Automated Discovery of Search Interfaces on the Web

Jared Cope
Australian National University
Jared.Cope@accc.gov.au

Nick Craswell and David Hawking
CSIRO Mathematical and Information Sciences
nick.craswell@csiro.au, david.hawking@csiro.au

Abstract

Web search engines work well for finding crawlable pages, but not for finding datasets hidden behind Web search forms. We describe a novel technique for detecting search forms, which could be the basis for a next-generation distributed search application. We use automatic feature generation to describe candidate forms and C4.5 decision trees to classify them. In two testbeds, we get an accuracy of more than 85% and a precision of more than 87%. One of our decision trees is effective on both testbeds, suggesting that it is a useful general-purpose tree.

Keywords: World Wide Web, distributed information retrieval, machine learning

1 Introduction

Many useful datasets on the Web are uncrawlable, so cannot be searched using conventional search engine technology. However, using techniques of distributed information retrieval, such datasets can be located and accessed automatically.

For example, to include the Oxford English Dictionary (`http://www.oed.com/`), a search engine would have to crawl it: a process of recursively downloading pages and following their hyperlinks. However, the OED site is uncrawlable for a number of reasons. It is a subscription service, so both the engine's crawler and end users would need subscriptions. Its pages are not reachable by following hyperlinks, so are inaccessible to crawlers (users access pages via a search form). Finally, the site forbids engines from crawling its main content areas, via the standard for robots exclusion (`robots.txt`).

Even if the pages were public, hyperlinked and non-excluded, there are problems with comprehensively crawling such a site. It has more than 615 164 words, 2 436 600 quotations, 139 900 pronunciations and 219 800 etymologies. If each of these appeared on one page, a crawl would generate noticeable network traffic and server load. However, in the current OED site each definition appears on many dynamically generated pages (alone, with etymologies, with quotations, with pronunciations, with both etymologies and quotations, and so on). Efficiently crawling the site would require sophisticated duplicate elimination methods. Even then, the crawled search might not be better than the original OED site's. The OED search is integrated with the rest of the site and includes "advanced search" capabilities which would be hard to emulate in a crawled system.

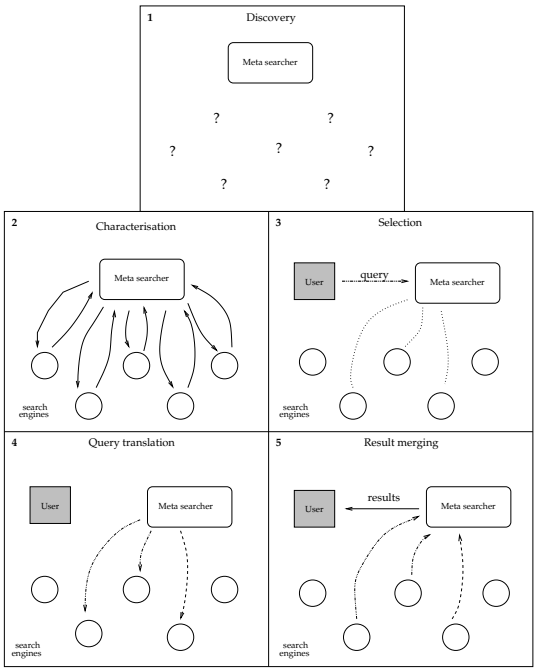

Figure 1: Five problems of distributed information retrieval.

The trend on the Web is for more Web sites to have database back ends, content management systems, dynamic content, multiple views of the same content and access restrictions. Each of these factors makes it more difficult to do a comprehensive crawl, and makes local search engines relatively more important.

Methods for distributed information retrieval can use whatever search interfaces are available to provide a unified search experience for the user. Rather than crawling, they assist users by selecting the right search interfaces, querying them and presenting their results in an integrated list.

This paper considers the most basic problem of distributed information retrieval, in the context of the Web. It considers how a distributed information retrieval system can identify the Web search systems over which it will operate, with candidates including the OED search interface and many thousands of others.

2 Motivation — Distributed information retrieval

Most work in distributed information retrieval has considered four main problems: characterization, selection, query translation and result merging. Before

it is queried, the meta search system characterizes the available search interfaces. Then given a query, it selects a subset of useful search interfaces, queries them and presents their results to the user.

This paper considers an overlooked problem that precedes the other four, discovery of search interfaces (Figure 1). The meta search system must first discover a set of search interfaces or be provided with such a set, before it can proceed with the other four steps.

Very little work has been done in this area. **Perkowitz, Doorenbos, Etzioni & Weld (1997)** investigated the problem of automatically learning to interact with information resources on the Internet. An agent was developed for finding shopping search forms, based on simple heuristics for discarding those which are clearly something else. The current paper also classifies forms, but with a more sophisticated approach. **Dreilinger & Howe (1997)** studied selection rather than discovery, but their discussion of future work foreshadows our current work. In particular, they suggest that an agent acting on auto pilot roaming the Web for new search resources could lead to a new search paradigm, especially if the newly discovered resources are incorporated into a meta searcher.

Once a search interface has been discovered, the meta searcher may find out what sort of information is available from the interface. This characterization problem is illustrated in box 2 of figure 1.

Gravano, Chang, García-Molina & Paepcke (1997) proposed a standard, *STARTS*, to aid meta searchers in choosing the best sources to query. The protocol requires search interfaces to export information about themselves in a standard format such as the list of words contained in the documents indexed. However in practice, such a protocol has limitations on the Web because there is no authority to make sure that all search interfaces cooperate and implement the protocol. Presently there are very few *STARTS* compliant engines. **Callan & Connell (2001)** presented query based sampling, which involves querying search interfaces, downloading some results documents and building a resource description based on those documents. Resource descriptions built from probe queries are more widely available, because they are available even if search interfaces are not explicitly cooperating with the meta searcher. **Ipeirotis, Gravano & Mehran (2001)** looked at automating the classification of search interfaces to build a Yahoo!-like topic hierarchy. Their system uses carefully chosen probe queries, to maximize the likelihood of correct classification.

For reasons of efficiency, reduced network congestion and possibly improving result quality, it is not always appropriate to send all queries to all known search engines. Selection is the problem of selecting a subset of the search engines for a particular user query. This problem is illustrated in box 3 of Figure 1. Manual selection is possible but trivial. The following studies consider automatic selection.

Callan, Lu & Croft (1995) introduced the well known CORI server selection function, based on term occurrence information which is available via STARTS or probe queries. Other similar functions are GlOSS (Gravano, Garcia-Molina & Tomasic 1999) and CVV (Yuwono & Lee 1997). **Craswell, Bailey & Hawking (2000)** evaluated CORI, vGIOSS and CVV using resource descriptions generated from probe queries and found CORI was the best selection algorithm. Other approaches, for use in varied environments, include (Hawking & Thistlewaite 1999, Dreilinger & Howe 1997, Fuhr 1999, Rasolofo, Abbaci & Savoy 2001).

Query translation is the problem of transforming the user query into a language that is accepted by the recipient search engine. This problem is illustrated in box 4 of Figure 1. Translation issues arise when different search engines accept different query languages, for example when processing boolean queries (Chang, García-Molina & Paepcke 1996, Chidlovskii & Borghoff 1998). However, on the Web a string of keywords is a useful lowest common denominator.

Result merging is the problem of presenting the combined results of engines in a useful fashion. For example, in a ranked list with the most relevant documents ranked near the top. This problem is illustrated in box 5 of Figure 1.

Again it is possible to rely on cooperative protocols. With cooperation, ranking can be done uniformly across all search engines, for example *STARTS* (Gravano et al. 1997) or simply use algorithms which generate comparable match scores. However, for similar reasons of non compliance, cooperative protocols are not widely used on the Web.

Effective merging can be based on the scores or ranks assigned by the selected search interfaces, or on the downloaded contents of the documents in question (Lawrence & Giles 1998). **Craswell, Hawking & Thistlewaite (1999)** found the document download method to be most effective, particularly when ranking with a high quality ranking algorithm and a reference set of collection-wide statistics.

With all this work in distributed information retrieval, it is surprising that so little work has been done on interface discovery, particularly since all the other work relies on having a set of known interfaces.

3 Interface Detection

A search interface allows a user to search some set of items without altering them. The user enters a query, by typing or selecting options, to describe the item(s) of interest. Results might be a page linking to items (usual search engine results), a page containing items (a phone list search) or a single page ("I'm feeling lucky" in Google). The item(s) found should somehow match the query.

The vast majority of search interfaces on the Web are HTML forms. For this reason we ignore other types of interfaces such as Java GUIs. It is easy to find large numbers of HTML forms, by crawling the Web and scanning crawled pages for form tags. Therefore the detection problem becomes: given an HTML form, is it a search interface? Note, most forms have a target URL (action), so although our classification is in terms of forms we sometimes refer to form targets (for example Table 3 and Table 10).

Web search interfaces commonly allow users to search "item sets" such as: Web pages from a general crawl or from single site, products for sale, geographic locations, people, dictionary definitions or bibliographic entries. Forms which are not search interfaces include discussion group interfaces, mailing list subscription forms, purchase forms in an online shop, Web-based email forms and Web site membership forms. In our ANU test set, about 50% of HTML forms were search interfaces.

When classifying HTML forms into search interfaces or non-search interfaces, it is possible to take a pre-query or a post-query approach. In a pre-query approach, the form itself and the page containing it can be analyzed, in order to make the classification. In post-query classification, one or more queries can be sent to the system in question, and the resulting pages used as an indicator. We choose the pre-query approach for reasons of politeness: it is impolite to send arbitrary queries to a purchase form or a discussion group simply for the sake of classification.

Form parameters for automatic features
name parameter for an *input* control
value parameter for an *input* control
name parameter for a *form*
distinct *word* from a form *action*

Table 1: This table describes four places in a HTML form from which we generate features.

	ANU set	Random Web Set
Num features	597	861

Table 2: Number of distinct features generated automatically for both the ANU and random Web training sets.

The remainder of this section describes methods we use for search interface detection. We take a machine learning approach, so we first describe the features upon which we base our classification, then the classification method itself.

3.1 Feature generation

HTML forms contain complex structure that can be exploited to obtain a rich set of features. This section describes one method for automatically generating features for an HTML form with the goal of obtaining a representation useful for interface detection.

Features can automatically be generated for a set of forms based on values for certain parameters found in the HTML form markup. The parameters considered in this paper are in table 1.

The distinct *word* from a form *action* refers to a string of characters that appear between slashes (/) in the form action (ignoring the colon required by the HTTP protocol). For example, if the action for a form is `http://search.anu.edu.au/external/` then the distinct words would be *http*, *search.anu.edu.au* and *external*.

Features can also be automatically generated based on the types of form controls present in the HTML form, for example existence of text controls and password controls. In addition, features can be generated from the number of controls, for example a form having a single text control versus the form having multiple text controls.

To illustrate the process of automatically generating features from the HTML code, consider the example in figure 2. The top box shows the HTML markup for a form and the bottom box the resulting features that are automatically generated.

In this study, automated feature generation was carried out separately on both the ANU training set and the random Web training set.

The resulting features from this automated generation method are too numerous to list individually. Instead, Table 2 summarizes the total number of automatically generated features obtained from the two training sets.

Table 2 suggests that there is more variation in the structure and content of forms on the Web than compared with the ANU Web domain. Although the number of forms in the random Web sample outnumber those in the ANU sample by 18% (260 vs 219), the number of unique features found from the random Web sample outnumber the ANU sample by 44%. Section 7 presents some consequences of this phenomenon.

	search	non-search
ANU training set	149 (t:34)	70 (t:43)
ANU test set	185 (t:24)	199 (t:60)
Random Web training set	150 (t:80)	110 (t:88)
Random Web test set	150 (t:81)	113 (t:92)

Table 3: Training and test sets. Listed are the number of forms of each type (search and non-search) in each set. Also listed are the number of unique URLs targeted by the forms (t:).

3.2 Classification

Having automatically generated a rich set of features, applying a classification algorithm is a simple matter. In this case we choose the C4.5 learning algorithm as our main algorithm, as implemented in Weka (UW 2001).

We chose the algorithm because it is well known, implemented in multiple places and amenable to the type of features generated (mostly binary). More importantly, the algorithm produces a classification rule (tree) which is easily understandable, publishable in its entirety and implemented in any language using nested conditionals. Further, tests in Section 7.2 show that other classifiers are not significantly more effective.

4 Testbeds

Collections from two domains are used in this study, an ANU (Australian National University) and random Web set. Collections were manually sampled by the authors and examples labelled as either a search interface or non search interface. These collections provide a testing ground for experimenting with HTML form features and evaluating the classification success of any features developed.

4.1 ANU collection

A training set of pages was obtained from hosts in the ANU Web domain (`anu.edu.au`), using a February 2001 crawl of around 430 000 pages. We identified the 6 500 pages containing HTML forms and sorted them in URL order. Then, selecting every 30th page, we identified a training set of 200 pages. For simplicity of judging and labelling, we then split files containing multiple forms, giving a training set of 219 forms.

This set was then manually judged by the authors and each form was labelled as either a search interface or a non search interface. As a result of this labelling, there were 149 search interfaces and 70 non search interfaces in this training set.

A test set of forms was also obtained from the same crawl. The same procedure in choosing and judging the forms was applied, except that 300 forms were picked at random from the crawl (three applications with a different offset in the list of 6 500, to make sure that the test set is different to the training set, and choosing every 60th page). After decomposing multiple forms from a page into their own files and judging this test set, there were 185 search interfaces and 199 non search interfaces.

The test set is larger than the training set because it was gathered later when the judging and labelling procedure was more refined.

```
<form name="altavista" method="GET"
action="http://www.altavista.yellowpages.com.au/cgi-bin/query">
<input type="text" name="q" maxlength="800" value="">
<input type="submit" value="Search" name="submit2">
<input type="hidden" name="mss" value="simple">
<input type="hidden" name="pg" value="q">
<input type="hidden" name="what" value="web">
<input type="hidden" name="enc" value="iso88591">
<input type="hidden" name="kl" value="XX">
<input type="hidden" name="locale" value="xx">
</form>
```

```
inputType-SingleText
inputType-Submit
inputType-Hidden
inputType-text:Name=q
inputType-submit:Name=submit2
inputType-hidden:Name=mss
inputType-hidden:Name=pg
inputType-hidden:Name=what
inputType-hidden:Name=enc
inputType-hidden:Name=kl
inputType-hidden:Name=locale
inputType-submit:Value=search
inputType-hidden:Value=simple
inputType-hidden:Value=q
inputType-hidden:Value=web
inputType-hidden:Value=iso88591
inputType-hidden:Value=xx
inputType-hidden:Value=xx
FormName:altavista
actionWord:http:
actionWord:www.altavista.yellowpages.com.au
actionWord:cgi-bin
actionWord:query
```

Figure 2: This figure shows automatic feature generation in action. The top box contains some sample HTML code for the declaration of a form. The bottom box shows the resulting features derived from the HTML form content.

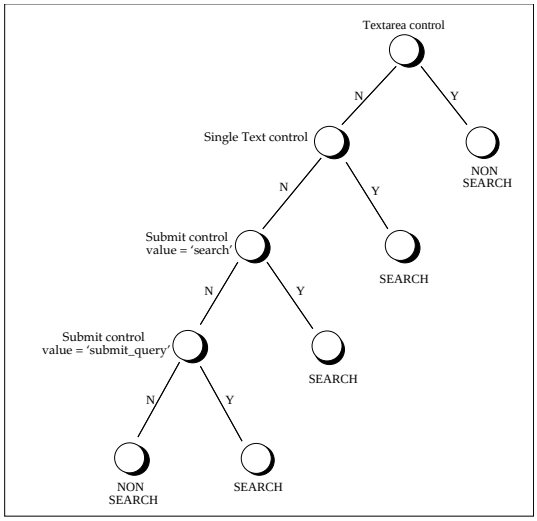

Figure 3: Decision tree built from the ANU training set. The tree was built with 597 available features.

4.2 Random Web collection

A sample set of pages was obtained from the Web and notably outside the ANU domain. Since a full crawl of the Web was not available, a different strategy from above was needed in order to obtain a training and test set of search interfaces and non-search interfaces.

The Web site `http://www.searchengineguide.com/` is a directory of search interfaces and was used to obtain a sample set of search interfaces for the random Web collection. These interfaces are submitted manually by site owners, rather than automatically detected[1]. Search interfaces were chosen from a broad range of topics that index news and media, business, society and science. From this source, a sample of 150 search interfaces were selected for the random Web training set and 150 were selected for the random Web test set. This method is argued to be random because this Web site acts as a pointer to a very broad range of actual search interfaces on the Web. So although the interfaces were obtained from one specific site, the interfaces actually originate from all over the Web.

To obtain the set of non search forms, a list of Web sites linked from `http://www.searchengineguide.com/` and `http://www.dmoz.org`[2] were followed to their home-pages. The home pages were then analyzed to find candidate forms. These forms were then judged and if they were non search interfaces, they were kept to make up the set of non search interfaces. With this method, a set of 110 non search interfaces were eventually found for the training set and 113 for the test set.

Table 3 compares the training and test sets for both the ANU and random Web collections.

5 Results

Decision trees are built with the C4.5 learning algorithm using the automatically generated features from section 3.1. The implementation of C4.5 used was obtained from (UW 2001).

	predicted search	predicted non search
actual search	146	3
actual non search	2	68

Table 4: Confusion matrix for the ANU training set using rules derived from Figure 3.

	predicted search	predicted non search
actual search	180	5
actual non search	9	190

Table 5: Confusion matrix for the ANU test set using rules derived from Figure 3.

5.1 Decision tree for the ANU

This section presents the decision tree that was generated from the ANU training set using automatically generated features.

Figure 3 shows the decision tree constructed for the ANU training set with 597 features. The success of the classification is given in table 4. Rules generated from the tree have an accuracy of 98% and a precision of 99%.

Table 5 shows the classification success when the rules are applied on the ANU test set. The accuracy for this classification is 96% and the precision is 95%.

5.2 Decision tree for the random Web

This section presents a decision tree generated from the random Web training set using automatically generated features.

Figure 4 shows the decision tree constructed for the random Web set with 861 automatically generated features. Table 6 shows the results of this classification on the training set with the tree from Figure 4. Using the rules from the decision tree in figure 4, the classification has an accuracy of 92% and a precision of 96%.

Table 7 shows the results of this classification on the test set with the rules generated from the decision

[1] personal correspondence on 25/09/01 with Robert Clough, Web-master of Search Engine Guide.

[2] A directory which acts as a pointer to a broad range of Web documents.

	predicted search	predicted non search
actual search	135	15
actual non search	5	105

Table 6: Confusion matrix for the random Web training set using rules derived from the tree in Figure 4

	predicted search	predicted non search
actual search	131	19
actual non search	20	93

Table 7: Confusion matrix for the random Web test set using rules derived from the tree in Figure 4

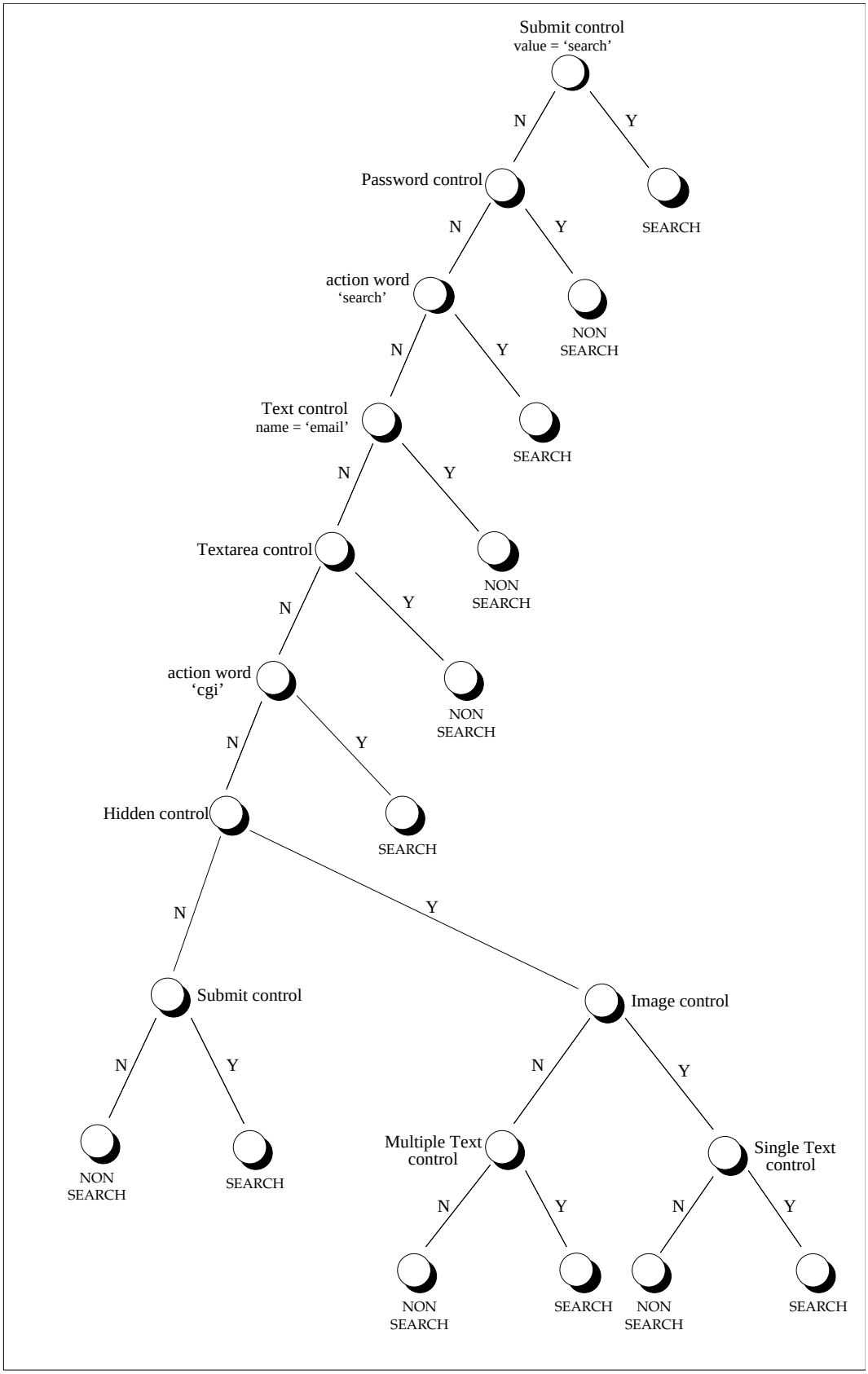

Figure 4: Decision tree built from the random Web training set with 861 available features.

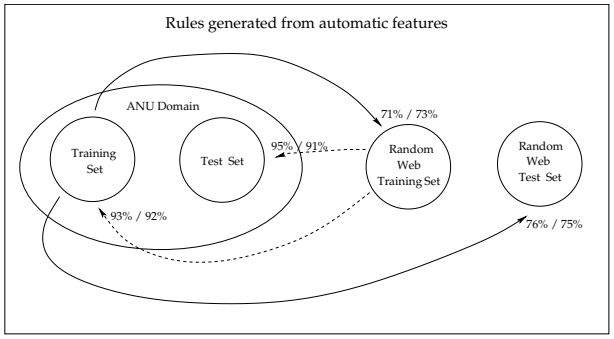

Figure 5: Cross validation across domains. Circles represent training/test sets and lines represent tests (dashed for the "random" rule and solid for the ANU rule). Each line is associated with a pair of percentages, accuracy and precision respectively.

tree in figure 4. The classification has an accuracy of 85% and a precision of 87%.

6 Discussion

For the ANU collection, the classification success was much better than the random Web collection. The results are near perfect with an accuracy of 98% and 96% for the training and test sets respectively. The precision is 99% and 95% for the training and test sets respectively. Since all these figures are reasonably similar, the rules generated by C4.5 from the automatically generated features appear to be robust and able to successfully classify new examples from the ANU domain.

The random Web classification obtained an accuracy of 92% and 85% for the training and test sets respectively. The precision was 96% and 87% for the training and test sets respectively. The accuracy and precision obtained in the training set is slightly higher than the test set. Overall, the success is lower than the ANU domain but this is believed to be due to more variation in the example interfaces found in the Web collection. A substantial number of the search interfaces found in the ANU collection are actually the same interface, being the search interface of Panoptic which is the ANU search service. So naturally, the ANU collection is not as diverse as the random Web collection. Overall, taking the lower bound on the classification for the random Web collection, an accuracy of 85% and precision of 87% is still very good.

7 Further analysis

We now address three further questions. First how general are the rules in Figure 3 and Figure 4? We test this by cross validation. Second, have our experiments been affected by our choice of the C4.5 classifier? Third, what can be learned from applying the Figure 4 rule on a large Web crawl?

7.1 Rule cross validation

This section compares the results of applying rules generated from the ANU collection onto the random Web collection and vice versa. Figure 5 shows the results of this cross validation where some general conclusions can be drawn. Firstly it appears that the ANU rules have limited application to the random Web collection. The best accuracy for these rules was 76% on the random Web test collection. This is significantly lower than an accuracy of 85% achieved with rules generated from the random Web decision tree. So this would indicate that forms found on the ANU are not representative of the Web at large and so the rules generated from this limited exposure to the Web consequently have limited success at large.

Rules generated with the random Web collection perform well on the ANU collection. The accuracy is about 95% and the precision 91%. This is only marginally worse than using the rules generated with the ANU decision tree.

With these cross validation experiments, two rule generation strategies emerge. Firstly, for the best classification results on a certain domain, training should be based on local examples, providing very good performance over that domain. Secondly, when no local training data is available, training can be based on general search interfaces, providing a general purpose rule (such as Figure 4) with good performance in multiple domains.

	Predicted search		Predicted non-search	
Actual search	C4.5	145	C4.5	4
	SVM	143	SVM	6
	Knn	146	Knn	3
Actual non-search	C4.5	4	C4.5	66
	SVM	4	SVM	66
	Knn	5	Knn	65

Table 8: Other classifiers, ANU.

	Predicted search		Predicted non-search	
Actual search	C4.5	132	C4.5	18
	SVM	136	SVM	14
	Knn	138	Knn	12
Actual non-search	C4.5	11	C4.5	99
	SVM	23	SVM	87
	Knn	38	Knn	72

Table 9: Other classifiers, Random.

7.2 Evaluating other classifiers

We now perform the same experiments with different Weka (UW 2001) classification schemes, to ensure that our results were not dependent on C4.5. In addition to C4.5 we use support vector machines (SVM) and K-nearest neighbor (Knn).

Results are in Table 8 and Table 9. None of the other classifiers provides a significant advantage over C4.5. Although the true positives are slightly better in Table 9 for SVM and Knn, this is counteracted by a higher false positive rate. In this application false positives are particularly undesirable in that they might lead to a query being sent to a non-search interface.

Perhaps with tuning, even better performance could be achieved using these and other classification schemes. However, for the purposes of this study, the C4.5 classifier has performed admirably.

7.3 Real-world testing

One of the authors, who had not worked with the original code, reimplemented the Figure 4 tree in Perl and ran it over a 2.5 million page crawl of Australian research institutions (see `http://rf.panopticsearch.com/`). After minor adjustment, the new script produced results which closely matched

University	Search interface targets	
	2+ forms	1+ forms
anu.edu.au	627	2 653
qut.edu.au	593	1 349
unimelb.edu.au	181	405
cqu.edu.au	176	489
latrobe.edu.au	87	389
monash.edu.au	86	1 610
usyd.edu.au	69	515
rmit.edu.au	67	287
unsw.edu.au	65	327
uq.edu.au	55	4 905
canberra.edu.au	51	624
uwa.edu.au	48	438
murdoch.edu.au	45	89
uws.edu.au	38	327
curtin.edu.au	28	89
gu.edu.au	26	75
deakin.edu.au	25	107
mq.edu.au	22	82
uow.edu.au	18	100
adelaide.edu.au	16	62
scu.edu.au	14	23
csu.edu.au	13	63
unisa.edu.au	13	40
ntu.edu.au	9	19
flinders.edu.au	7	27
uts.edu.au	7	319
swin.edu.au	6	13
utas.edu.au	5	26
jcu.edu.au	4	79
acu.edu.au	4	9
newcastle.edu.au	4	20
une.edu.au	3	12
ballarat.edu.au	3	7
usq.edu.au	2	16
ecu.edu.au	1	2
bond.edu.au	1	15
usc.edu.au	1	2
vu.edu.au	0	4

Table 10: Target URLs of detected search forms, by university. For example, of the 19 337 detected targets 2 653 were at ANU and 627 of these were targeted by two or more detected search forms.

those of Tables 6 and 7. The final Perl script, including test code for generation of confusion matrices, was 150 lines.

Of the 530 763 forms in Research Finder crawl 212 569 or 40% were flagged as search forms.

More interesting than forms are form targets. For example, 2945 search forms are detected, all of which target the URL `http://search.anu.edu.au/anu` (via their form action). Viewed by a user or meta searcher, this is one search rather than 2945, although it might be used in different ways by different forms. Note, this also allows us to get a mixed view, when a URL is targeted by both detected search forms and detected non-search forms.

We found 44 744 form targets in the Research Finder crawl, of which 19 337 or 43% were targets of detected search forms. Of these 1563 were mixed, being targeted by both detected search forms and detected non-search forms.

Table 10 is a summary of how many of the 19 337 search targets appeared at each Australian University. It counts the targets of searches, rather than the sources. For example, if `http://www.uq.edu.au/search/index.asp` is the target URL of 4681 forms, it is counted once in Table 10. We include a separate count for forms which are targets of two or more detected search forms.

An interesting case is University of Queensland (`uq.edu.au`), which included 55 detected search targets with multiple forms, but a further 4850 with only one form. The top ten most referenced search targets are:

```
http://www.uq.edu.au/search/index.asp yes=4681
http://www.uq.edu.au/search/index.asp? yes=1662
http://www.sph.uq.edu.au/search/sphsearch.idq yes=1180
http://www.uq.edu.au/myadvisor/index.html yes=263
http://www.its.uq.edu.au/factsheets_search.html yes=234
http://www.commerce.uq.edu.au/cgi-bin/htsearch yes=104
http://www.its.uq.edu.au/faq_search.html yes=77
http://www.uq.edu.au/myadviser/index.html yes=57
http://asc.uq.edu.au/gradnet/main.php yes=56 no=1
http://www.library.uq.edu.au/uql/cgi-bin/subjectsearch.pl yes=27
```

all of which are search interfaces.

The least referenced UQ targets included 4065 of the form `http://student.uq.edu.au/~sNNNNNNN/`, where NNNNNNN is a student number. These forms appear on disclaimer pages, asking people to click on a form button to proceed. Because the forms have few other features, they reach the leftmost SEARCH node in Figure 4.

The worst case we examined was ANU (`anu.edu.au`), with the top ten targets:

```
http://search.anu.edu.au/anu yes=2600 no=345
http://search.anu.edu.au/external yes=660 no=65
http://tux.anu.edu.au/twiki/bin/search/know/ yes=65
http://netserve.anu.edu.au/commpro.html yes=36
http://msowww.anu.edu.au/cgi-bin/htsearch.wrap yes=35
http://arp.anu.edu.au/arp-cgi-bin/esc yes=31
http://law.anu.edu.au/legalworkshop/lwscripts/(none) yes=23
http://tux.anu.edu.au/twiki/bin/search/twiki/ yes=22
http://tux.anu.edu.au/twiki/bin/rdiff/documentation/
                                        webstatistics yes=20
http://tux.anu.edu.au/twiki/bin/view/documentation/
                                        webstatistics yes=20
```

They include mixed judgments on a definite search interface (`http://search.anu.edu.au`) and a number of interfaces which fall foul of the same issue that included UQ student pages (leftmost SEARCH node in Figure 4).

We conclude that, although the Figure 4 rule was easily implemented and allowed the first ever automated overview of its type, further work would be required to discover which of the 19 337 "detected search interfaces" are actually search interfaces. Despite this, the large number of detected search targets suggests that a meta search project, paralleling the Research Finder search engine project, would have access to plenty of search interfaces. The list of 19 337 would be a useful starting point in such an endeavor.

8 Conclusion

This paper has shown how to automatically discover search interfaces from a set of HTML forms. A decision tree was developed with the C4.5 learning algorithm using automatically generated features from the HTML markup that can give a classification accuracy of about 85% for general Web interfaces.

An obvious next step is to apply methods such as Callan & Connell (2001) or Ipeirotis et al. (2001) to invent the first fully automated search engine for search engines. Our tests in Australian research institutions show that there are many candidate search engines, although further work will be needed to eliminate false positives.

References

Callan, J. & Connell, M. (2001), Query-based sampling of text databases, *in* 'ACM Transactions on Information Systems', Vol. 19, pp. 97–130.

Callan, J. P., Lu, Z. & Croft, W. B. (1995), Searching distributed collections with inference networks, *in* 'SIGIR'95', ACM Press, pp. 21–28.

Chang, C.-C. K., García-Molina, H. & Paepcke, A. (1996), 'Boolean query mapping across heterogeneous information sources', *IEEE Transactions on Knowledge and Data Engineering* **8**(4), 515–521.

Chidlovskii, B. & Borghoff, U. M. (1998), Query translation for distributed information gathering on the web, *in* 'International Database Engineering and Application Symposium', pp. 214–223.

Craswell, N., Bailey, P. & Hawking, D. (2000), Server selection on the World Wide Web, *in* 'Proceedings of the Fifth ACM Conference on Digital Libraries', pp. 37–46.

Craswell, N., Hawking, D. & Thistlewaite, P. B. (1999), Merging results from isolated search engines, *in* 'Australasian Database Conference', pp. 189–200.

Dreilinger, D. & Howe, A. E. (1997), 'Experiences with selecting search engines using metasearch', *ACM Transactions on Information Systems* **15**(3), 195–222.

Fuhr, N. (1999), 'A decision-theoretic approach to database selection in networked IR', *ACM Transactions on Information Systems* **17**(3), 229–229.

Gravano, L., Chang, C.-C. K., García-Molina, H. & Paepcke, A. (1997), STARTS: Stanford proposal for Internet meta-searching, pp. 207–218.

Gravano, L., Garcia-Molina, H. & Tomasic, A. (1999), 'Gloss: text-source discovery over the internet', *ACM Transactions on Database Systems (TODS)* **24**(2), 229–264.

Hawking, D. & Thistlewaite, P. (1999), 'Methods for information server selection', *ACM Transactions on Information Systems.* **17**(1), 40–76.

Ipeirotis, P., Gravano, L. & Mehran, S. (2001), 'Probe, count, and classify: categorizing hidden web databases', *ACM SIGMOD* **30**(2), 67 – 78.

Lawrence, S. & Giles, C. L. (1998), 'Context and page analysis for improved web search', *IEEE Internet Computing* **2**(4), 38–46.

Perkowitz, M., Doorenbos, R., Etzioni, O. & Weld, D. (1997), 'Learning to understand information on the internet: An example-based approach', *Machine Learning* (to appear).

Rasolofo, Y., Abbaci, F. & Savoy, J. (2001), Approaches to collection selection and results merging for distributed information retrieval, *in* 'CIKM'01', ACM Press, pp. 191–198.

UW (2001), 'Weka machine learning project'. A free suite of machine learning tools available from the University of Waikata. http://www.cs.waikato.ac.nz/ ml/.

Yuwono, B. & Lee, D. L. (1997), Server ranking for distributed text retrieval systems on the internet, *in* R. Topor & K. Tanaka, eds, 'DASFAA '97', World Scientific, Singapore, Melbourne, pp. 41–49.