DOCUMENT CLUSTERING ON TARGET ENTITIES USING PERSONS AND ORGANIZATIONS

JEREMY R. KEI

National University of Singapore

2003

DOCUMENT CLUSTERING ON TARGET ENTITIES USING PERSONS AND ORGANIZATIONS

BY

JEREMY R. KEI

(B. Sc. Hons, NUS)

A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF SCIENCE

Table of Contents

List of Tables

List of Figures

Abstract

Web surfing often involves carrying out information finding tasks using online search engines. These searches often contain keywords that are names, as in the case of Persons and Organizations (abbreviated “PnOs”). Such names are often not distinctive, commonly occurring, and non-unique. Thus, a single name may be mapped to several named entities. The result is users having to sift through mountains of pages and put together manually a set of information pertaining to the target entity in query.

In an effort to circumvent this inconvenience, a new methodology to cluster the Web pages returned by the search engine has been conceived. The PnOClassifier system relies on innovative feature space reductions, high-quality small sample-size classifier training, partitioning and rule inductions. This unsupervised approach works in a way so that pages belonging to different entities are clustered into different groups automatically. The algorithm uses a combination of named entities, link-based, structure-based and content-based information as features to partition the document set into direct, indirect and irrelevant pages. In the process, a general-purpose web-page decision-tree classifier is trained and modeled after our test collections and set to work on new queries, such that it chooses the distinct direct pages as seeds to cluster the document set into different clusters. The PnOClassifier system also represents

another important towards our objective to automatically and intuitively generate reader-centric partitions of collections of documents. That said, the system can be adapted to specific domains of web pages on the Internet based on user queries on names of Persons and Organizations.

The exact contributions to document clustering techniques applicable to the vast and varied collections of World Wide Web are therefore summarized as follows. First, a Named Entity (NE) based feature identification and extraction strategy is proposed. This PnO mechanism is capable of dealing with target entity related document clustering. For our purpose, we selected text documents in the English language on Persons and Organizations as the target of our experimentation. Second, we combined conventional clustering techniques in hierarchical and partitioning approaches to incrementally improve the performance of the algorithm. Third, we programmatically realized the proposed PnO mechanism through a pipeline implementation of PnO NE-based components. Fourth, we show that the induced rules generated by our cross-validated training data are meaningful and understandable. Fifth, the clusters produced by the trained PnOClassifier pipeline when fed both small or reasonably big input data is of high-quality, with results comparable to that of recent TREC efforts and systems in related categories. Finally, the proposed approach to document clustering can handle “feature noise” effectively without undue reduction in quality of resultant clusters. The document clusters produced by the PnOClassifier pipeline is seen to be more humanized and reader-

centric. Search results are also partitioned by human subjects and placed alongside with clusters produced by the system and judged.

Our approach is unique in its PnO target entity focus, and to the best of our knowledge there is no existing system running close to this effort. The pipeline algorithms we have proposed and implemented is effective in addressing Web-based document clustering. Some of the potential usage scenarios and extensions will be covered.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval - Selection process

General Terms

Algorithms, Performance, Experimentation

Key Words

Web clustering, persons and organizations, machine learning, text classification, information retrieval, named entities

1 Introduction

Information finding is a regular task performed during online Internet surfing. It is ubiquitous knowledge that search engines on the web produces hits on objects, people, companies and of other targets using terms we supply in our query. At other times, users may use the more esoteric features offered by individual search engines or meta crawlers to refine or narrow down their searches. For instance, search engines such as Google, Yahoo! and Altavista offer Boolean operators on keywords supplied as query terms. In addition, we can also supply specific names of these target entities to further constrain the returned document set. For instance, searching for "laptop" may return multiple hits from different vendors, whereas "IBM and laptop" produces an immediately constrained query result set on mobile stations produced by the aforementioned vendor.

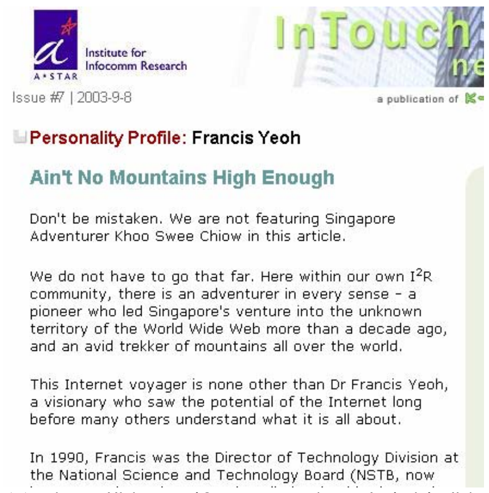
This dissertation describes research into techniques on feature detection and identification for target entity-based document clustering on the World Wide Web. In particular, we focus on and compare results returned for queries about Persons and Organizations. Top ranked results retrieved by search engines on these entities are usually sufficiently accurate for its purpose. However, while they usually include the target entity in the query, they encompass many observable problems and issues outlined below:

- The number of pages returned by a search engine may reach thousands. However, most users only have patience to browse the first few pages only.
- Search results may contain several different target entities whose names are the same as the query string. It would facilitate user browsing if the search results can be grouped into different clusters, each containing pages about different entities.
- Some useless pages are completely irrelevant but are displayed nonetheless as return results because they contain phrases that are similar to the name of requested PnOs. For example, a fable page or AI research page may appear in the query of “Oracle”, when the user is only interested to find information about the software company “Oracle Corp”.
- The low-ranking pages listed at the rear of the result list may often be of only minor importance, but they are not always useless. In some cases, novel or unexpectedly valuable information can be found in these pages.

As shown in Figure 1, when we submit the query "Francis Yeoh" to Google (www.google.com), at least 3 different persons named " Francis Yeoh" will be returned. Here, pages (a) and (b) are the homepages of two different persons: an Entrepreneur in Singapore and another in Malaysia. Page (c) refers to a General Manager in a London Studio, though its style is different from that of the earlier pages. It is however unclear whether the person in (c) is the same as the one in (a) or (b).

It can be seen that the search engine returns a great variety of both related and unrelated results. If we are able to identify and partition the results into clusters about different target entities according to their ownership, for example, in this case, into three clusters for three different individuals, it will facilitate users in browsing the results.

The aim of this research is to develop a search utility to support PnO searches on the Web. In particular, it partitions the search results returned by a PnO name query into distinct clusters, with each containing document pages about a particular target entity. For instance, for search on person named “Francis Yeoh”, we expect to get one cluster about Francis Yeoh in Singapore, another about Francis Yeoh in Malaysia, etc. The unknown fragment pages are discarded into an unknown cluster. So it is different from general document and web clustering problems.



Institute for Infocomm Research

InTouch

Issue #7 | 2003-9-8

a publication of

Personality Profile: Francis Yeoh

Ain't No Mountains High Enough

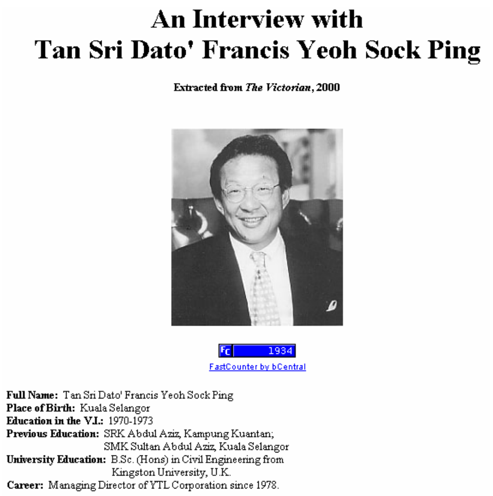
Don't be mistaken. We are not featuring Singapore Adventurer Khoo Swee Chiow in this article.

We do not have to go that far. Here within our own I^{2}R community, there is an adventurer in every sense - a pioneer who led Singapore's venture into the unknown territory of the World Wide Web more than a decade ago, and an avid trekker of mountains all over the world.

This Internet voyager is none other than Dr Francis Yeoh, a visionary who saw the potential of the Internet long before many others understand what it is all about.

In 1990, Francis was the Director of Technology Division at the National Science and Technology Board (NSTB, now

(a) http://kbatsu.i2r.a-star.edu.sg/cti_bin/kbatsu/letter/07/



An Interview with
Tan Sri Dato' Francis Yeoh Sock Ping

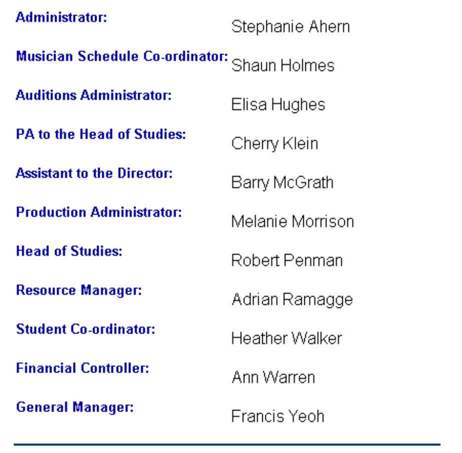
Extracted from *The Victorian*, 2000

FC 1934

FastCounter by bCentral

Full Name: Tan Sri Dato' Francis Yeoh Sock Ping
Place of Birth: Kuala Selangor
Education in the V.I.: 1970-1973
Previous Education: SRK Abdul Aziz, Kampung Kuantan; SMK Sultan Abdul Aziz, Kuala Selangor
University Education: B.Sc. (Hons) in Civil Engineering from Kingston University, U.K.
Career: Managing Director of YTL Corporation since 1978.

(b) http://viweb.freehosting.net/viint_F-Yeoh.htm



Administrator:	Stephanie Ahern
Musician Schedule Co-ordinator:	Shaun Holmes
Auditions Administrator:	Elisa Hughes
PA to the Head of Studies:	Cherry Klein
Assistant to the Director:	Barry McGrath
Production Administrator:	Melanie Morrison
Head of Studies:	Robert Penman
Resource Manager:	Adrian Ramagge
Student Co-ordinator:	Heather Walker
Financial Controller:	Ann Warren
General Manager:	Francis Yeoh

(c) http://www.london-studio-centre.co.uk/staff_directory.html

Figure 1. Typical pages when "Francis Yeoh" is submitted to Google (Partial list)

To support this process, we need to identify three types of pages from the returned pages:

- Direct page (DP): Its content is almost entirely about the users' focus. Examples of such pages include the homepages, profiles, resumes, CVs, biographies, synopsis, memoirs, etc. The relevance between them and the query is the highest and could be selected as the seed (center) of the corresponding cluster.

- Indirect page (IDP): In such pages, the target entity is only mentioned occasionally or indirectly. For instance, the person’s name may appear in a page about the staff of a company, record of a transaction, or the homepage of his friend.

- Irrelevant page: the page is not about any target entity named as the query string.

We use a combination of named entities, link and structure information extracted from the original content as features to perform the clustering. Our tests indicate that this approach is promising. The main contribution of this research is in providing an effective clustering methodology for PnO pages.

The contents of this effort are organized as follows. Section 2 introduces related work and Section 3 discusses named entity based, link-based, content-based and structure-based document features and presents the algorithm to identify DPs and seeds of the clusters. The method of delivering IDPs into clusters is described. The implementation of the PnOClassfier system is detailed in Section 4. The results of our experiments and the conclusions are presented in Section 5 and conclusions with future directions outlined in Section 6.

2 Related Work

2.1 Common Document Clustering Algorithms

Document Clustering algorithms attempt to identify groups of documents that are similar to each other more than the rest of the collection. Here each document is represented as a weighted attribute vector, with each word in the entire document collection being an attribute in this vector (vector-space model [1]). Besides probabilistic technique (such as Bayesian), a priori knowledge for defining a distance or similarity among them is used to compare two documents. Common clustering algorithms employing hierarchical and partitioning approaches are based on these basic principles of feature vector representation [38].

One of the important tasks in our research is to develop techniques to identify direct pages to PnO queries. Our direct page finding task is similar to but more complex than the home (entry) page and key resource finding tasks in TREC [2] [3]. The homepage finding task [3] aims to find the home or site entry page about the topic. The home page usually has introductory information about the site and navigational links to other pages in the site. It is a subset of direct page as a direct page may include other type of PnO related pages such as the resume or profile. The key resource finding task [3] aims to find pages that contain lots of information, usually in the form of links to relevant pages, about the topic. A key resource page can therefore be located based on the number of out-links a page has to useful authority pages. In contrast, a direct page is more self-contained and includes useful information about a specific PnO with links to other pages within the sites.

The main approaches for finding homepages exploit content information as well as URL and link structure [5]. It was generally found that using only content information could achieve a mean reciprocal rank (MRR) score of only 30% based on the top 10 ranked results. However, combining content with anchor text and URL depth [5] could achieve an MRR of 77.4%, which is the best reported result in TREC10 evaluations. Craswell, et al. [7] confirmed that ranking based on link anchor text is twice as effective as ranking based on document content. Kraaij, et al. [8] further analyzed the importance of page length, the number of incoming links and URL form such as whether it is of type root, sub-root, index or ordinary file. They discovered that URL form was a good predictor of home pages. Xi & Fox [9] reported a learning–based approach that uses decision tree followed by regression analysis to filter out homepages using the document features of URL depth, number of in- and out-links, keywords, etc. They reported a MRR of over 80% on a subset of WT10g corpus. These works indicate that homepage finding depends largely on information beyond contents, where URLs, links and anchors play important roles.

For key resource task, Zhang et al. [10] employed techniques based on link structure, link text and URL, especially the out-degree, of the pages. They achieved the best results in TREC-11 evaluation with a precision of 25% among the top 10 retrieved pages. However, the second best performing run [11] was a straightforward content retrieval run based on Okapi BM25, and achieved a precision of about 24%. The overall results reveal that the page content is as good as non-content features in key resource finding task.

After we have found distinct direct pages for target entities, the second stage is to perform clustering to deliver IDPs for the corresponding Target entities. PnO page clustering is a special case of web document clustering, which attempts to identify groups of documents that are more similar to each other than the rest of the collection. Information foraging theory [12] notes that there is a trade-off between the value of information and the time spent in finding it. The vast quantity of Web pages returned as the search result means that clustering or summarization of the results is essential. Several new approaches have emerged to group or cluster Web pages. These include association rule hyper-graph partitioning, principal direction divisive partitioning [12], and suffix tree clustering [14]. The Scatter/Gather technique [14] clusters text documents according to their similarities and automatically computes an overview of documents in each cluster. Steinbach et al. [15] compared a number of algorithms for clustering web pages on a variety of test corpuses. Their reported performance in terms of F1 measure varies from 0.59 to 0.86.

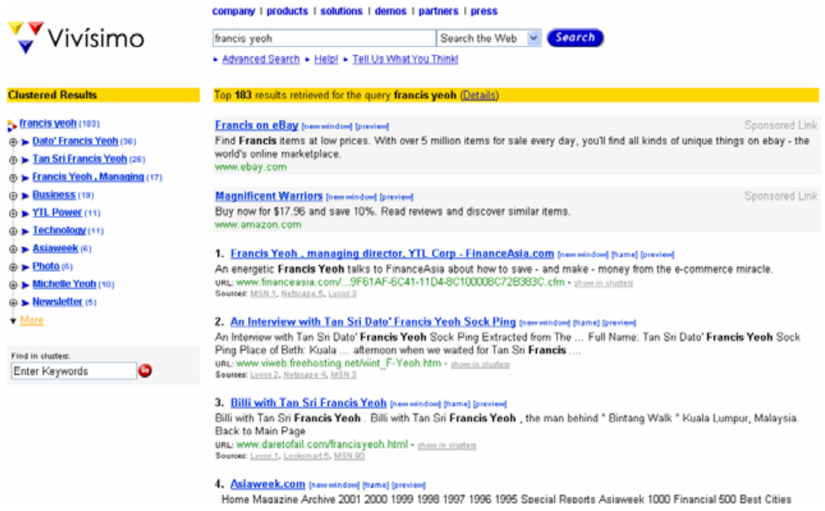
Many of these traditional algorithms employ the bag of words representation to model each document. The resulting feature space tends to be very large, in the order of ten of thousands. As a result, most traditional clustering algorithms falter due to the problem of data sparseness when the dimensionality of the feature space becomes high relative to the size of document space. Because of the unpredictable performance of clustering methods, most search engines at present do not deploy clustering as a regular procedure during information retrieval.

2.2 Meta-Search Engines Compared

Meta-search crawlers, the multi-faceted engines that used to sift through the mountains of web pages indexed by the web's independent search engines are no longer simple collators. Some modern-day meta-crawlers possess distinctive capabilities that make them good alternatives in terms of document coverage to main-stream reader-oriented engines as either a starting point or as a supplementary search tool. Google, currently one of the largest search engines online, covers limited parts of the web, albeit some portions are months out of date [39]. However, one cannot expect to see good search results all of the time, especially when some engines are tuned specifically for a particular methodology such as topical clustering, or into collections of specialty databases. It is difficult to compare the effectiveness and efficiency of different cluster approaches and systems in the absence of well-known or authoritatively representative testing methodologies or evaluation measures. Here an empirical approach is taken to evaluate the engines practically by submitting our queries to them. We document the examples for the particular querying and clustering

PnO pages below, which in corollary also demonstrate some benefits of our PnO NE approach.

One of these commercial document clustering engines, Vivisimo (www.vivisimo.com), is best known for its human-readable "folders", or topics into which it groups search results. This is determined by analyzing title and URL and a short description extracted from page content, with the resulting folders or topics arranged hierarchically. Our clustering category is however different from Vivisimo, where the similarity is determined by word similarity, but not the ownership of target entity. For example, the clustered results for "Francis Yeoh" by Vivisimo include 183 pages (each search returns a default of 500 results at the time of this research) shown in first 10 clusters, such as Dato' Francis Yeoh, Tan Sri Francis Yeoh, Business, YTL Power, Technology, Asiaweek, and so on (Figure 2). Here we observed that the content about the particular target entity, Francis Yeoh in Green Dot Internet Services appear in cluster Technology, while multiple targets are spread over the first 3 clusters. It is evident from this simple example that this presentation approach is not the best solution for PnO query tasks when users are interested in the particular target entity. Another example is the query about organization "Mobile Payment". Vivisimo provide 362 pages in first 10 clusters (Mobile Payment Forum, Payment Systems, Card, Payment Solutions, Mobile Payment Services, Wireless, Business, Press Releases, Phones and New Mobile Payment). Again, these clusters do not correspond to any specific entities that we require.



francis yeoh

Top **183** results retrieved for the query **francis yeoh** (Details)

- francis yeoh (183)
  - Dato' Francis Yeoh (36)
  - Tan Sri Francis Yeoh (26)
  - Francis Yeoh , Managing (17)
  - Business (19)
  - YTL Power (11)
  - Technology (11)
  - Asiaweek (6)
  - Photo (6)
  - Michelle Yeoh (10)
  - Newsletter (5)

Francis on eBay [new window] [preview] Sponsored Link
Find **Francis** items at low prices. With over 5 million items for sale every day, you'll find all kinds of unique things on ebay - the world's online marketplace.
www.ebay.com

Magnificent Warriors [new window] [preview] Sponsored Link
Buy now for $17.96 and save 10%. Read reviews and discover similar items.
www.amazon.com

1. Francis Yeoh , managing director, YTL Corp - FinanceAsia.com [new window] [frame] [preview]
An energetic **Francis Yeoh** talks to FinanceAsia about how to save - and make - money from the e-commerce miracle.
URL: www.financeasia.com/...9F61AF-6C41-11D4-8C100008C72B383C.cfm - show in clusters
Sources: MSN 1, Netscape 5, Lycos 3

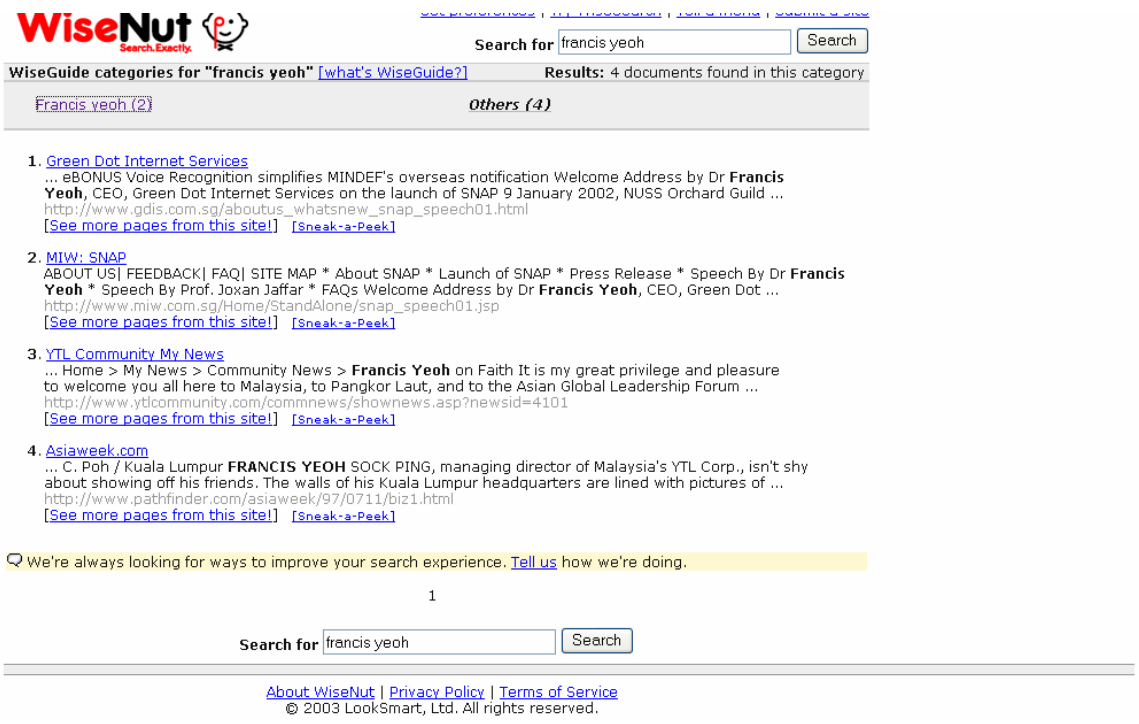
2. An Interview with Tan Sri Dato' Francis Yeoh Sock Ping [new window] [frame] [preview]
An Interview with Tan Sri Dato' **Francis Yeoh** Sock Ping Extracted from The ... Full Name: Tan Sri Dato' **Francis Yeoh** Sock Ping Place of Birth: Kuala ... afternoon when we waited for Tan Sri **Francis** ....
URL: www.viweb.freehosting.net/viint_F-Yeoh.htm - show in clusters
Sources: Lycos 2, Netscape 4, MSN 3

3. Billi with Tan Sri Francis Yeoh [new window] [frame] [preview]
Billi with Tan Sri **Francis Yeoh** . Billi with Tan Sri **Francis Yeoh** , the man behind " Bintang Walk " Kuala Lumpur, Malaysia. Back to Main Page
URL: www.daretofail.com/francisyeoh.html - show in clusters
Sources: Lycos 1, Looksmart 6, MSN 90

4. Asiaweek.com [new window] [frame] [preview]
Home Magazine Archive 2001 2000 1999 1998 1997 1996 1995 Special Reports Asiaweek 1000 Financial 500 Best Cities

Figure 2. Vivisimo Search Results

Another commercial search engine that performs document clustering is WiseGuide (http://www.wisenut.com). When we submitted "Francis Yeoh" to WiseGuide, it returned only six pages in two clusters: "Francis Yeoh" and "Others". Here the web pages are not partitioned by their ownership. We need to browse both the two clusters, though our focus is only on one particular target entity. For "Mobile payment" query, WiseGuide returned 20,240 documents in a hierarchical category (Figure 3), where there are four labels, Mobile Payment, Press Releases, World First and others, listed in the first layer. Obviously, we cannot link any particular target entity to the cluster with the above names. WiseNut uses a combination of content-based words, links and entropy measures based features [30], thus it is unable to cluster returned documents into separate entity groups as desired.



Search for francis yeoh Search

WiseGuide categories for "francis yeoh" [what's WiseGuide?] **Results:** 4 documents found in this category

Francis yeoh (2) ***Others (4)***

1. Green Dot Internet Services
... eBONUS Voice Recognition simplifies MINDEF's overseas notification Welcome Address by Dr **Francis Yeoh**, CEO, Green Dot Internet Services on the launch of SNAP 9 January 2002, NUSS Orchard Guild ...
http://www.gdis.com.sg/aboutus_whatsnew_snap_speech01.html
[See more pages from this site!] [Sneak-a-Peek]

2. MIW: SNAP
ABOUT US| FEEDBACK| FAQ| SITE MAP * About SNAP * Launch of SNAP * Press Release * Speech By Dr **Francis Yeoh** * Speech By Prof. Joxan Jaffar * FAQs Welcome Address by Dr **Francis Yeoh**, CEO, Green Dot ...
http://www.miw.com.sg/Home/StandAlone/snap_speech01.jsp
[See more pages from this site!] [Sneak-a-Peek]

3. YTL Community My News
... Home > My News > Community News > **Francis Yeoh** on Faith It is my great privilege and pleasure to welcome you all here to Malaysia, to Pangkor Laut, and to the Asian Global Leadership Forum ...
http://www.ytlcommunity.com/commnews/shownews.asp?newsid=4101
[See more pages from this site!] [Sneak-a-Peek]

4. Asiaweek.com
... C. Poh / Kuala Lumpur **FRANCIS YEOH** SOCK PING, managing director of Malaysia's YTL Corp., isn't shy about showing off his friends. The walls of his Kuala Lumpur headquarters are lined with pictures of ...
http://www.pathfinder.com/asiaweek/97/0711/biz1.html
[See more pages from this site!] [Sneak-a-Peek]

We're always looking for ways to improve your search experience. Tell us how we're doing.

1

Search for francis yeoh Search

About WiseNut | Privacy Policy | Terms of Service
© 2003 LookSmart, Ltd. All rights reserved.

Set preferences | Try WiseSearch | Tell a friend | Submit a site

Search for francis yeoh Search

WiseGuide categories for "francis yeoh" [what's WiseGuide?] **Results:** 4 documents found in this category

Francis yeoh (2) ***Others (4)***

1. Green Dot Internet Services
... eBONUS Voice Recognition simplifies MINDEF's overseas notification Welcome Address by Dr **Francis Yeoh**, CEO, Green Dot Internet Services on the launch of SNAP 9 January 2002, NUSS Orchard Guild ...
http://www.gdis.com.sg/aboutus_whatsnew_snap_speech01.html
[See more pages from this site!] [Sneak-a-Peek]

2. MIW: SNAP
ABOUT US| FEEDBACK| FAQ| SITE MAP * About SNAP * Launch of SNAP * Press Release * Speech By Dr **Francis Yeoh** * Speech By Prof. Joxan Jaffar * FAQs Welcome Address by Dr **Francis Yeoh**, CEO, Green Dot ...
http://www.miw.com.sg/Home/StandAlone/snap_speech01.jsp
[See more pages from this site!] [Sneak-a-Peek]

3. YTL Community My News
... Home > My News > Community News > **Francis Yeoh** on Faith It is my great privilege and pleasure to welcome you all here to Malaysia, to Pangkor Laut, and to the Asian Global Leadership Forum ...
http://www.ytlcommunity.com/commnews/shownews.asp?newsid=4101
[See more pages from this site!] [Sneak-a-Peek]

4. Asiaweek.com
... C. Poh / Kuala Lumpur **FRANCIS YEOH** SOCK PING, managing director of Malaysia's YTL Corp., isn't shy about showing off his friends. The walls of his Kuala Lumpur headquarters are lined with pictures of ...
http://www.pathfinder.com/asiaweek/97/0711/biz1.html
[See more pages from this site!] [Sneak-a-Peek]

We're always looking for ways to improve your search experience. Tell us how we're doing.

1

Search for francis yeoh Search

About WiseNut | Privacy Policy | Terms of Service
© 2003 LookSmart, Ltd. All rights reserved.

Figure 3. KillerInfo Search Results

KillerInfo (http://www.killerinfo.com/), another content aggregator, also uses Vivisimo's clustering technology. In addition to its Vivisimo-based baseline indexes, it also carries databases for specialty sources in news, healthcase, law, sciences, and other subject areas. This makes it a more domain-independent crawler, unlike Vivisimo, it does not have to be customized specifically for one index. Manual search results however does not appear to result in any gains in performance nor effectiveness as the final clusters are too wide from a user's point of view. Ez2wWw.com, a meta-search portal from Holomedia, also includes aspect-based information databases spanning across popular reader-oriented news, weather and currencies customizable to a particular geographical region. The global meta-search provides for seven engines and on-page controls for number of hits and search time allotment. The Advanced Search supports parallel searching of more than 1,000 specialty databases organized by subject, from the arts to Web design. A summary at the bottom of the page reports the number of hits retrieved from each engine. Setting the search at a larger depth can increase the number retrieved. Search results from the global search (but not necessarily from advanced search) are grouped into clusters based on frequently occurring phrases. Infonetware operates at another level of sophistication with the use of text analysis in its results manipulation. Terms are extracted from the results set and presented in index-style formatting with documents ranked by relevance. Infonetware offers a Quick View and Drill Down option allowing users to narrow down and combine or exclude terms and documents, effectively similar to query modification. The clustering features make these meta-searchers very useful for broad, exploratory queries. The topics can bring out alternate contexts, patterns, and main themes. Larger result sets are ideal for meta-searchers because they provide better granularity.

However, as shown in the actual usage and screenshots of the clusters returned by the engines, it is evident that the results are determined by bag-of-words similarity approaches and not based on the target entities we so desire. Instead, different people with the similar names are aggregated together in the same cluster. This does not make it easier for the user to sift through the document results. In addition, from our practical experiments in using these engines, we found that pages we expect to be returned as clusters are not in the target results set. The issue of directing document clusters at the people who will read them is a crucial factor that will make the resultant clusters of documents useful. This makes our approach at clustering and aggregating PnO target-based information competitively unique and more ergonomically useful.

3 Document Feature Representation

Most clustering approaches compute the similarity (distance) between a pair of documents using the cosine of the angle between the corresponding vectors in the feature space. Many techniques, such as TFIDF and stop word list [16], have been used to scale the feature vectors to avoid skewing the result by different document lengths or possibly by how common a word is across many documents. However, they do not work well for PnOs. For instance, given two resume pages about different persons, it is highly possible that they are grouped into one cluster because they share many similar words and phrases, such as the words “graduate”, “university”, “work”, “degree”, “employment” and so on. This is especially so when their style, pattern and glossary are also similar. On the other hand, it is difficult to group together a news page and resume page about the same target entity, due to the diversity in subject matter, word choice, literary styles, document formats and length among them. To solve this problem, it is essential to choose the right set of features that reflect the essential characteristics of target entities.

In general, we observe that PnO named entities (PnO NEs) in the web pages about PnOs are higher than that in the other type of pages. In a direct page (DP), there is typically a large number of PnO NEs, such as the names of graduation schools, contact information (phone, fax, e-mail, and address), working organizations and experiences (time and organizations). Here, PnO related NEs include person, location and organization name, time and date, fax/phone number, currency, percentage, e-mail and so on. For simplicity, we called these entities collectively as PnO NEs. We could therefore use PnO NEs as the basis to identity PnO pages. To support our claim, we analyzed 1,000 PnO pages together with 1,000 other type of pages that we randomly obtained from the Web. We found that the percentage of PnO NEs in PnO direct pages is at least 6 times higher than that in other types of pages, if we ignore PnO NEs of type number and percentage. We could therefore use PnO NEs as the basis to identity PnO pages.

The finding is quite consistent with intuition, as PnO NEs play important roles in semantic expression and could be used to reflect content of the pages, especially when human activities are depicted. The typical number of PnO NEs appearing in the results of a search is typically around hundreds or thousands, which means that it is feasible to use them as the features of search results about PnOs. Our analysis also shows that PnO NEs is good in partitioning pages belonging to different persons or organizations, and the use of frequent phrases and words, such as degree, education, work etc, is not effective for this task.

However, not all pages with many PnO NEs are DPs. Examples of such pages include attendee lists of conferences and stock price lists etc. We thus need to further check the roles played by the PnO NEs in this text. The rationale is that a DP is highly likely to repeat its name in its URL, title, or at the beginning of its page. In general, if the target entity appears in important locations, such as in HTML tags <title>, <H1> and <H2>, or appears frequently, then the corresponding pages should be DPs and their topic is about the users' target. We could detect the trace of page topic using the technology like wrapper rules [17] to decipher the structure information of the page.

Furthermore, we know from the TREC evaluations that URL, HTML structure and link structure tend to contain important heuristic clues for web clustering and information retrieval [17]. Links could be used to improve document ranking, estimate the popularity of a web page, and extract the most important hubs and authorities related to a given topic [19]. Moreover, links, URLs and anchors could improve the results of the content-only approach for IR [5]. A short DP, even though it may contain few PnO NEs, usually has many links to those pages referring to the target entity. The positions of and the HTML markup tags around the PnO NEs could provide hints to the role of these entities in the corresponding page. To better identify the role of links in DP, we further identify the form of URLs as: root (entry page of site), sub-root, index and ordinary file. The URL form has been found in [7] to be a particularly good predictor for finding home pages.

Based on the above discussion, we combine three categories of features to identify DPs and IDPs. They are the named entities, links and structure-based features. The resulting set of features, as listed in Table 1, can be considered as original feature transformation. As the number of such features is smaller than the number of tokens in the collection, there is considerable dimension reduction. This will alleviate the problem of low quality of clustering because of data sparseness when the sample size is small.

3.1 Identifying Direct Pages as Cluster Seeds

DPs (Direct pages) can be used as candidate seeds to divide the retrieved documents into clusters of distinct target entities. In case where there is more than one DP about a target entity, we need to select the best one as the seed for clustering. To select the best DP of a target entity, we therefore need to solve two problems. First we must be able to identify a DP from the collection. Second, in the case of multiple DPs for the same target entity, we must be able to select the best one.

The process is carried out as follows. First we view the identification of DPs as a classification problem of dividing the document collection into the DP and IDP sets. Here we employ the decision tree to predict whether a page is a DP or IDP based on the feature set as listed in Table 1.

Table 1. Features of web pages representation

No.	Feature	Explanation
1	PERSONS_COUNT	Number of persons

2	PERSONS_NE_RATIO	Number of persons to total number of Named Entities ratio
3	ORGANIZATIONS_COUNT	Number of organizations
4	ORGANIZATIONS_NE_RATIO	Number of organizations to total number of Named Entities ratio
5	EMAILS_COUNT	Number of E-Mail addresses
6	NUMBERS_COUNT	Number of numeric; fax, phone number and zip code are included; but the series of number list are ignored
7	PERCENTAGES_COUNT	Specific count of percentages (numbers or alphanumeric) are included; but the series of number list are ignored
8	DATES_COUNT	Specific count of dates (numbers or alphanumeric) are included; but the series of number list are ignored
9	PHONES_COUNT	Specific count of phone numbers are included; but the series of number list are ignored

10	MONEY_COUNT	Specific count of financial figures (numbers or alphanumeric) are included; but the series of number list are ignored
11	FTP_COUNT	Number of FTP links
12	FTP_URLS_RATIO	Number of FTP links to total URLS ratio
13	HTTP_COUNT	Number of HTTP links
14	HTTP_URLS_RATIO	Number of HTTP links to total URLS ratio
15	NE_TOTAL	Sum of the above PnO NEs
16	WORDS_TOTAL	Number of words in a page excluding the HTML tags
17	TOKENS_TOTAL	Number of all tokens
18	NE_TOKENS_RATIO	Ratio of NE_TOTAL and TOKENS_TOTAL
19	NE_WORDS_RATIO	Ratio of NE_TOTAL and

		WORDS_TOTAL
20	TARGET_TITLE	Boolean; whether target entity or its variant appears in the title, head or the beginning of the page; e.g. “Francis Yeoh Homepage”
21	QUERY_TITLE_RATIO	A statistical representation of TARGET_TITLE, determines how many segments of the query matches the title of the document.
22	URLS_IN	Number of incoming links to this page
23	URLS_IN_RATIO	Number of URLS_IN to sum of URLS_IN and URLS_OUT ratio
24	URLS_OUT	Number of outgoing links from this page
25	URLS_OUT_RATIO	Number of out-links to sum of URLS_IN and URLS_OUT ratio
26	URLS_COUNT	The sum of URLS_IN and URLS_OUT
27	URL_SLASH_COUNT	The depth of URL

28	URL_FORM	Four types of forms: root; sub-root (roots of sub-trees); index/path; file. Sub-roots are considered for sub-searches only.
29	TARGET_NE_RATIO	Number of target entities appearing in the page
30	IN_TARGET_URL	Boolean; Whether target entity or its variant appears in URL. E.g. target is “Francis Yeoh" and URL is “http://somewhere.com/~francis/”
31	QUERY_URL_RATIO	A statistical representation of TARGET_URL, determines how many segments of the query matches the title of the document. Sub-roots have normalized ratios taken from the sub-root being index “0”.

Next, we need to resolve the case of multiple DPs found for the same target entity. If we preserved those overlapping DPs in the seed set of clusters, there would appear more than one clusters mapping to the same target entity. We observe that if both the homepage and resume of the same person are selected as DP, then these two pages will share many similar NEs related to this specific person, such as the university graduated, employers, etc. Thus we could evaluate the similarity between two DPs by examining the overlaps in the instances of unique PnO NEs. Here we use TFIDF to estimate the weight of each unique NE as follows.

$$W_{i,j}=tf_{i,j}*\log(N/df_i) \qquad (1)$$

where $tf_{i,j}$ is the number of NE i in page j; df_i is the number of pages containing NE i; and N is the total number of pages.

The normalized similarity of the DPs, p_i and p_j, could therefore be expressed by their cosine distance as:

$$sim(p_i, p_j) = \frac{\sum_k (w_{k,i}{}^c * w_{k,j})}{\sqrt{\sum_k (w_{k,i})^2 * \sum_k (w_{k,j}{}^c)^2}} \qquad (2)$$

If $sim(p_i,p_j)$ is larger than a pre-defined threshold τ_1 (See Algorithm 1), then p_i and p_j are considered to be similar. The page that has more NEs will be used as the seed and the other will be removed. Because the number of DPs is a small fraction of the search results, and the number of PnO NEs in DPs is usually less than hundreds, thus the computational cost in eliminating redundant DPs is acceptable.

Algorithm 1 summarizes the procedure to identify seeds of clusters.

Algorithm 1:

```
Detect_seed (page_set)  {
  set page_set = {the set of all pages found};
  set seed_set=null;  //the collection of candidate seeds

  //select direct pages using decision tree algorithm as follows:
  for each (page $p_i$  in page_set){
    build transformed feature set of $p_i$
    if (decision_tree($p_i$) == TRUE)
      move $p_i$ from page_set into seed_set;
  }

  //eliminate the redundant elements in seed_set
  for each (pair {$p_i$, $p_j$} in seed_set){
    if (Sim ($p_i$,$p_j$)> $\tau_1$) { // are about same target entity
      if ( |NE| in $p_i$ >|NE| in $p_j$)
        move $p_j$ from seed_set into page_set;
      else
        move $p_i$ from seed_set into page_set;
    }

  return seed_set;
}
```

At the end of the process, the pages remaining in the *seed_set* could be used as seeds for the clusters. They are representatives of distinct entities named in the query. Since the elements in seed_set are largely less than that in all page_set after the elements in DPs are chosen using the decision tree module, the calculation cost in comparison between all candidate pairs is acceptable.

The remaining of the candidate seeds (or remaining direct pages) are then evaluated against the cluster seeds and appropriately sent to the closest matching seed based on their corresponding similarity ratios (Algorithm 2). These Direct Pages then make up our entry level bag-of-clusters to which we shall deliver the Indirect Pages. Indirect Pages however do not share the same forthcoming characteristics as Direct Pages, and much less the Seed Pages. Instead, they will be considered to have more ambiguous and conflicting features, along with a host of other possibly irrelevant information. The next section details the algorithms we use in determining how Indirect Pages can be delivered using the 31 attributes as was outlined in the aforementioned discussion.

Algorithm 2:

```
Init_cluster {

 // cluster the rest of the remaining seeds
 for each ({$S_j$} in seed_set) {
   create doc_cluster $C_j$
 }

 // Move remaining candidate pages into each appropriate cluster
 // where similarity of the page to a seed is highest
for each ({$p_i$, $S_j$} in remaining page_set, doc_cluster $C_j$) {
    move $p_j$ from page_set into doc_cluster $C_j$
          where Sim ($p_i$,$S_j$) highest
  }
}
```

3.2 Delivering Indirect Pages to Clusters

Compared to DPs, IDPs provide less information about the target entity. Nevertheless, it does not mean that they are less important. Actually, the information extracted from IDP may be more novel and provide more valuable information to the users. In general, IDP could provide additional information such as the activity or experience of the target entity; and support or oppose the content in DP irrespective of whether they are consistent or not. Most importantly, IDP may provide critical or negative information that is not contained in the DP. For instance, a report of a company involving in a fraud may be ranked at the bottom of thousands of returning pages, but such pages may be significant to users in correctly evaluating the worthiness of the company. It can thus provide important information to evaluate the Target entities fairly and integrality.

We must therefore explore an approach to link DPs and IDPs properly. In other words, we want to add IDPs into the clusters anchored by the seeds (DPs). We make the assumption that clusters do not overlap and an IDP can be assigned to only one cluster. In addition, we drop pages whose cluster cannot be determined using similarity measures. This approach will contribute positively towards Precision figures at the expense of Recall.

As discussed earlier, we use the entities extracted from the original sources to calculate the distance between two pages. In topic locality assumption theory [8], pages connected by links are more likely to be about the same topic than those that are not. It is therefore reasonable to extend cluster along links via spreading activation or to perform probabilistic argumentation. We can also assume that pages sharing more entities, including links, URL and PnO NEs, should be grouped together. This is consistent with the intuition that the Target entities in two pages having same e-mail, birth date or birth place may have some intrinsic associations. Also, pages that link to the same root or each other may belong to the same target entity. So these evidences provide support for them to be grouped together.

In addition, the similarity between two entities is beyond the simple exact matching. For instance, “Francis Yeoh” is different from “Francis”, but their similarity is not zero because the latter is an informal expression (“short-form”) of the former. Conventional feature-based approaches are however infeasible for this task for various reasons. Firstly, the diversity of document types means we will not be able to pre-determine the vector space dimensionality a priori. Secondly, we are unable to estimate beforehand the feature counts such as named entities, links and anchors, would appear in a corpus. Moreover, the similarity between different features may not be zero (e.g. xxx.com and xxx.com/aaa). Thus we chose to use a different approach in page similarity resolution:

Let

$a_1, a_2, \ldots, a_m$ denote the features extracted from page a.

$b_1, b_2, \ldots, b_n$ denote the features extracted from page b.

and $S(a_i, b)$ denote the similarity between a_i and its most similar features in page b:

$$S(a_i, b) = Max\left\{S(a_{i,}b_1), S(a_i, b_2), ..., S(a_i, b_n)\right\} \quad (3)$$

Where we categorize into 3 distinct sets by our definition (defining non-overlapping sets simplifies the classification approach):

$$S(a_i, b_j) = \begin{cases} 1 & if\ a_i\ is\ subset\ of\ b_j \\ 0 & if\ no\ common\ terms\ are\ shared \\ x & if\ a_i\ is\ not\ proper\ subset\ of\ b_j, e.g. URL\ segments \end{cases} \quad (4)$$

The situation in URL and links are more complex and merits further explanation. If the roots of URLs are the same (such as www.xxx.com and www.xxx.com/aa), or components of URLs are similar (such as www.xxx.com and www.aaa. xxx.com), there should have a non-zero similarity. Let a_i and b_j be the respective number of segments of links i and j that is separated by dot or slash, and S_{ij}

be the number of identical segments among them. The similarity Sim(a,b) between a and b is calculated as:

$$\mathrm{Sim}(a_i, b_j)=S_{ij} / (S_i*S_j)^{1/2} \quad (5) \text{ which is equivalent to } x \text{ in equation (4)}$$

S(a, b) denotes the similarity from page a to page b, and S(b, a) denotes the similarity from page b to page a. S(a, b) is not equal to S(b, a) under general circumstances as they are asymmetrical.

$$\begin{aligned} S(a,b) &= \sum_{i=1}^{m} w_i \cdot S(a_i, b) \\ S(b,a) &= \sum_{i=1}^{n} w_i \cdot S(a, b_i) \qquad (6) \\ \overline{S(a,b)} &= \sqrt{S(a,b) \cdot S(b,a)} \end{aligned}$$

Here, $\overline{S(a,b)}$ is the Geometrical Average of S(a, b) and S(b, a), and w_i is the weight.

Finally, we derive the similarity between an indirect page i and seed j, *Sim(Page$_i$, Seed$_j$)*, by combining the similarities between PnO NEs (Equation 4), links and URLs (Equation 5), links. To achieve this, asymmetrical similarities between each IDP and a Seed is computed with suitable weights. This pair is then averaged geometrically to give a final figure. Different weights are configured for named entities, links and anchors in order to balance their effects on the importance of their roles in the Similarity matching processes.

We now outline the algorithm to select and link IDPs to a seed cluster.

Algorithm 3:

```
Arrange_indirect_page (page_set, cluster_set)
//clusters are represented by their seeds
{
  set unknown_set=null;  //collection of unknown pages
  for each ($page_i$ in page_set)
  {
    j = arg max sim($page_i$, $seed_i$)
    if ($j>\tau_2$)
      add $page_i$ into $cluster_j$;
    else
      add $page_i$ into unknown_set;
  }
}
```

where τ_2 is geometric similarity threshold for an indirect page to remain relevant to any existing cluster, otherwise it will be dropped into Irrelevant Page category.

3.3 Overall Procedure

Figure 4 shows the overall process of PnO searches and processing on the web. The user first submits a target entity name as the query to the system. The system then downloads the list of pages P_{all} related to the target. This step may involve other meta search engines. Second, a classifier is initiated to partition P_{all} into three groups: the set of DPs, S_{DP}, and the set of IDPs, S_{IDP}. Third, only distinctive pages about different Target entities in S_{DP} are used as seeds of the clusters. The other redundant pages in S_{DP} are moved to S_{IDP}. Fourth, each page p_i in S_{IDP} will be clustered to the closest cluster whose seed is the nearest to the current page. If p_i cannot be matched to a sufficiently similar seed, i.e. the similarity between them is less than τ_2, it will be discarded into an unknown set. Fifth, we use the name of organization (or person) that appears in the seed as the label to the corresponding cluster. The resulting set of clusters found is then presented to the users.

There are many ways through which we can improve user comprehension and acceptance of system usability. When user submits more constraints, for example, using the term “Virginia” to constrain the query “Francis Yeoh”, the system can utilize the constraint to rank the clusters so that the more relevant cluster appears at the top. Information in each cluster can also be extracted into a predefined template as concise summary to the users. It can also be presented as a set of navigable documents ranked first by the seed of each cluster, followed by the direct pages ascending in Direct Page similarities, and finally by the Indirect Pages.

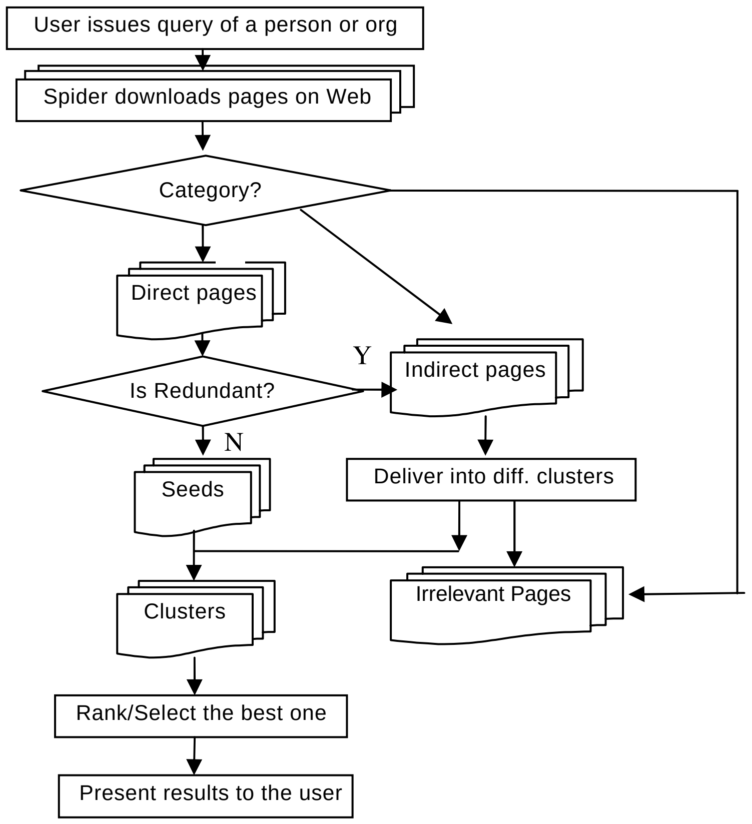

User issues query of a person or org
Spider downloads pages on Web
Category?
Direct pages
Indirect pages
Y
Is Redundant?
N
Seeds
Deliver into diff. clusters
Irrelevant Pages
Clusters
Rank/Select the best one
Present results to the user

Figure 4. The Process of a Web-based Information Extractor (Page Classifier)

4 Design and Implementation

This section outlines in detail the components that go into the PnOClassifier system. It also summarizes their functionality and the many considerations that have gone into building new components. Integration with reliable third-party, public domain tools and the processes of tuning or enhancing the tools for our pipelines are covered.

4.1 Systems Architecture

The PnOClassifier prototype system is engineered and developed as a cross-platform pipeline of crawlers, aggregators, classifiers and generators. Behind the scenes, database servers, middleware components and a host of other cutting-edge tools and libraries supported the pipeline with operations to scaffold the downloading, metadata excavation, feature extraction, named entity identification, decision-tree classification, and finally evaluation and profiling of the experimental results. Almost all of the components in the pipeline are statistically based.

All in all, there are a total of 15 major pipeline pit stops. First, a meta-crawler takes the user's query down the Internet to fetch relevant documents down to a local cache. While it's at it, the crawler also indexes the documents with relevant meta-data and checks for document type, ignoring all others except HTML. In addition, the crawler also checks for document completeness of downloaded items and converts them into plain text formats. An HTML validation engine then runs to convert the HTML files into XHTML files, conforming to that of a well-formed XML file. This step is necessary so we can rid ourselves of inconsistent, overlapping or missing tags otherwise tolerated by visualization tools such as the web browser. Once this process is completed, we can be sure the files are consistent and ready for additional tagging by our name entity engine.

At the same time, a URL analyzer runs to extract and index all types of HTTP, FTP, EMAIL links to and from the documents in the collection. This includes the ratio of incoming and outgoing links as well as the total occurrences of these URLs. At this point, the Name Entity analyzer goes to work by running against the documents one at a time to extract and tag into the files PERSONS, ORGANIZATIONS, DATES, MONEY, PHONES, and ADDRESSES. Following this, a well-formed consistency check is again performed on the transformed documents, after which an XPATH-based engine is fired to calculate token and entity figures. A metadata analyzer then runs to tidy up the metadata for these documents and reconciles the final ratios and statistical totals before going into the final step.

The last and final processing cycle involves classifiers and similarity engines. On a training cycle, a supervised classifier is executed for manual tagging and metadata generation. A decision-tree model is then generated as the output from this pass. On a test run, a default classifier generator is executed to perform preprocessing on the documents before running it against the decision-tree models generated by training pass. The results from the test run is parsed and assimilated into the corresponding document metadata. Finally, a similarity analyzer is executed to calculate the similarities between vectors of statistical features among the Direct Pages, Indirect Pages and in the process sift out more irrelevant pages. The output from this final pit stop is clusters of pages led by a Seed Page in each of them.

4.2 Design and Implementation Methodologies

The design and implementation of the PnOClassifier System architectures are built on quick turnaround prototyping methodologies resembling that of the original Spiral model [40]. Where appropriate in the development process, design patterns modeled after [41] [42] [43] [44] practical to the implementations are modeled to glue the variety of components together. One implementation is based on a client-server design, with ports connecting perpetual clients together in a daemon-mode chain. The alternative implementation is a loosely coupled pipeline of components. The different implementation paradigms was made so it is easy to insert a new component into the processing pipeline while having transient thread-safe operations on each and every client-server-based module without having to restart.

A uniform logger is also implemented so unattended and unsupervised operations can be carried out and activities tracked and captured for forensic investigation. Most of the components in the pipeline are based on the Java language, with Apache Jakarta Log4J [32] and Commons Logging [33] as the bridge between the console, log files, and remote loggers. The system currently runs on both Unix and Win32 platforms. On windows, the Gnu Utilities are deployed as a common set of local and web utilities among all the platforms. Environment variables are used as the initial bootstrap configuration dataset during the initialization of all components in the pipeline. Database handlers are derived from DBCP (Apache Jakarta's Database Connection Pooling) [34] away from the initial PoolMan [35] implementation. Backend database engine used is MySQL, with the abstraction and pooling layer based on DBCP and the PnOClassfier DatabaseAccess mechanism.

There are 2 types of storage available in the PnOClassfier. The first uses the native file system abstracted to store metadata and other forms of information about any downloaded web document. Filenames are generated based on the current timestamp and a humanized suffix using a dictionary to improve readability and navigability. Each type of information is stored in a file with the same filename but different extensions. All extensions and formats are configured and accessed via a shared Configuration module so components in the pipeline can import the module and adhere to the standards set down by the previous component in the line.

The second type of storage is independent of the file system and resides in an SQL database. The aforementioned cross-platform DBCP pooling mechanism is adapted to provide shared access via a common DatabaseAccess singleton offering functions to all modules in the pipe.

Apart from the storage mechanism, a standard bridge is also built to exploit functionalities already existing in standard utilities ported to various platforms. This includes the GNU utilities (on Win32), Lynx, and a dozen of other utilities in the same line. Threaded accesses to these functionalities are also implemented together with exception handling routines to arrest any runaways during unattended operations.

4.3 Supporting Resources

4.3.1 Test Collections

Training and test data are mandate in all Information Retrieval experiments and systems; the PnOClassfier System is no exception. Building on our previous efforts, current data sources consists of primarily 3 segments: commercial, academic, and our own collections.

Commercial offerings studied consist of both structured as well as unstructured documents and data. Among them, we selected Google because of its cross-platform API (the Google API) was among one of the most mature and open to multiple languages across different platforms [24]. The Google API allows up to 1,000 queries a day, but each query is limited to a certain number of retrieved documents. At the time of evaluation, the number fluctuates from 50 to 100, and affected our document collection efforts as we needed a count of some 1,000 documents for each query target entity, be it a person or an organization.

In line with TREC participations, we also outlined data experimentation strategies around the more updated WT10G collections. This was because the TREC Web Corpus (WT10G), built upon its predecessor, the WT2G collection, was a more substantial and higher quality data set that eliminates non-English and binary data documents. In addition, the 1.6 million sized collections also eliminate documents from “uninteresting” servers as well as redundant or duplicate data. This allow for full concentration on evaluating the pipeline against specific selections from the filtered collection for Persons and Organizations.

Last but not least, in an effort to bring our pipeline results closer to reality, we collected some 15,000 Web pages from Google on Persons and Organizations. This we christened our WebPnO Collection, and after post-processing and filtering, were made an important secondary training and test set (eg. Francis Yeoh, Sanjay Jain document and data sets).

Each Document in the collections outlined above consists of 3 main sets of data. The first is document metadata. This contains primarily server information, document title, number of links on the page, length of the page, and so on. The second set of data is based on information processed by our pipelines. This consists of text-based interpretation and extracted information such as the ratio of Named Entities, incoming URLs or outgoing URLs, query-to-title-relevance, and so on. Last but not least, the original document itself is definitely the most important part of the data set.

Among the collections, our initial testing and evaluation criteria focused more on the more authoritative Google API and TREC documents with emphasis on target set rules extraction. In the most recent and updated version of our system, we concentrated on bringing forth the system to more practical scenarios on the web, and gave more emphasis to our WebPnO collections.

4.3.2 GATE (General Architecture for Text Engineering)

GATE is an implemented architecture of components with a visual environment built to scaffold research and development work in language engineering. Within GATE, a document is represented by annotations and feature maps of name-value pairs. Processing Resources (PRs) are GATE components within the system that operates on these documents. Specifically, the ANNIE (A Nearly New Information Extractor) is modified for use in our PnO NE detection. The core of this research effort hinges on the accuracy and effectiveness of a Named Entity Detection system. Practically all of the features identified to be useful in segregating Direct Pages from the Indirect Pages and Irrelevant Pages depended on Named Entities. For instance, if the target named entity in question is “Francis Yeoh” and “francis”, “francis_yeoh” or any of the entities or their permutations appear partially or wholly in the URLs of query, the chances of the page being a Direct Page will be considerably increased. Conversely, if tokens of an entity other than the target are to be found in the URL of a page, the chances of it being Indirect Page containing derivative information about the target entity, or even an irrelevant page, will be much higher.

The GATE system’s class libraries are comparatively more difficult to adapt for use in a different pipeline system. The component-based ANNIE system [29], together with its set of PR components are coupled together in with modifications and embedded into our pipeline. Among them includes the following CREOLE (Collection of Reusable Objects for Language Engineering) resources:

- the English sentence splitter (gate.creole.splitter.SentenceSplitter)
- an input tokenizer that produces words (gate.creole.tokeniser.DefaultTokeniser)
- a POS tagger (gate.creole.POSTagger)

- a simple gazetteer of common terms (gate.creole.gazetteer.DefaultGazetteer)
- Coreferencer called the orthomatcher (gate.creole.orthomatcher.OrthoMatcher)
- entity transducers (gate.creole.ANNIETransducer)

GATE’s implementation is based on a large pool of past resources and experiences, and is effective in addressing general NLP tasks. However, the latest versions requires patching to its code. Among other problems, it hangs on various kinds of documents at various stages in its component system.The GATE-based Named Entity detection pipeline component we have incorporated thus far demonstrates that when properly planned and designed, a module that’s loosely coupled with the rest of the Information Extraction application can perform surprisingly inexpensive and good performance, and can be integrated with other modules in a pipeline execution model with minimal effort. The final question remains as whether there is a possibility that an integrated component completely dependent on one particular system such as the GATE architecture is more malleable than what we have come up with. However, the intrinsic value of such integration inevitably erodes with the complexity of the system and its learning curve, alongside with the many issues that we have to resolve to get the system up to deal with real world documents. For example, the parsers in both implementations are modified to detect non-ASCII characters and filter through them allowing us to focus on English documents. These characters include accents, umlauts, circumflexes and other possible non-standard lower and higher ASCII bytes.

4.3.3 OpenNLP

A named entity detector to work on sentence fragments, based on a Maximum Entrophy Model was derived from an Open Source Natural Language Processing component known as the OpenNLP. The original components and interfaces are created by Dr Jason Michael Baldridge, at the University of Edinburgh’s Institute for Communication and Collaborative Systems [25]. Components from the OpenNLP project consists of Natural Language Processing components useful for parsing and furthering work in syntactic and semantic fields of text processing. Of these, the OpenNLP Java Interfaces, Leo - the architecture for defining XML specifications of grammars for Natural Language parsing systems, MaxEnt – a Java-based package for training and using Maximum Entrophy Models, and finally, Grok – the collection of natural language processing tools based on the aforementioned are adapted for used. In short, Grok is a collection of NLP tools that provides a library of modules implementing the interfaces specified by OpenNLP.

The implementation was based on the following selected OpenNLP.Common interfaces from Grok’s “preprocess” packages:

- the sentence detector (sentdetect.EnglishSentenceDetectorME)
- a tokenizer (tokenize.EnglishTokenizerME)
- part of speech tagger (postag.EnglishPOSTaggerME)
- the variable Multi-Word Expression parser (mwe.EnglishVariableLexicalMWE)
- the English language category tagger (cattag.EnglishCatterME)
- a heavily modified version of the Named Entity detection modules (namefind.EnglishNameFinderME)
- a simple Email detector (namefind.EmailDetector)

For the OpenNLP version, version 0.51 was available over SourceForge at the time of implementation, and quickly became the open-source choice for our development effort. The Sheffield University's GATE program was then comparatively more complicated and documentation was scarce. In addition, it was a complete package tightly coupled with their visualization component meant for academic and research demonstrational purposes at that point in time.

Most of the development time was spent on patching the source code so it won't break on simple items like single quotes, and to significantly improve the accuracy which at that time was not too good (in particular, the EnglishTokenizer and EnglishNameFinder). As the processing time was tremendous, we packed the modified components into a pipeline and implemented a TCP-based client-server solution from which our clients can send information into processing threads and obtain output. Entity-based Processors were written in addition to the pipelines to detect different classes of Persons, Organizations, Addresses (Emails, Street Names, Building Names) and different kinds of cardinal digits (numbers). The training of the final tool requires large amounts of training data in specific domains for the Entrophy Models. We require a more re-targetable engine that can be adapted for different texts without having to extensively retrain the models. The OpenNLP-based implementation effort was later replaced in most situations for by the faster performing GATE [29] where complete texts are encountered. At this time of writing, the OpenNLP project has moved on to a more advanced realization of the Multi-Modal Combinatory Categorical Grammar formalism, christened the OpenCCG Project. It’s primary focus is now on Dialog Systems working on human speech and sentence fragments.

4.3.4 WEKA (The Waikato Environment for Knowledge Analysis)

Of the many automated classifiers (such as Naïve Bayes, NN), WEKA, a collection of machine learning algorithms was selected as a learning tool for our pipeline implementation [36] [37]. The C4.5 [21] implementation of WEKA 3 (http://www.cs.waikato.ac.nz/ml/weka/) known as the J48 was tuned to work with our similarity algorithms and results compared with others available (such as regression, Kstar, JRIP [36]). At the end of the day, we found that our adapted C4.5 approach gave us the best overall results in most cases.

A total of 3 components are implemented to achieve our objectives. The “Supervised Classifier”, an ANSI text-based tool with PnO NE tagging is created to support and scaffold manual class tagging. The “Dummy Classifier” prepares the system for unsupervised tagging, and the “Weka Generator” creates metadata prior to similarity analysis stage of the pipeline. All data formats are made to conform with the ARFF (Attribute-Relation File Format) specification which defines a data set in terms of one header list of attributes followed by relations with corresponding columns of values (question marks represents unknown values). The C4.5 algorithm was selected and adapted for our algorithms (1, 2, 3) because of its general retargettability and ability to cater for various circumstances [21]. Components created in this stage includes the WekaClassifier which is our primary workhorse for identifying DPs, the WekaAnalyzer that calculates the rest of the similarities against the seeds into the temporal databases, SimilarityAnalyzer which finally tags the indirect and irrelevant pages. The significant outputs from these implementation is presented Section 5.

4.3.5 Web Spider

The pipeline's first component is based largely on the Google API, with the capability to launch and monitor multiple threads with timeout and metadata collection on unlimited number of document retrievals. A Session Manager is implemented that creates and maintains the state for the pipeline for the duration of the retrieval in preparation for the remaining of the Named Entity processing. All pages other than HTML and plain text are ignored. Among the other reasons, primary consideration is the time required to parse and convert these documents, and the fact that seeds are very unlikely to be flashy PowerPoint, lengthy WinWord or PDF files.

The initial version of the crawler engine is based on the Compaq Web language, known as WebL at the time of initial web spider implementation. It is an imperative, interpreted language with built-in support for common protocols on the Internet, such as HTTP and FTP. It also supports data types like the common-place HTML, and XML. It was selected because its *service combinators* and *markup algebra* was useful in giving us a head start to building the first component in our pipeline system. We then realize the limitations of the WebL language quickly made it necessary for us to delve into its Java-based internals for tweaking. It was later determined that the scripting language will not meet with our requirements on functionality and performance tuning. In our case, parsing of incomplete or complex HTML breaks often, and downloading of documents cannot be made to invoke user-defined handling mechanisms or be threaded with more specific controls. Large amounts of data therefore cannot be downloaded in a streamlined manner.

The current implementation of the Grabber utilities are based on an interface derived from the Google API. The implementation code however is completely independent. Specific engine bindings can be implemented based on the search engine in mind, for instance, Altavista, Lycos, or Yahoo!. In addition, various sections of the interface have been designed so it can function as a component in a pipeline, and are not limited to that of a web crawler. Features include:

- Extensible Search Engine Interface
- Pipeline capable design
- Cross-platform configurable download limits, fetch sizes
- Configurable Threading and monitoring timeouts fetching
- Supports unlimited results fetching in batches from Google (unlike the Google API limits)
- Extensible input query optimization
- Blazing fast X-Path engine for data extraction (titles, etc)
- JavaCC, CyberNeko, and JTidy based HTML to XHTML Parser and Converter
- Humanized filename suffixes with timestamps via dictionaries
- File-type filtering and fetching
- Options for number of retries (or unlimited) on unreliable servers
- Configurable options for recursive fetch (down to N levels)

- Configurable for spanning servers (internal and external links) with follow options
- Configurable for Robots compliance
- Supports HTTP, HTTPS, FTP
- Option to save headers
- Configurable directory options
- Text extraction utility
- Links (HTTP, EMAIL, FTP, MAILTO, etc) Extractor
- Formatted HTML to Formatted Text Converter
- Visible and Invisible Links (img, cgi, mailto, etc) extractor
- Metadata extraction using above functionalities, as well as ratios (eg. query_title_ratios), and so on.

The Grabber is a very important tool because the documents it fetches and the metadata it constructs are the basis on which all other modules and components in the rest of the pipe operates upon. For this reason the variety of configurable options and threading support is built in with a high degree of reliability and robustness.

The preprocessed metadata and other information are used as inputs into the next component along the pipeline for named entity detection and structural analysis.

5 Experiments and Discussions

This section covers the empirical results of the experiments. Various aspects of the results are discussed alongside the variants in the input data and conclusions derived.

5.1 Selecting Test Samples from the Web

Experiment of web information processing is a time-consuming task, where each search typically returns hundreds, or even thousands of pages. Moreover, evaluating the effectiveness of clustering is notorious even though there are many guidelines to measure the quality of clustering such as the entropy measures, clustering error, and average precision [20]. Because of the general lack of standard authoritative test data sets for our specific task involving the clustering of web pages concerning Persons and Organizations on the World Wide Web, we have resorted to deriving a set of web pages for testing based on the following methodology:

a. In our experiments, we collected the names of 12 persons and 12 organizations (such as companies, governments and schools) from Yahoo (www.yahoo.com) and MSN (www.msn.com). In order to conduct meaningful tests, we removed PnOs that belong to large companies and famous persons (such as Microsoft or

George W. Bush). This is because there would be too many pages in the search results for such PnO names. For example, Google returns 2,880,000 pages for Microsoft, and the first hundreds of pages are about only one specific target. To ensure that there is sufficient data for the analysis, we also excluded those PnOs that return less than 30 pages (table 2).

b. We used every PnO name as the query string to Google. We downloaded the first 500 pages of each search, with the web spider filtering out files whose formats are not HTML and plain text (i.e. PDF, PS, PPT formats and DOC), and those whose lengths are less than 100 or more than 10,000 characters. The average number of validated text pages returned per PnO is about 421 (421.21).

c. We manually examined and tagged the returned pages to provide the ground truth for the tests. We determined the number of distinct Target entities for each query, and tagged all the DPs belonging to each target entity.

d. Further experimental results are cross-validated against previous test runs and results averaged.

Persons	**Pages**	**Organizations**	**Pages**
frank herbert	445	multisoft corporation	426
francis yeoh	402	innovision corporation	411
sanjay jain	423	yunnan agency	424
david beckham	411	suntec industries	423
mabel ong	431	famosa pte ltd	418
george bush	415	singapore university	432
catherine lim	429	singapore polytechnic	404
stanley ho	408	shaw corporation	419
stefanie sun	417	intuit enterprise	409
john doe	455	advantech	398
michael owens	442	indigo systems	428
harry lee	425	creative technologies	414
Total	5,103	**Total**	5,006

Table 2. List of persons and organizations used in the PnOClassifier experiments

The resulting set of web pages contains about 10,109 pages for 12 person and 12 organization names. We christened this set of web pages our **WebPnO** collection.

In order to compare our results with other reported systems for general web searches, we adopted the **WT10g** data set used in the homepage finding task of TREC-2001 evaluations. It consists of 10-gigabyte subset of the VLC2 collection and is designed to have a relatively high density of inter-server hyperlinks.

5.2 Testing using WebPnO Collection

We used a subset of the WebPnO collection to train and test our classifier for direct pages. For actual experiments, 90% of the pages are used for training, and the rest of the 10% for testing. Each sample is represented using 31 features, metadata of which are listed in Table 1, together with one decision class attribute (PAGE_CATEGORY). The current adaptive version of our WebPnO modified learning component is built based on the machine-learning algorithm C4.5 (http://www.cse.unsw.edu.au/~quinlan/) and WEKA 3 (http://www.cs.waikato.ac.nz/ml/weka/).

Training sets of 3 retrieval classes for persons are drawn from our WebPnO collection (Direct, Indirect and Irrelevant). The pages are then pre-parsed for meta-data extraction and categorized by hand with complete information including page category. These collections are then fed into our decision-tree engine with emphasis on cross-validation, where results obtained are averaged over 10 folds randomly selected from and partitioned within the WebPnO collection.

In order to provide insights into the roles of features and the set of rules extracted for finding DPs, we list **some** of the decision rules found as follows:

```
1)   URLS_COUNT <= 19 & PERSONS_COUNT <= 63 & NE_TOTAL >
  67 → Class DP
2)   NE_TOTAL > 4 & NE_TOKENS_RATIO > 0.06883 &
  NE_TOKENS_RATIO <= 0.22727 & WORD_COUNT <= 91 → Class
  DP
3)   ORGANIZATIONS_COUNT > 1 & NE > 14 & NE_TOTAL <= 67 &
  URL_SLASH_COUNT > 3 → Class IDP
4)   URLS_COUNT > 19 & URL_SLASH_COUNT > 3 → Class IDP
5)   NE_TOTAL <= 4  → Class IDP
6)   QUERY_TITLE_RATIO <= 0 & URL_LEN <= 50 & TOKENS >
  588 & URLS_IN_RATIO <= 0 & PERSONS > 2 & URL_LEN > 42 &
  TOKENS <= 1532: Class DP
7)   QUERY_TITLE_RATIO > 1 & PERSONS > 9: Class DP
```

where DP – Direct Page, IDP - Indirect Page, IRP - Irrelevant Page*

*** pages which are classified to be a DP or an IDP becomes an IRP.**

Here, Rule 1 implies that good DPs should have many PnO NEs but relatively few links and person names. Otherwise, they may be index pages or attendee lists. Rule 2 indicates that good DPs tend to be shorter, but contain a high percentage of PnO NEs. In general, they are home pages of persons. Rule 3 and Rule 4 show that IDPs have deeper URL depth. In addition, Rule 5 indicates that those pages that have fewer NEs must be IDPs. These two rules reveal that PnO NEs do play important roles in the classification of pages into DPs and IDPs. Rule 6 reflects one of the more complicated rules which is essentially a consolidation of the aforementioned (1 to 5); additionally, it also mentions that the Length of the URL should be generally short (somewhere between 42 to 50 characters), and that the number of tokens (excluding tags) should be constrained. Among others on Organizations, rules 7 also suggests that if the Person's tokens from the query is found in the title, that even if there are many person names on the page, it may well be a set of web pages describing a list of people, in detail, one on each page.

We used representative folds of 10 partitions from the person or organization categories to test the trained classifiers. We achieved an F_1 measure of about 87.77% (precision 88.26% and recall 87.27%). Our result is comparable to the best results reported for the homepage finding task (92%) in TREC-2001, a task which can be seen as a subset of our current classifier in the case where home pages are direct pages. We are encouraged by this result as we believe that DP detection is a more difficult task than homepage finding. This is because the latter deals only with a relatively simple task, where the decision depends mostly on URL length and whether the URL ends with a keyword or "/". Our experiment uses 8 of the URL based features from a total of 31.

5.3 Testing using WT10g Collection

In order to compare the performance of our system with others on similar tasks, we first compared the performance of our decision model with that reported in [9] on the homepage finding task. [9] performed the document analysis by employing decision tree and regression analysis using the feature set based mostly on URL depth, number of in- and out-links, and keywords. They tested on a subset of WT10g collection and reported a F_1 measure of 92%. We conducted similar test using our algorithm based on our original feature set "without tuning", where a larger balanced test set rather than the unbalanced set in [9] was used. We obtained a F_1 measure of about 91%, which is comparable to that reported in [9]. Although the results are not strictly comparable, the results do indicate that our technique is effective, even for the home page finding task which our system is not tuned to perform.

In our second test, we randomly selected about 50 DPs and 50 IDPs for organizations from the WT10g collection. We did not conduct a similar test for persons as there are very few (about 10) direct pages about persons. Our classification shows that we could achieve an F_1 measure of about 94%. This is higher than that achieved using on our larger WebPnO collection. The test demonstrates that our WebPnO collection is representative and demanding, and we could obtain better results from the random subset of the WT10g collection.

5.4 Our WebPnO Collection Clustering Results

We now discuss the full experiments on clustering web pages based on our WebPnO collection. We evaluated the performance of our clustering approach according to two aspects. First, we evaluate the quality of seeds. This involves the detection of candidate seeds from all direct pages derived from our experiments and the statistical discrepancies. Second, we evaluate the quality of the entire set of clusters. This is accomplished by measuring the average number of clusters formed through candidate seed detection and Indirect Page deliveries.

5.4.1 Direct Page Clustering Results

Table 3 gives the detailed performance of detecting seeds. As shown, the average ratio of missing clusters and redundant clusters (N_m / N) is lower than 10% (8.67% for Persons and 9.75% for Organizations). The number of missing clusters is represented by the number of DP Seeds undetected by the engine. This indicates that the seeds are stable and reliable. The number of seeds found varies from 4 to 10. From our experiments, we found that the number of seeds for a person tends to be higher than that for an organization, leading us towards the inclination that the number of persons with the same name is larger than that of organizations. This hypothesis can be concluded in subsequent experiments tabulating results from a larger corpus.

The quality of seeds is pivotal because it controls the distribution of segmentation. Missing a seed will mean the loss of a cluster and cause some IDPs to be assigned into wrong or unknown (Irrelevant Pages) set. On the other hand, if there are redundant seeds, IDPs about the same target may be delivered into different clusters, resulting in the need to perform non-trivial merging of the similar sets together. Fortunately, the results indicate that our technique is effective in differentiating between DPs and IDPs, and in removing redundant DPs. In addition, the implemented pipeline is able to perform multi-pass IRP filtering, thus further streamlining the cluster differentiation process.

Table 3. Direct Page Detection Performance using PnOClassfier Pipeline

Type	**N**	**N_c**	**N_i**	**N_m**	**Precision**	**Recall**	**F-Measure**
Person	196	168	28	17	85.71%	90.81%	88.26%
Organization	277	235	42	27	84.84%	89.69%	87.27%

Overall*	**520**	**458**	**25**	**30**	**85.28%**	**90.25%**	**87.77%**

* Overall N, Nc, Ni, Nm denotes arithmetic total for the sole purpose of calculating F-Measure only. N gives the number of candidate seed samples, N_C, N_I, N_M respectively denotes the number of correct, incorrect and missing DPs found. Recall = $N_c / N_c + N_m$, Precision = $N_c / N_c + N_i$. Results are averaged from runs over several queries under the same category. N_c counts include redundant seeds. Redundant seeds are DPs with Similarity of >= 0.95 <= 1.00

For practical real-world searches, the PnOClassfier pipeline performs within the aforementioned results. There is a general count of 3.84 Direct Pages (candidate seeds) for every 100 web documents found. Figure 5 shows the average direct page detection performance indicators. The averages are based on empirical results derived from the runs derived from our WebPnO Collection of 12 persons and 12 organizations. The overall trend indicates an encouraging F-Measure of some 87.77%, with Precision and Recall at 88.26% and 87.27% respectively. From derivative experiements, we also observed that there is a drop in Precision and Recall values across the board as the sample size, or number of web documents increases. This is most evident among web pages on Organizations, where duplicate documents with almost identical contents are found from different publications or web portals covering a particular event. It is also possible that these articles are written by journalists on a particular company who has them published on multiple sites. Our experiments shows that the features such as MONEY, PERCENT, and other specific

RATIOS in the empirical test suites contributed towards the detection of seeds pertaining to Organizations. This is consistent with intuition that economic figures are more relevant to some types of firms than to some persons.

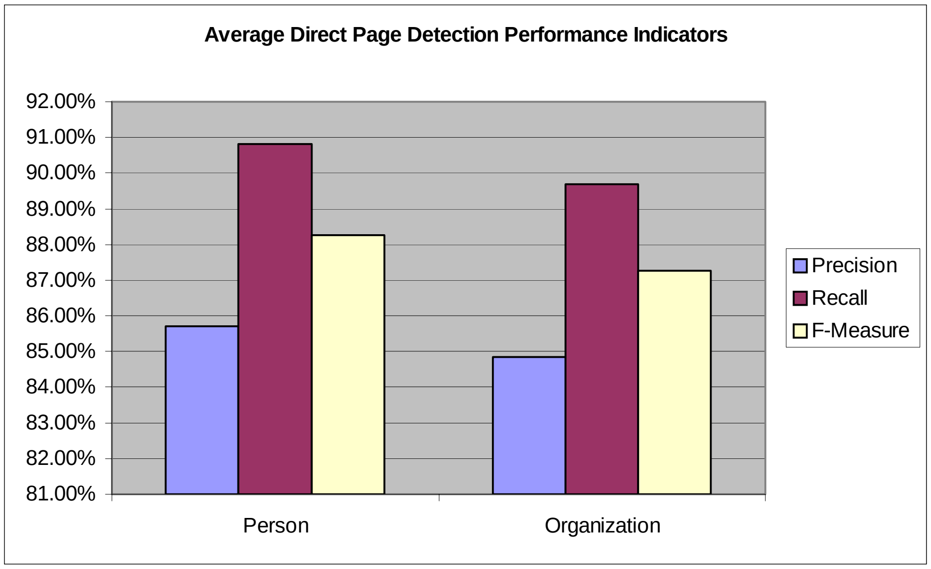
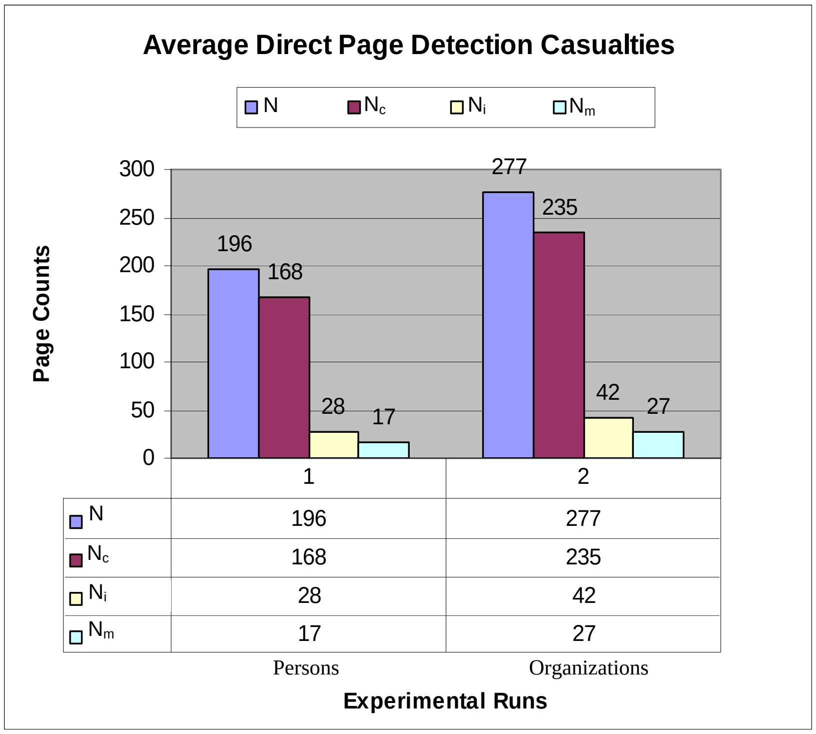


Figure 5. Average Direct Page Detection Performance Indicators

The empirical results show that our techniques and pipeline implementation is effective in addressing the correct detection of Direct Pages and Candidate Seeds. We can see that the number of correctly identified Direct Pages increases proportionately with an increase in the number of total web documents in each sample sets (Figure 6), maintaining an average Precision of 85.28%.

The scores on the incorrectly classified pages deserve some attention (Table 3). In particular, it is observed that there is a steady increase of incorrectly identified pages as the sample size N increases. This can be attributed to an expected increase in the “noise” of the samples when we increase the set of test or training documents, and averages out to be increasing less than proportionately in relation to N. The number of redundant direct pages however, appears to increase more than proportionately relative to the total sample pages in a set. As with the case of organizations, duplicates tend to be published in part or wholly on different web portals, newspapers and other aggregator sites.



	1	2
N	196	277
N_c	168	235
N_i	28	42
N_m	17	27

Figure 6. Average Direct Page Detection Casualties for Incorrect, Missing Candidate Seeds, Direct Pages

Given a sample size of 200 pages, we have a yield of 26 direct pages of clustered document delivery hits with bias set to 10 (meaning, at least 10 baseline entity matches for either NE (Per, Org, Loc (Address, Email)). If we release the bias, all pages will be classified (unless they're irrelevant), giving high recall, but much lower precision. This is because when the lowest matching pages are used for direct page seed clusters may not be accurate. Increasing the bias drops some of the more statistically ambiguous samples (Table 5), reducing recall and increasing precision of page delivery ratios. The prototype engine works well on small collections with the trained data (Table 4).

Table 4. Direct Page Detection for small sample size of 200 pages

Type	**N**	**Nc**	**Ni**	**Nm**	**Precision**	**Recall**	**F-Measure**
Person	8	5	3	1	62.50%	83.33%	72.92%
Organization	18	15	3	2	83.33%	88.24%	85.79%
Overall*	**26**	**20**	**5**	**3**	**72.92%**	**85.79%**	**79.36%**

5.4.2 Indirect Page Clustering Results and Irrelevant Pages

Secondly, we clustered IDPs into each cluster using Algorithm 3. We found that the average number of IDPs in clusters about organizations is considerably higher than that about persons. The number of redundant pages (pages with Similarity >= 0.95) are also higher for organizations compared to persons. The quality for the entire set of clusters is evaluated as follows. Table 5 lists the performance of assigning IDPs to clusters. The Table shows that we could deliver over 50% of IDPs to the clusters (53.59% for Persons and 46.87% for Organizations). The rest of less than 50% of pages are placed in the unknown and irrelevant page set. As there are no comparable results available on our specific task, it is hard to compare our results in comparison to other reported systems. However, the results reported in [15] showed that the state-of-the-art clustering methods could achieve a performance of between 59% and 87% in F1-measure on a range of test corpuses. This places the performance of our system near the top range, suggesting that tuning can bring us up towards the top end of the performance scale. Despite the lack of comparative experiments in this specific area, our results strongly suggest that the PnOClassifier approach is effective and reliable on practical web tasks.

Our analysis on the pages assigned to the unknown set shows that they tend to be dispersed pages that lack evidence for their assignments. This may be caused by missing heuristic information, when some target entities do not have DPs or the contents related to target entities are only expressed indirectly. We conducted another experiment in clustering IDPs without using the NE features. We found that the F_1-measure decreased by nearly 15%. The results again show that the NE features are important for this task.

The performance of the pipeline features reaches up to an average of 46.87% for Organization-based IDP delivery (Table 5). There is a considerable increase in the number of irrelevant pages detected. This contributes to higher quality Indirect Page delivery ratio. The stricter criteria set by the increase in the high number of feature set and an effective final IDP similarity threshold adjustment factor τ_1 and τ_2 in Algorithms 1 and 3 ("Bias") is an additional positive contribution factor. Figure 7 presents this information (Table 5) visually.

Table 5. The performance of assigning IDPs

IDP page	**N_{Total}**	**$N_{Indirect}$**	**$N_{Delivered}$**	**$N_{Irrelevant}$**	**Delivery**	**Bias**
Person	5,103	2,644	1,417	3,518	53.59%	10
Organization	5,006	2,010	942	3,829	46.87%	50
Overall	5,054.5	2,327	1,179.5	3,673.5	50.23%	-

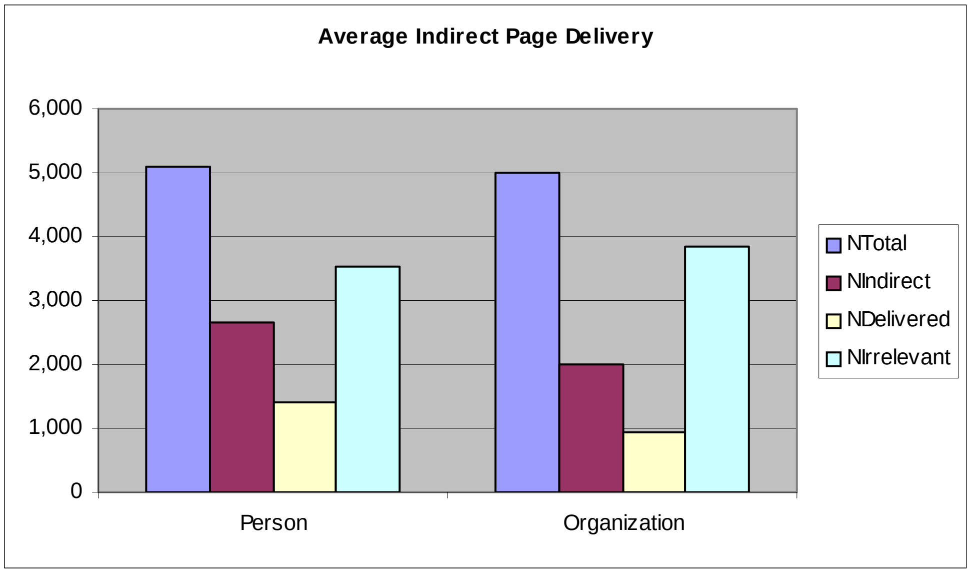


Figure 7. Average Indirect Page Delivery Performance for classifying IDP correctly.

The quality of Indirect Pages that lie at the boundaries of the unknown sets (or Irrelevant Pages) is not high. That is to say, they are not likely to actually shift from normal clusters to being an irrelevant page if we actually do tune these parameters, simply because the overlapping features are very low. They are most relevantly spurious or illegitimate Indirect Pages in many cases we have studied and experimented with. Therefore, whether or not these pages are included does not actually affect to any significant extent the information we really extract from the corresponding cluster. In fact much of the extracted information comes from the higher-quality pages such as the Direct Pages and candidate Seeds sharing highest similarity counts relative to the actual Seeds, which in turn are the ones that really represents the target entities detailed in the original query.

Under real world conditions, the “Bias” parameter settings for in practical applications are dependent on the users’ preferences. Increase the ratio of delivery to detected clusters will reduce missing clusters/pages, thereby increasing Recall. If they need higher Precision, we can choose to drop the deliveries’ performance thus ignoring lower-quality pages.

The extensiveness of these experiments took a considerable amount of time to complete under pipeline components. Many repeated sets of experiments are conducted to ensure coherence and consistency of the generated clusters and the statistics derived. On to performance, it is estimated that the speed of pipeline processing can be considerably reduced by about 75% through proper indexing and PnO Named Entities detection on pre-compiled pages on domain-specific Collections.

A noteworthy observation on the empirical clustering experiments shows that search results deemed indecipherable eventually by human judges through manual partitioning within a result set also expectedly drew ambiguous conclusions from the PnOClassifier.

6 Conclusions and Future Work

PnO is one of the common types of queries posed by users when surfing the Internet. The problems with normal web search engines are that they return too many irrelevant pages and are unable to distinguish between different entities having the same name. An effective PnO crawler and aggregator for the web is one of the first efforts in clustering different target entities of the same name correctly and naturally. The clustering algorithm described and implemented herein uses Named Entities as the basis for most of its statistical functions. In short, PnO NEs are the key to the entire picture.

Empirical results on the actual web using the names of approximately 12 persons and 12 organizations show that the current algorithmic and implementation method is effective for practical PnO retrieval. It has achieved an F1 measure of 87.77% for finding the cluster seeds, which are direct pages of distinct target entities expressed in the query. The embedded techniques can also practically assign over 50% of indirect pages to the clusters (Table 5).

This approach is regarded as being reader-centric and thus more naturally acceptable with higher comprehension rates than that of machine-generated folder names. It provides an effective way for users to summarize information pertaining to specific targets. It also provides tracking capabilities over various feature aspects of these PnOs as we relied on a fresh method of weighed sum reconciliation similar but different to the features in Xi [45] and Crofts [46] and more directed at the task at hand. These included query-related keywords and duplicate detection, addressing the fact firstly, different target queries may use the same feature-vocabulary to represent their content-category, for instance, “education”, “working experience”, “address”, “phone number”, and secondly, the pages presenting the target query in question may be using quite different terms in many aspects, which lead us to the fact that they will be divided into different clusters when standard approaches are employed, thus creating a new requirement for further reconciliation/merging of these otherwise related clusters.

The current prototype implementation works on full-featured web pages. Snippets and fragments of text, on the other hand, could prove to be a more efficient form of source information. The baseline algorithms however are Named Entity based. The use of these fragments may give rise to a myriad of “fake” PnO NEs, due in large part to the arbitrary manner in which many colloquial statements and truncated sentences are interpreted. The clustering quality may therefore be aversely impacted. Further research in this area is needed before any conclusive directions can be identified.

Calculations and detections of Named Entities is currently the most time-consuming task next to web page fetching from the web. It will be nice to have an index of entities on a specific domain catered for our clustering purposes. Pre-indexed databases, abstracts and summaries can all contribute positively towards the clustering pipeline.

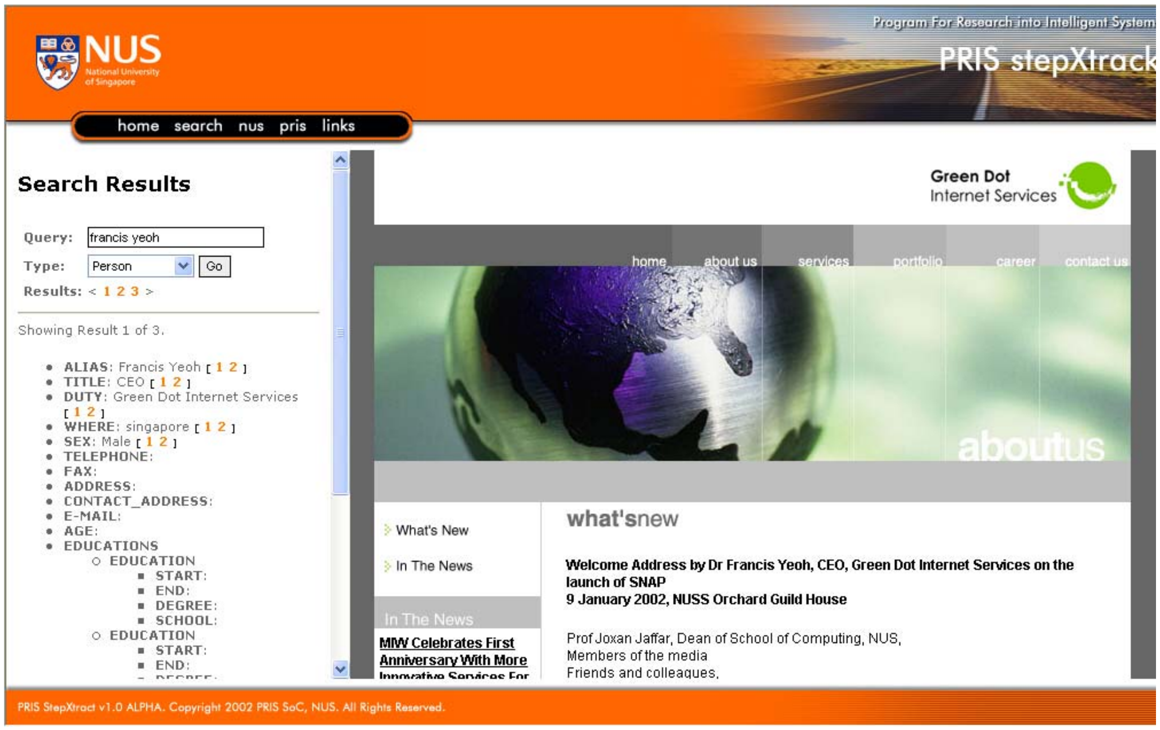
We believe the initial query data sets can be considerably improved by using various query optimization techniques which are not incorporated at this moment by our system. This will guarantee us more accurate search engine results to begin with. For instance, constraints such as “National University of Singapore” will lead us to more pages about the person, if the query user knows that the person is currently employed under the University. From this lead, we can be reasonably assured that the search engine results are ranked according to these criteria, with possibly more candidate Seeds and Direct Pages upfront nearer to the top of the returned results and Indirect Pages thereafter.

Future research can also be carried out as follows. Firstly, the existing classifier algorithms can be improved especially in areas of real-time performance on large document sets. In the aforementioned sections, I touched on the necessity for a search engine that can wholly or partially serve NE-based meta-data to our classifiers. This very important and critical component relieves the pipeline from having to crawl and download pages from different engines. In specific time-critical deployment scenarios, such an engine also proved to be the key in moving towards a real-time implementation.

Secondly, information extraction can be performed using template-based algorithms on the clustering results (namely, the DPs and IDPs) and the aggregated information presented to end-users to afford an at-one-glance view of the data. This area of research will involve competent word-sense disambiguation and co-referencing. This is a separate research area on its own and can be taken in a different direction.

Thirdly, we plan to extend our techniques to organize and extract information in other domains such as research documents in specific areas of expertise. More research on the effective set of features for other domains needs to be carried out. The expected enhancements to the current PnOClassfier system includes bias threshold adjustments as well as a more elaborate similarity algorithm to bring the Indirect Pages up to an even higher catch-all rate. Multi-pass functions are expected to be required for these enhancement stages.

Finally, an ideal prototype application for a proof-of-concept showcase to this effort is depicted in the Figure 8, with an information template carrying the extracted data neatly displayed with the relevant source links and a browser window to preview the document source. The generated information aggregated from documents clustered with the PnOClassifier system will be closer to and more effective in addressing reader-oriented information extraction requirements.


NUS
National University of Singapore
Program For Research into Intelligent Systems
PRIS stepXtrack
home search nus pris links

Search Results

Query: francis yeoh
Type: Person Go
Results: < 1 2 3 >

Showing Result 1 of 3.

- ALIAS: Francis Yeoh [1 2]
- TITLE: CEO [1 2]
- DUTY: Green Dot Internet Services [1 2]
- WHERE: singapore [1 2]
- SEX: Male [1 2]
- TELEPHONE:
- FAX:
- ADDRESS:
- CONTACT_ADDRESS:
- E-MAIL:
- AGE:
- EDUCATIONS
  - EDUCATION
    - START:
    - END:
    - DEGREE:
    - SCHOOL:
  - EDUCATION
    - START:
    - END:

Green Dot
Internet Services
home about us services portfolio career contact us
aboutus
What's New
In The News
In The News
what'snew

MIW Celebrates First Anniversary With More Innovative Services For

Welcome Address by Dr Francis Yeoh, CEO, Green Dot Internet Services on the launch of SNAP
9 January 2002, NUSS Orchard Guild House

Prof Joxan Jaffar, Dean of School of Computing, NUS,
Members of the media
Friends and colleagues,

PRIS StepXtract v1.0 ALPHA. Copyright 2002 PRIS SoC, NUS. All Rights Reserved.

Figure 8. Template-based Prototype Interface for next-generation PnOClassfier System

7 References

[1] G. Salton, Automatic Text Processing. Addison-Wesley, New York, (1989)

[2] Text REtrieval Conference (TREC) Home Page, http://trec.nist.gov/

[3] Ellen M. Voorhees, Donna Harman, Overview of TREC 2001, NIST, TREC 2001, pp1-15

[4] N. Craswell, D Hawking, Overview of the TREC-2002 Web Track, TREC 2002, pp1-16

[5] D. Hawking and N. Craswell, Overview of the TREC-2001 Web Track, TREC 2001, pp61-67

[6] T. Westerveld, Wessel Kraaij, and Djoerd Hiemstra, Retrieving Web pages using Content, Links, URLs and Anchors, TREC 2001, pp663-672

[7] N. Craswell, D. Hawking, S. Robertson, Effective Site Finding using Link Anchor Information; SIGIR, 2001

[8] W. Kraaij, T. Westerveld, D. Hiemstra, The Importance of Prior Probabilities for Entry Page Search, SIGIR2002

[9] W. Xi, E. A. Fox, Machine Learning Approach for Home Page Finding Task, TREC 2001, pp686-697

[10] Min Zhang, et al, THU at TREC2002: Novelty, Web and Filtering, TREC 2002, pp29-42

[11] MacFarlane, A. MacFarlane, Pliers at TREC 2002, page 311, TREC 2002, pp311-313

[12] Pirolli, P. & Card, S. Information foraging in information access environments. Proc. of Conf. on Human Factors in Computing Sys., 51-58, 1995

[13] Daniel Boley, et al, Partitioning-based Clustering for Web Document Categorization, in: Decision Support System 27, 329-341, 1999

[14] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In: Proc. ACM SIGIR'98, pp46-54, 1998

[15] O. Zamir, O., Etzioni, Grouper: a dynamic clustering interface to Web search results, in Computer Networks, 31(11), pp1361-1374, 1999

[16] M. Steinbach, G. Karypis, and V. Kumar, A comparison of document clustering techniques. Text Mining Workshop, KDD, 2000.

[17] G. Salton, M. J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, NY, 1983

[18] Mark Craven, Dan DiPasquo, et al, Learning to Extract Symbolic Knowledge from the WWW, Proc. of AAAI-98, Madison, USA, pp509-516, 1998

[19] K. Yang, Combining Text-, Link-, and Classification-based Retrieval Methods to Enhance Information Discovery on the Web, Ph D. thesis, UNC-CH., 2002

[20] J. Picard, J. Savoy, Using Probabilistic Argumentation Systems to Search and Classify Web Sites, 24(3), pp33-41, IEEE Data Engineering Bulletin,2001

[21] Quinlan, J. R. Learning decision tree classifiers. ACM Computing Surveys, 28(1):71-72, 1996

[22] A.K. Jain, M. N. Murty, and P.J. Flynn, Data clustering: A review, ACM Computing Surveys, 31(3), pp264-323, 1999

[23] TREC Web Corpus, WT10g:
http://www.ted.cmis.csiro.au/TRECWeb/wt10g.html

[24] The Google Web API: http://www.google.com/apis/

[25] Jason Michael Baldridge's OpenNLP Project:
http://www.iccs.informatics.ed.ac.uk/~jmb/

[26] The OpenNLP Organizational Center: http://opennlp.sourceforge.net/

[27] The OpenNLP Maximum Entropy Effort:
http://maxent.sourceforge.net/about.html

[28] The OpenCCG Dialog Parsing Realizer System: http://openccg.sourceforge.net/

[29] Hamish Cunningham, Diana Maynard, et. Al.: GATE: an Architecture for Development of Robust HLT Applications

[30] WiseNut Multi-Level Categorization Engine:
www.wisenut.com/pdf/WISEnutWhitePaper.pdf

[31] S. M. Ruger and S. E. Gauch: Feature reduction for document clustering and classification, Imperial Colledge of London, Technical Reports, 2000

[32] The Apache Jakarta Project's Log4J:
http://jakarta.apache.org/log4j/docs/index.html

[33] The Apache Jakarta Project's Commons Logger:
http://jakarta.apache.org/commons/logging.html

[34] The Apache Jakarta Project's DBCP: http://jakarta.apache.org/commons/dbcp

[35] PoolMan v2.0: http://sourceforge.net/projects/poolman/

[36] WEKA: The waikato environment for knowledge analysis: Stephen R. Garner, Department of Computer Science, University of Waikato, Hamilton.

[37] Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations: Ian H. Witten, Eibe Frank, Morgan Kaufmann, October 1999

[38] A Comparison of Document Clustering Techniques: Michael Steinbach, et. Al. Department of Computer Science and Engineering, University of Minnesota; Technical Report #00-034

[39] Design and Implementation of a High-Performance Distributed Web Crawler: Vladislav Shkapenyuk, Torsten Suel; CIS Department, Polytechnic University, Brooklyn, NY 11201.

[40] B. Boehm. A spiral model of software development and enhancement. IEEE Computer, 21(5):61-72, 1988.

[41] Design Patterns: Elements of Reusable Object-Oriented Software; Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides; October 1994, Addison-Wesley ISBN 0-201-63361-2.

[42] Antipatterns: Refactoring Software, Architectures, and Projects in Crisis; W.J.Brown,R.C.Malveau,W.H.Brown,H.W.IIIMcCormick,andT.J.Mowbray; John Wiley & Sons, 1998

[43] The Pattern Depot: A Repository of Practical Software Patterns; www.patterndepot.com

[44] Applied Java Patterns: Stephen A. Stelting (Author), Olav Maassen (Author); Pearson Higher Education; 1st edition (December 31, 2001), ISBN: 0130935387

[45] Jack G. Conrad , Xi S. Guo , Cindy P. Schriber, Online duplicate document detection: signature reliability in a dynamic retrieval environment, Proceedings of the twelfth international conference on Information and knowledge management, November 03-08, 2003, New Orleans, LA, USA

[46] S. Cronen-Townsend, Y. Zhou, and W.B. Croft, "Predicting Query Performance", in the Proceedings of ACM SIGIR 2002, 299-306, 2002.

Appendix A: TREC Web Corpus : WT10g

http://www.ted.cmis.csiro.au/TRECWeb/wt10g.html

Note (2002-10-11): The following description of WT10g was last updated in March 2000. To obtain WT10g and/or the more recent .GOV test collection, see our access to data page.

Goals in the preparation of WT10g

There were a number of goals in the preparation of WT10g. These included:

- A more substantial quantity of Web data than was available in WT2g.
- A higher "quality" of Web data than is present in either WT2g or VLC2. This meant trying to eliminate non-English and binary data documents. (Foreign language documents are not uninteresting, but retrieval over mixed language collections is currently served by the cross-language track in TREC and the new cross-language workshop.) It also meant trying to eliminate "uninteresting" servers and/or documents.
- Elimination of large quantities of redundant or duplicate data.
- A larger number of inter-server links than was present in WT2g.
- Better support for distributed information retrieval experiments.
- Preservation of certain statistical properties from the VLC2, such as server size distribution.

Properties of WT10g

- 1 692 096 documents
- 11 680 servers
- an average of 144 documents per server
- a minimum of 5 documents per server
- 171 740 inter-server links (within the collection)
- 9977 servers with inter-server in-links (within the collection)
- 8999 servers with inter-server out-links (within the collection)
- 1 295 841 documents with out-links (within the collection)
- 1 532 012 documents with in-links (within the collection)

Appendix B: Typical Document Metadata File

Sample WebPnO Web Page Document Metadata file

```
#Wed Oct 12 07:17:16 SGT 2003
money.count=0
urls.in=0
timestamp=1068401251199
emails.count=0
phones.count=0
persons.count=24
url=http\://www.comp.nus.edu.sg/~leews/learning.html
query=sanjay jain
percentages.count=0
persons.ne.ratio=0.8
urls.out.ratio=0.0
query.url.ratio=0.0
urls.out=0
tokens.total=1097
ftp.count=0
url.base=http\://www.comp.nus.edu.sg
ne.tokens.ratio=0.03
organizations.ne.ratio=0.2
page.category=irrelevant
organizations.count=6
url.slash.count=2
weka.id=1
http.count=6
urls.in.ratio=0.0
ne.total=30
ftp.urls.ratio=0.0
url.length=47
urls.count=6
date=Mon Nov 10 02\:07\:31 SGT 2003
http.urls.ratio=1.0
query.title.ratio=0.0
dates.count=9
title=
```

Appendix C: Typical Classifier Decision Tree Result

Typical Sample Run of C4.5 (WEKA J48) Adaptive WebPnO Modified Classifier Pruned Tree Results over approx 250+ random samples with cross-validation partitions set to 10. The algorithm prunes the final trees produced using “subtree-raising” [21] which in turn increases general retargetability of the decision trees.

```
query_title_ratio <= 1
|   http <= 32
|   |   query_title_ratio <= 0
|   |   |   tokens <= 113: irrelevant (17.0/1.0)
|   |   |   tokens > 113
|   |   |   |   url_len <= 38: irrelevant (5.0/1.0)
|   |   |   |   url_len > 38
|   |   |   |   |   url_slashes <= 3
|   |   |   |   |   |   org_ratio <= 0.36
|   |   |   |   |   |   |   persons <= 17: indirect (9.65/3.24)
|   |   |   |   |   |   |   persons > 17: irrelevant (9.65/3.41)
|   |   |   |   |   |   org_ratio > 0.36: indirect (62.71/16.53)
|   |   |   |   |   url_slashes > 3: irrelevant (30.0/10.0)
|   |   query_title_ratio > 0: irrelevant (9.0)
|   http > 32
|   |   http <= 37
|   |   |   tokens <= 495: indirect (3.0/1.0)
|   |   |   tokens > 495: direct (29.0/1.0)
|   |   http > 37
|   |   |   tokens <= 779: irrelevant (13.0/1.0)
|   |   |   tokens > 779
|   |   |   |   tokens <= 1892: indirect (12.0/2.0)
|   |   |   |   tokens > 1892: irrelevant (7.0)
query_title_ratio > 1
|   persons <= 9: indirect (2.0/1.0)
|   persons > 9: direct (30.0)

Number of Leaves  :   14

Size of the tree :    27


Time taken to build model: 0.15 seconds
Time taken to test model on training data: 0.01 seconds

=== Error on training data ===

Correctly Classified Instances        199               83.2636 %
Incorrectly Classified Instances       40               16.7364 %
Kappa statistic                         0.7456
K&B Relative Info Score             16366.8055 %
K&B Information Score                 255.406  bits      1.0686 bits/instance
Class complexity | order 0            372.8138 bits      1.5599 bits/instance
Class complexity | scheme             131.0766 bits      0.5484 bits/instance
Complexity improvement     (Sf)       241.7371 bits      1.0115 bits/instance
Mean absolute error                     0.1667
Root mean squared error                 0.285
Relative absolute error                38.1617 %
Root relative squared error            60.9892 %
Total Number of Instances             239


=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
0.951     0.006      0.983     0.951     0.967    direct
0.852     0.165      0.726     0.852     0.784    indirect
0.742     0.092      0.847     0.742     0.791    irrelevant


=== Confusion Matrix ===

a  b  c   <-- classified as
58  2  1 |  a = direct
0 69 12 |  b = indirect
1 24 72 |  c = irrelevant



=== Stratified cross-validation ===

Correctly Classified Instances        166               69.4561 %
Incorrectly Classified Instances       73               30.5439 %
Kappa statistic                         0.5382
K&B Relative Info Score             13138.6461 %
K&B Information Score                 204.9549 bits      0.8576 bits/instance
Class complexity | order 0            372.8872 bits      1.5602 bits/instance
Class complexity | scheme           14120.4635 bits     59.0814 bits/instance
Complexity improvement     (Sf)     -13747.5764 bits    -57.5212 bits/instance
Mean absolute error                 0.2282
Root mean squared error               0.3739
Relative absolute error            52.2211 %
Root relative squared error          79.9988 %
Total Number of Instances            239


=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
0.967     0.028      0.922     0.967     0.944    direct
0.654     0.272      0.552     0.654     0.599    indirect
0.557     0.176      0.684     0.557     0.614    irrelevant


=== Confusion Matrix ===

a  b  c   <-- classified as
59  2  0 |  a = direct
3 53 25 |  b = indirect
2 41 54 |  c = irrelevant
```