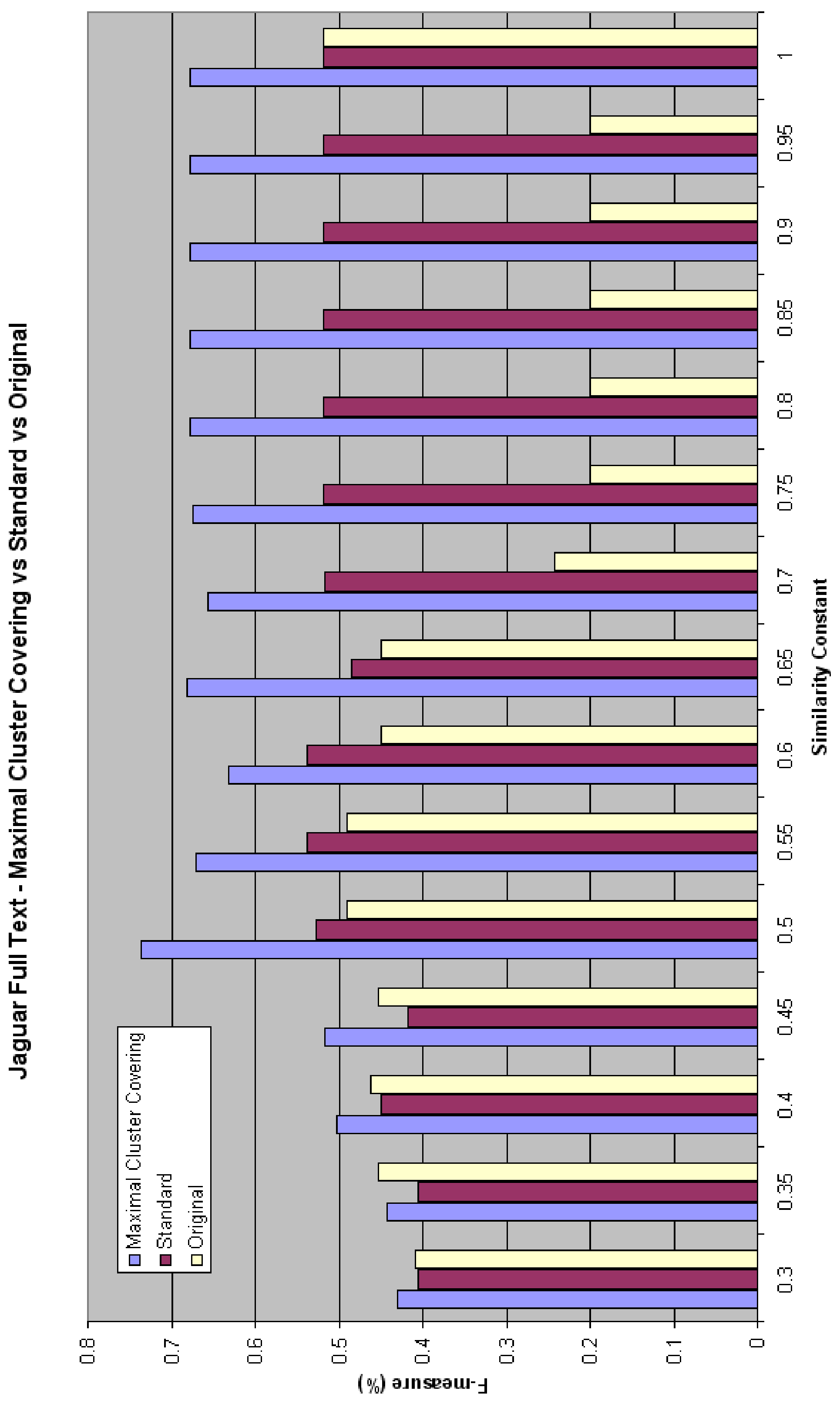


Figure 9.5: Jaguar Full Text - Maximal Cluster Covering vs Standard vs Original - F-measure

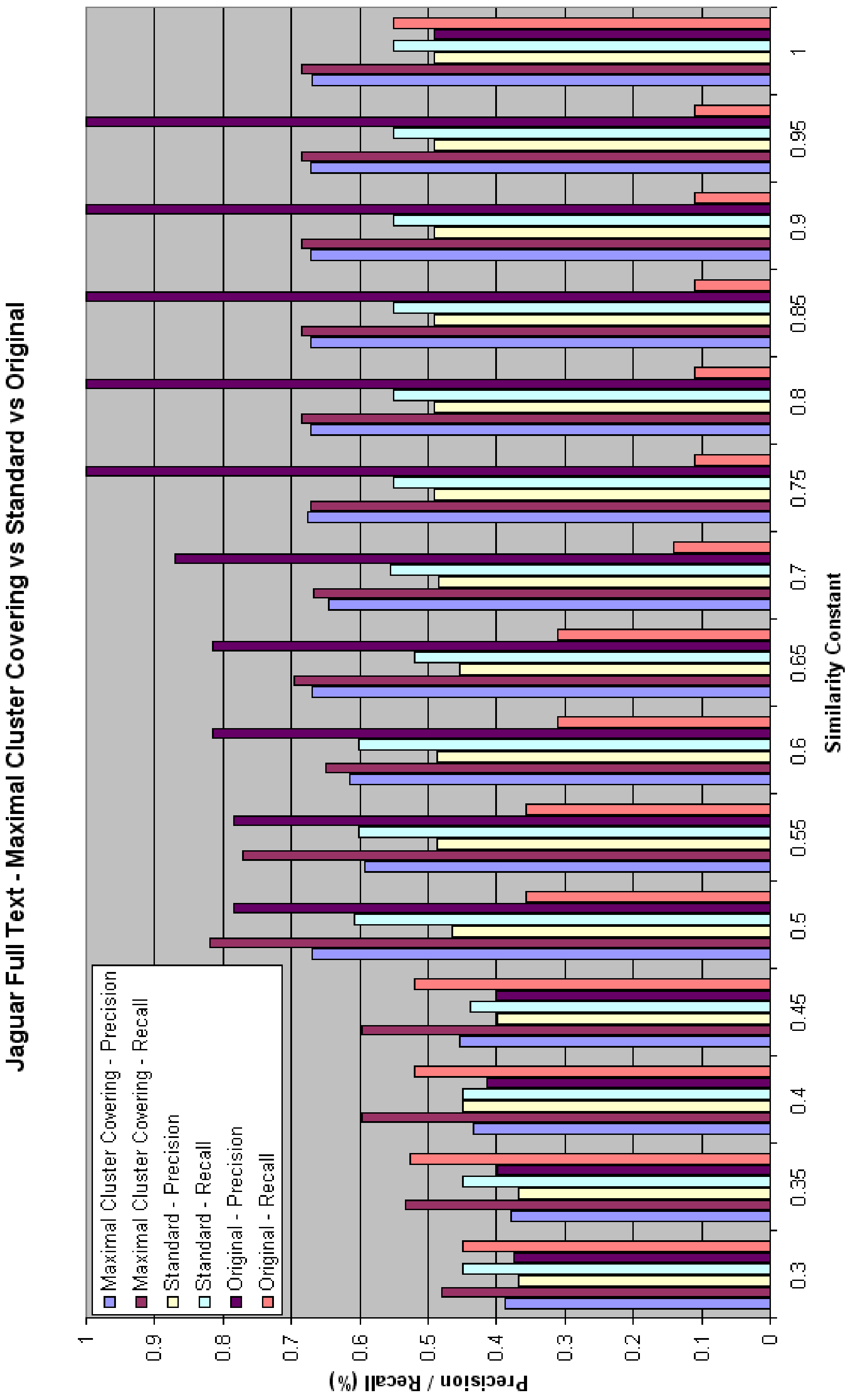


Figure 9.6: Jaguar - Maximal Cluster Covering vs Standard vs Original - Precision and Recall

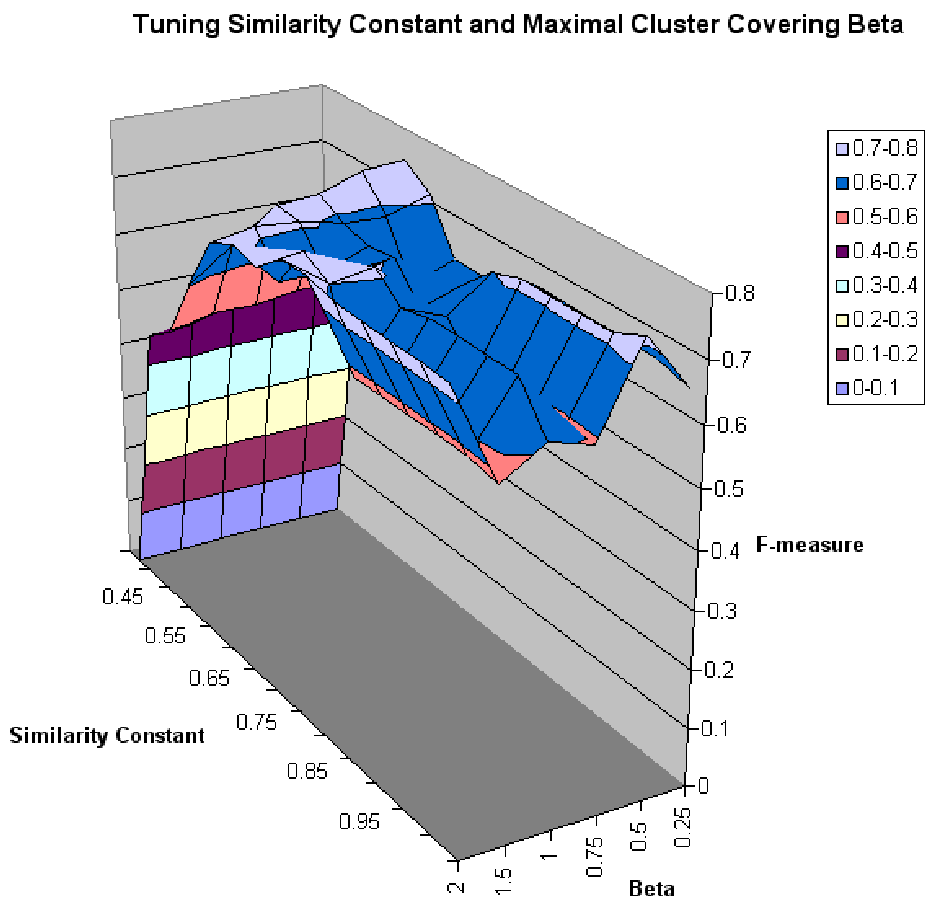


Figure 9.7: Tuning Similarity Constant and Maximal Cluster Covering Beta

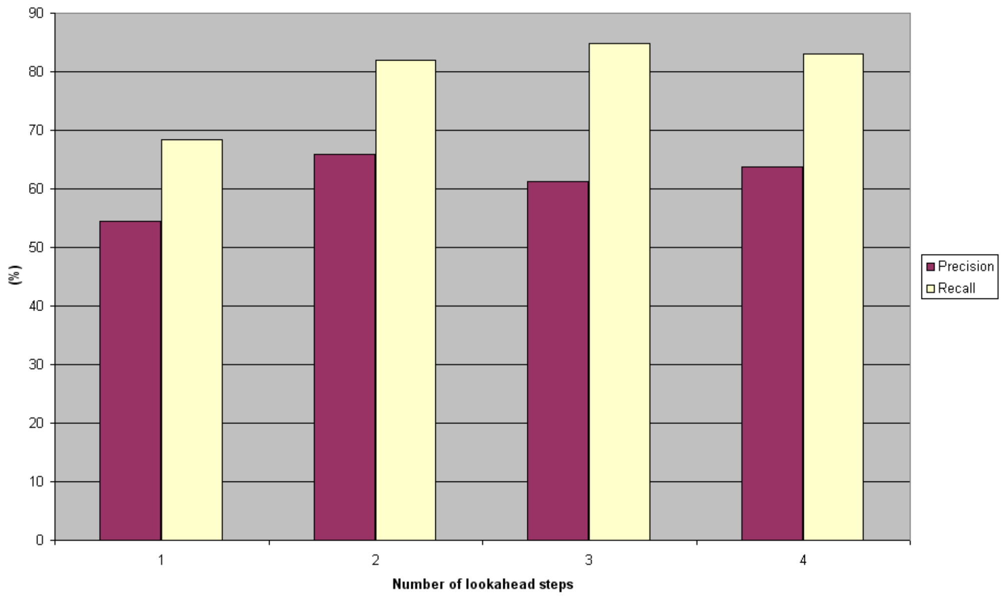


Figure 9.8: Comparison of different numbers of lookahead steps

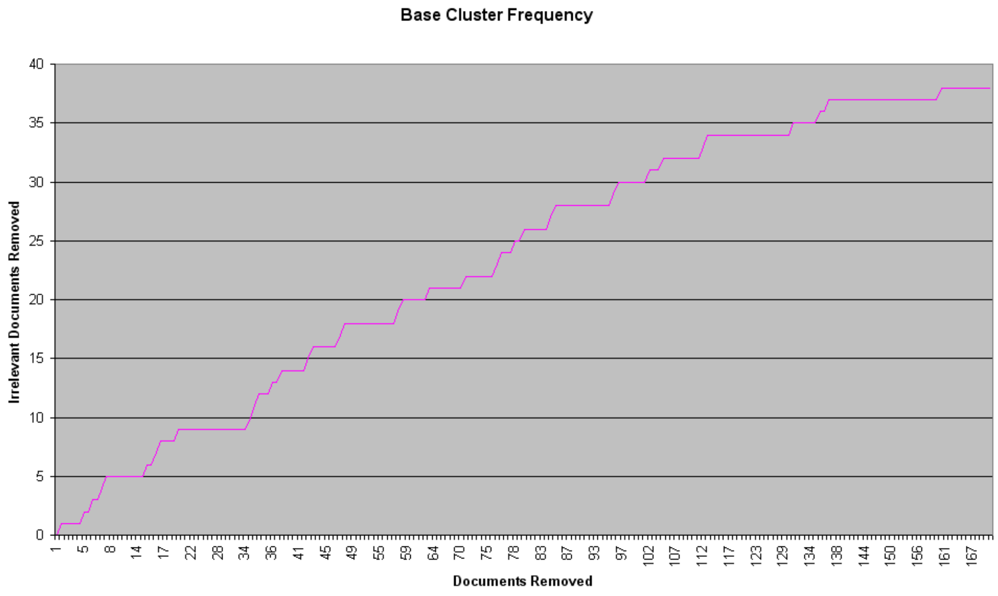


Figure 9.9: Irrelevance Identification based on Base Cluster Frequency

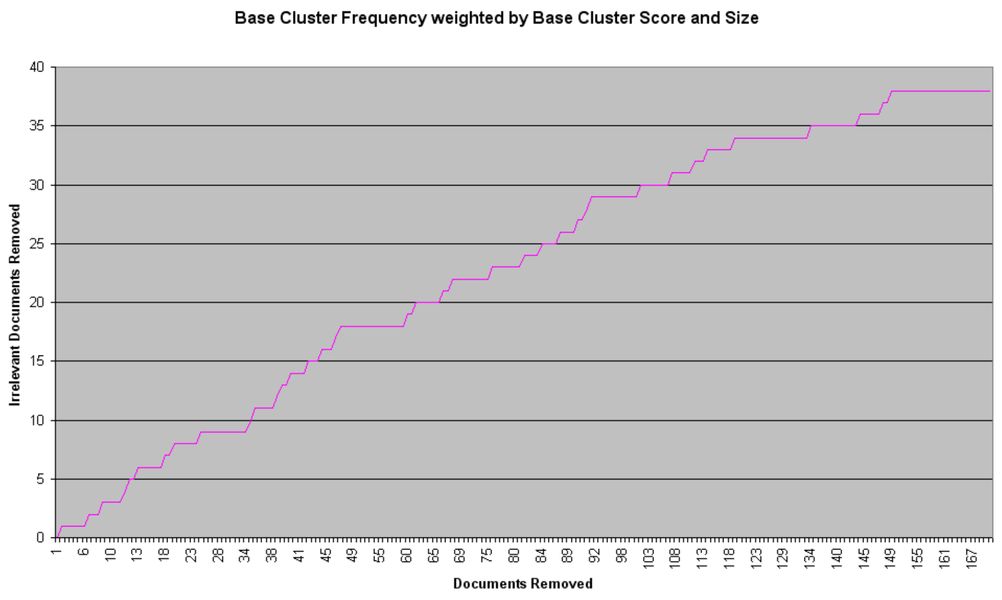


Figure 9.10: Irrelevance Identification based on Weighted Base Cluster Frequency

Chapter 10

Future Directions

10.1 Introduction

There are many different directions for future work on web search clustering covering all four phases of the clustering problem, preprocessing (phases 1 and 2), clustering (phase 3) and evaluation (phase 4). Throughout this report, suggestions for future work, ideas, descriptions and algorithms have been presented which could form the basis of future work in the web search clustering area. This chapter summarises these and suggests a few more directions for future work.

10.2 Preprocessing

While it seems like there is not going to be anything ground breaking within the preprocessing area, the preprocessing area is an area where most of the processing time is spent in the current post search clustering applications. This leads to several different research projects. One is to build a search engine with clustering built in. This would require an efficient way of accessing and storing preprocessed information so that post search clustering is far more efficient. Another approach would be to research the generation of effective page summaries: being able to generate an effective page summary of a full web page that can provide full text cluster quality with snippet sized amounts of data would allow for more efficient suffix tree building, one of the slowest stages of the clustering process at this time. Generating effective page summaries would also be useful in developing an integrated search and clustering engine and could also be useful for aiding in describing clusters.

There are also significant areas of page cleaning that could be investigated. The phrasifying and splitting process in particular is quite important and variations on this could be considered, contrasted and compared. Intelligent dynamic cleaning and splitting agents could also be developed to dynamically split and clean pages depending on the content, layout or other parameters. The impact of the stop word list and dynamic word removal could be investigated. The effects of stemming and how these affect the clustering process is another potential area of investigation.

Quite a different approach than anything attempted in this project is to use linguistic, natural language and semantic information to provide additional input to the clustering process.

Consider documents which frequently contain terms relating to cars. These terms could be transformed into higher level terms, such as 'car' becoming 'vehicle' or transformed into equivalent terms for instance, 'automobile', 'auto' and 'car' all becoming 'car'. Other language based techniques could also be applied and this is a large area of research that could be applied to web page clustering.

Any real application that was to build on the work in this project should also handle aspects not handled in this project, that are handled in other clustering applications. These aspects were not implemented in this project due to time constraints. Other extensions of this project could include combining snippets from multiple search engines, results from multiple engines, better page handling such as handling frames, redirections, foreign language pages.

10.3 Clustering

There are three main areas of investigation into clustering techniques identified in this report: new clustering algorithms, large scale parallel clustering algorithms to handle large data sets and, cluster descriptions / naming.

On the clustering algorithm front, using suffix trees or the knowledge that 2-grams or 3-grams can provide equivalent information as the basis for new clustering algorithms seems like a good idea and the textual information that can be obtained from the pages alone combined in new ways can provide a multitude of new and improved algorithms. Earlier in this report, several possible variations on the single link clustering method were described, as were several radically different methods based on K-means and agglomerative approaches. Implementing and testing these new algorithms would provide a great direction for future research. Another direction research could take is to use other information such as more textual information found from the hyper-link structure of the internet or from natural language or linguistic methods, as mentioned in the previous section. Image data and page or site layout are other areas that could be researched, with some of the research in these areas indicating that there are benefits to combining these kinds of information with more traditional textual information; these areas are well worth investigating further.

Large-scale clustering is a problem that has not been dealt with in any great depth. With the preprocessed data stored across multiple computers (typical of any large search engine), a parallel algorithm with the suffix tree nodes split across multiple computers could be beneficial. However, the documents and clusters would still need to be combined on a single machine at some point and doing that on large data sets is time consuming. Quicker and significantly more efficient parallel clustering algorithms will be required. Based on the findings of this project, sampling may be a better direction. If links between documents that are predetermined to be similar were stored, then a nearest centroid based clustering method could allow efficient en-mass clustering.

Finally, a problem that was not addressed at all in this project is the problem of naming or describing the clusters to the user. Depending on the interface, descriptions may have to be as short as a few words and the words used in generated the cluster are typically so wide in scope that selecting just a few of the most important and descriptive words is no easy task. The first run approach to this problem would approach it in ways similar to what other researchers have done up until this point and name clusters by using heuristics and statistical methods. However, a linguistic approach may be required to generate optimally descriptive names.

10.4 Evaluation

The new evaluation method has been described, analysed and tested. But there are still two flaws in the method. One is the problem of evaluating large data set clustering techniques and clustering performance on large data sets. The sampling method seems like the most applicable approach to this and this should be investigated and the number of samples required needs to be determined. The other problem is the issue with the bad solution of permutations of documents within sub-clusters. Several different approaches, including fuzzy set membership and hierarchical topic structure definitions, provide potential solutions and should be investigated.

One of the last tests performed discovered that altering the percentages for non-scorable words and removing them before building the suffix tree can provide a better method for comparing two clustering methods. This approach should be investigated and whether this holds true of other non suffix tree clustering methods should also be investigated. If this holds true, then this would provide an excellent extension of the web clustering evaluation method described in this report.

Finally for the evaluation method described here to really form a standard, many different clustering algorithms should be implemented and evaluated using this evaluation method. Standard result sets of different complexities and sizes should also be developed to provide a benchmark and standard for comparison of all new research on web search clustering. Providing this benchmark is key to bringing together the research conducting by different people in this area which is currently quite uncoordinated due to the lack of a comprehensive and universal evaluation method.

Chapter 11

Conclusions

This report has covered a wide variety of material on almost all aspects of the web clustering process. It has presented the problem of search on the internet and described a variety of existing search tools and clustering techniques, particularly the suffix tree clustering algorithm. It also presented several approaches for modifying the suffix tree algorithm by integrating various other clustering algorithms.

The report discussed different approaches to evaluating clustering techniques and presented a new method based on sound evaluation criteria. It then described the system implemented for this project and presented the results of experimental evaluation of this system and a comparison with other systems.

In this project there were five main contributions:

- the maximal cluster covering extension to suffix tree clustering
- the new evaluation method and evaluation criteria
- outlines of several new approaches for combining suffix tree clustering with other clustering algorithms
- an investigation into irrelevant document identification
- the implementation and evaluation of the suffix tree clustering algorithm

Maximal cluster covering extended the suffix tree clustering algorithm, although it could just as easily extend any other clustering algorithm where final cluster selection from a large number of clusters is required. Maximal cluster covering significantly improved both the precision and recall of the results and computing the maximal cluster covering added little overhead to the algorithm with only a one-step look-ahead needed to obtain high quality results.

The new evaluation method provides an effective measure of cluster quality based soundly on a set of criteria that allow evaluation and comparison of different techniques. Combined with the precision-recall approach described in chapter 9, the evaluation method can directly compare different techniques based on the amount of data provided. The evaluation method agreed with results of past research and provided an effective measure of differences between clustering techniques. The report proposed the evaluation method as a standard benchmark for future web clustering research, and suggestions on how to achieve this.

The problems of single link suffix tree clustering such as chaining were presented, some implementation considerations were described, and several new approaches for combining suffix trees with other clustering algorithms to fix the problems of the single link method were outlined.

Irrelevant documents were identified as a major cause of loss of precision in suffix tree clustering results. Several methods were proposed to identify irrelevant documents. Two were implemented and ruled out of contention by being shown to be ineffective.

The suffix tree clustering algorithm was implemented and evaluated by reproducing results of past research. The results confirmed that full text data produces better results than snippet only data. Of particular note is that all the clustering algorithms were implemented based on the very limited description in [39]. The results and methods as implemented based on the limited description were thus developed independently of the work described in the thesis [38] which was only found late in the project. Upon comparison, the methods independently developed in this project follow very closely those of Zamir, and the suffix tree building implementation in particular was as time efficient, although not as space efficient as the methods described and referenced from [38].

The report also laid out a variety of directions for future work, building on the contributions of this project.

Appendix A

Stop Words

a
about
above
according
across
actually
adj
after
afterwards
again
against
all
almost
alone
along
already
also
although
always
among
amongst
amp
an
and
another
any
anyhow
anyone
anything
anywhere
are
aren
around
as
at
b
back
be
became
because
become
becomes
becoming
been
before
beforehand
begin
beginning
behind
being
below
beside
besides
between
beyond
billion
both
but
by
c
can
cannot
caption
click
com
copy
copyright
could
couldn
d
did
didn
do
does
doesn
don
dont
down
during
e
each
eight
eighty
either
else
elsewhere
end
ending
enough
etc
even
ever
every
everyone
everything
everywhere
except
f
few
fifty
find
first
five
for
former
formerly
forty
found
four
from
further
g
get
h
had
has
hasn
have
haven
hence
her
here
hereafter
hereby
herein
hereupon
hers
herself
him
himself
his
home
homepage
how
however
http

hundred
i
if
in
inc
indeed
instead
into
is
isn
it
its
itself
j
just
k
l
last
later
latter
latterly
least
less
let
like
likely
ltd
m
made
make
makes
many
may
maybe
meantime
meanwhile
might
million
miss
more
moreover
most
mostly
mrs
much
must
myself
n
namely
nbsp
neither
never
nevertheless
new
next
nine
ninety
no
nobody
none
nonetheless
noone
nor
not
nothing
now
nowhere
o
of
off
often
on
once
one
only
onto
or
other
others
otherwise
our
ours
ourselves
out
over
overall
own
p
page
per
perhaps
please
q
quot
r
rather
recent
recently
s
same
search
see
seem
seemed
seeming
seems
seven
so
seventy
several
she
should
shouldn
since
site
sites
six
sixty
some
somehow
someone
something
sometime
sometimes
somewhere
still
stop
such
t
taking
ten
than
that
the
their
them
themselves
then
thence
there
thereafter
thereby
therefore
therein
thereupon
these
they
thirty
this
those
though
thousand
three
through
throughout
thru
thus
together
to
too
toward
towards
trillion
twenty
two
under
unless
unlike
u
unlikely
until
up
update
updated
upon
us
used
using
v
very
via
w
was
wasn
we
web
well
were
weren
what
whatever
when
whence
whenever
where
whereafter
whereas
whereby
wherein
whereupon

wherever	whom	won	you
whether	whomever	would	your
which	whose	wouldn	yours
while	why	www	yourself
whither	will	x	yourselves
who	with	y	z
whoever	within	yes	
whole	without	yet	

Appendix B

Jaguar Result Set

```
Test: STC Full Text - Maximal Cover
Search: 'jaguar', maximum 300 results

————————————————————————————-
Phrase: car
car
Score: 36.5
Num Docs: 73
Summary: Technology=1, Physics=1, Xfiles=1, Animal=3, Games=1, Car=65, Story=1
————————————————————————————-
Phrase: softwar
comput
Score: 34.5
Num Docs: 58
Summary: Power Supply=1, Physics=1, Links=1, Games=16, Network=1, Aircraft=2, Mac-
intosh=31, Web Design=1, Car=1, Chemistry=2, Story=1
————————————————————————————-
Phrase: base
wwf provid aid protect remain rain forest area south america provid refug major remain
jaguar popul
monkei lower branch larg rain forest tree rang man live close jaguar compet human hunter
poacher take speci big cat peak declin sixti seventi 000 jaguar kill year sought coat
leopard cat
presenc small dot irregular shape larger rosett mark stocki muscular bodi shorter tail
larg head stocki forelimb good wai differenti cat wild
member panthera famili america far biggest cat contin jaguar rang span southern state usa
tip south america centr north central part south american contin jaguar predominantli forest
dweller highest popul densiti centr lowland rain forest amazon
... [8 pages removed] ...
wwf provid aid protect remain rain forest area south america provid refug major remain
jaguar popul
refug major remain jaguar popul
Score: 25.5
Num Docs: 42
Summary: Bat=1, Physics=1, Maya=1, Animal=29, Unknown=1, Star Trek=1, Games=1,
Car=4, Pizza=1, Teeshirt=1, Story=1
——————————————————-
Phrase: sale
Score: 15.0
Num Docs: 25
Summary: Games=4, Macintosh=3, Web Design=1, Car=17
——————————————————-
Phrase: 32 bit system maximum regist size programm processor 68000 graphic processor
dma sound processor sai jaguar consid 64 bit system 64 bit compon gpu access 64 bit data
requir lack
64 bit system 64 bit compon gpu access 64 bit data requir lack
architectur jez san argonaut softwar sai jaguar consid 32 bit system maximum regist size
programm processor 68000 graphic processor dma sound processor sai jaguar consid 64 bit
system 64 bit compon gpu access 64 bit data requir lack
ask question
realli 64 bit system question hard resolv
... [11 pages removed] ...
develop
monei run
system
Score: 26.0
Num Docs: 20
Summary: Games=16, Network=1, Macintosh=2, Car=1
——————————————————-
Phrase: enthusiast express permiss violat prohibit
jag technic help xj technic help look brochur jdht certif help
cat devic separ combin regist trademark properti jaguar car
devic separ combin regist trademark properti jaguar car
1922 todai
enthusiast express permiss violat prohibit
jag lover manner impli endors commerci activ whatsoev prohibit word jaguar leap cat devic
separ combin regist trademark properti jaguar car
... [1 page removed] ...
therefrom origin alter form page includ jaguar enthusiast express permiss violat prohibit
jaguar car
Score: 17.0
Num Docs: 5
Summary: Car=5
——————————————————-
Phrase: onlin regist free click topic topic java emul badboi begameboi bepcengin calic koyot
oswan virtual jaguar
style style
doubl side style style
jaguar video iwar 4
host emu franc thank smix help
member onlin regist free click topic topic java emul badboi begameboi bepcengin calic koyot
oswan virtual jaguar
video iwar 4
... [1/2 page removed] ...
emulaccin import url theme perman doubl side style style
```

```
topic java emul badboi begameboi bepcengin calic koyot oswan virtual jaguar
java emul badboi begameboi bepcengin calic koyot oswan virtual jaguar
Score: 15.0
Num Docs: 3
Summary: Games=2, Car=1
--------------------------------------------
Phrase: book
client
rendezv
sherlock
extrem
itun
firewal
imovi
carbon
Score: 12.5
Num Docs: 20
Summary: Web Hosting=1, Macintosh=18, Database=1
--------------------------------------------
Phrase: chang
Score: 26.5
Num Docs: 27
Summary: Games=3, Macintosh=20, Aircraft=1, Car=1, Model=1, Chemistry=1
--------------------------------------------
Phrase: protect
Score: 12.0
Num Docs: 17
Summary: Animal=13, Car=3, Hotel=1
--------------------------------------------


--------------------------------------------
Unique Combination of All Clusters Returned, shows total coverage, compare to 210 documents of original data set.
Score: 0.0
Num Docs: 181
Summary: Unknown=1, Xfiles=1, Games=24, Physics=2, Hotel=1, Bat=1, Links=1, Maya=1, Technology=1, Pizza=1, Car=71, Model=1, Star Trek=1, Web Design=2, Aircraft=2, Story=2, Web Hosting=1, Teeshirt=1, Macintosh=32, Chemistry=2, Network=1, Database=1, Animal=29, Power Supply=1
--------------------------------------------


Similarity Constant: 0.5
Overall Precision : 0.74482757
Overall Recall : 0.84210527
```

Glossary

Cluster
a group of documents which are all related to some topic

Cluster Description
a cluster Description is a set of phrases that subsets of the documents in the cluster share in common

Cluster Name
a short description identifying which gives the expected topic of the cluster

Clustered Search
a Search Engine with a Web Page Search Clustering extension

Clustering Algorithm
an algorithm which given a data set returns clusters of related data

Common Link
a link which two or more pages have in common, this link is common to the set of pages containing it

Common Phrase
a phrase which two or more pages have in common, this phrase is common to the set of pages containing it

Common Word
a word which two or more pages have in common, this word is common to the set of pages containing it

Data Entry
entering in search keywords, selecting filters or similar

Document
see 'Web Page'

Filter
see 'Search Filter'

Hierarchical Clusters
clusters that form a hierarchy with some clusters being subsets of other clusters

In-Link
a link to page y on a page x, is an in-link of page y

In-Linking Page

y is an in-linking page when a link to page x is on page y.

In-Linking Page Text
the text on an in-linking page

Linear Ordered List
list of web pages displayed in a linear fashion one after another

Link Information
information about the content of a page which can be obtained by the out-links and in-links of the page

Mobile User
a user using a device such as a cell-phone, PDA or other similar device

Ordered List
see 'Linear Ordered List'

Out-Link
a link to page y on a page x, is an out-link of page x

Out-Linked Page
y is an out-linked page when a link to page y is on page x

Out-Linked Page Text
the text on an out-linked page

Overlapping Clusters
clusters that share pages in common

Page
see 'Web Page'

Phrase
a string of words

Phrasal Information
information about the content of a page which can be obtained through the phrases found on the page

Query
see 'Web Query'

Related Pages
a set of pages are related if they can all be related to the some topic x

Refinement
see 'Search Refinement'

Result Set
a set of web pages that are related to a web query

Results
see 'Result Set'

Search Engine
a web agent which given a set of keywords can provide a list of URLs of web pages which

best match the set of keywords

Search Filter
a condition that pages must have to remain within a result set, when applied to a result set a subset of pages that meet the condition will remain

Search Refinement
narrowing the search results by eliminating one or more topics, this may be achieved by filtering the results or refining the keywords

TLD
see 'Top Level Domain'

Top Level Domain
the rightmost part of a domain name, for example .com, .net, .org, .nz, .au, .uk, .de

Search Results
see 'Result Set'

User
normally a human, but possibly a machine which wants to use the clustered search

URL
Uniform Resource Locator, the address of a web page

Web Page
a document on the world wide web, typically in HTML format

Web Page Cluster
a set of related web pages

Web Page Clustering
finding web page clusters from a set of web pages

Web Page Search Clustering
applying web page clustering to search results

Web Search Clustering
see 'Web Page Search Clustering'

Web Query
a text phrase the specifies what pages a user would like to find

Bibliography

[1] ALI, R., GHANI, U., AND SAEED, A. Data clustering and its applications.

[2] ALTAVISTA. http://www.altavista.com, 2004.

[3] ARUL PRAKASH ASIRVATHAM, K. K. R. Web page classification based on document structure.

[4] ASK JEEVES. Teoma search - http://www.teoma.com, 2004.

[5] BITAR, W. Generalized suffix trees and some of their applications - introduction to bioinformatics - http://www.msci.memphis.edu/ giri/compbio/f00/wally/wally.html, September 2000.

[6] CAO, G., SONG, D., AND BRUZA, P. Suffix tree clustering on post-retrieval documents, July 2003.

[7] CHIU WONG, W. *Incremental Document Clustering for Web Page Classification*. PhD thesis, 2000.

[8] CHIU WONG, W., AND FU, A. Incremental document clustering for web page classification. In *IEEE 2000 Int. Conf. on Info. Society in the 21st century: emerging technologies and new challenges (IS2000), Japan* (November 2000).

[9] DMOZ. Open directory project - http://www.dmoz.org, 2004.

[10] GOOGLE. http://www.google.com, 2004.

[11] GROXIS. Grokker - http://www.groxis.com/service/grok, 2004.

[12] HENZINGER, M. Link analysis in web information retrieval. In *IEEE Data Engineering Bulletin, 23(3)* (September 2000), pp. 3–8.

[13] HOU, J., AND ZHANG, Y. Utilizing hyperlink transitivity to improve web page clustering. In *Proceedings of the Fourteenth Australasian database conference on Database technologies, Adelaide, Australia* (2003), vol. 17, pp. 49–57.

[14] INFONETWARE. http://www.infonetware.com, 2004.

[15] INFOSPACE. Dogpile - http://www.dogpile.com, 2004.

[16] LOOKSMART. Wisenut - http://www.wisenut.com, 2004.

[17] MACKAY, D. J. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[18] MANGIONE, C. Performance tests show java as fast as c++. In *Java World* (February 1998).

[19] MICROSOFT CORPORATION. Msn search - http://www.msn.com, 2004.

[20] MYSQL. Database server - http://www.mysql.com, 2004.

[21] OPEN TEXT CORPORATION. Queryserver - http://www.queryserver.com, 2004.

[22] PORTER, M. F. An algorithm for suffix stripping. *Program 14*, 3 (July 1980), 130–137.

[23] REINERT, K., AND GRPL, C. An exposition based on Dan Gusfield's, algorithms on strings, trees, and sequences. computer science and computational biology. Cambridge University Press. 1997. Pages 99ff, ISBN 0-521-58519-8.

[24] SHAKESPEARE, W. Hamlet, 1602.

[25] STEINBACH, M., KARYPIS, G., AND KUMAR, V. A comparison of document clustering techniques. In *KDD Workshop on Text Mining* (2000).

[26] SUN MICROSYSTEMS. Sun's official java website - http://java.sun.com, 2004.

[27] TANG, X. A critical analysis of clustering for web search.

[28] TONELLA, P., RICCA, F., PIANTA, E., GIRARDI, C., ITC-IRST, LUCCA, G. D., FASOLINO, A. R., TRAMONTANA, P., DI NAPOLI FEDERICO II, U., NAPOLI, AND ITALY. Evaluation methods for web application clustering. In *5th International Workshop on Web Site Evolution, Amsterdam, The Netherlands* (September 2003).

[29] VIVISIMO. http://www.vivisimo.com, 2004.

[30] WANG, Y., AND KITSUREGAWA, M. Combining link and contents in clustering web search results to improve information interpretation. In *Proceedings of 2002 Data Engineering Workshop (DEWS'2002), Kurasiki, Japan* (March 2002), pp. C4–2.

[31] WANG, Y., AND KITSUREGAWA, M. Evaluating contents-link coupled web page clustering for web search results. In *Proceeding of 11thInternational conference on Information and Knowledge Management (CIKM 2002), McLean, VA, USA. ACM Press.* (2002), pp. 499–506.

[32] WANG, Y., AND KITSUREGAWA, M. On combining link and contents information for web page clustering. In *13th International Conference on Database and Expert Systems Applications DEXA2002, Aix-en-Provence, France* (September 2002), pp. 902–913.

[33] WANG, Y., AND KITSUREGAWA, M. Use link-based clustering to improve web search results. In *Proceeding of 2nd International conference on Web Information System Engineering (WISE2001), IEEE Computer Society* (December 2002), pp. 115–124.

[34] WANG, Z. Improved link-based algorithms for ranking web pages.

[35] WEISS, R., VELEZ, B., SHELDON, M. A., NEMPREMPRE, C., SZILAGYI, P., DUDA, A., AND GIFFORD, D. K. Hypursuit: A hierarchical network search engine that exploits content-link hypertext clustering. In *Proceedings of the Seventh ACM Conference on Hypertext, Washington, DC* (March 1996).

[36] XUE, G.-R., ZENG, H.-J., CHEN, Z., MA, W.-Y., ZHANG, H.-J., AND LU, C.-J. Implicit link analysis for small web search. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, Toronto, Canada* (2003), pp. 56–63.

[37] YAHOO! http://www.yahoo.com, 2004.

[38] ZAMIR, O. *Clustering Web Documents: A Phrase-Based Method for Grouping Search Engine Results*. PhD thesis, 1999.

[39] ZAMIR, O., AND ETZIONI, O. Web document clustering: A feasibility demonstration. In *Research and Development in Information Retrieval* (1998), pp. 46–54.

[40] ZAMIR, O., AND ETZIONI, O. Grouper: a dynamic clustering interface to Web search results. *Computer Networks (Amsterdam, Netherlands: 1999) 31*, 11–16 (1999), 1361–1374.