This paper has been submitted to the Journal of Universal Computer Science.

ProFusion*: Intelligent Fusion from Multiple, Distributed Search Engines[1]

Susan Gauch
(Department of Electrical Engineering and Computer Science
The University of Kansas
sgauch@eecs.ukans.edu)

Guijun Wang
(Department of Electrical Engineering and Computer Science
The University of Kansas
gwang@eecs.ukans.edu)

Mario Gomez
(Department of Electrical Engineering and Computer Science
The University of Kansas
mgomez@eecs.ukans.edu)

*http://www.designlab.ukans.edu/ProFusion.html

Abstract: The explosive growth of the World Wide Web, and the resulting information overload, has led to a mini-explosion in World Wide Web search engines. This mini-explosion, in turn, led to the development of ProFusion, a meta search engine. Educators, like other users, do not have the time to evaluate multiple search engines to knowledgeably select the best for their uses. Nor do they have the time to submit each query to multiple search engines and wade through the resulting flood of good information, duplicated information, irrelevant information, and missing documents. ProFusion sends user queries to multiple underlying search engines in parallel, retrieves and merges the resulting URLs. It identifies and removes duplicates and creates one relevance-ranked list. If desired, the actual documents can be pre-fetched to remove yet more duplicates and broken links. ProFusion's performance has been compared to the individual search engines and other meta searchers, demonstrating its ability to retrieve more relevant information and present fewer duplicates pages. The system can automatically analyze queries to identify its topic(s) and, based on that analysis, select the most appropriate search engines for the query.

Categories: Information Systems

[1]This paper is an extension of "Information Fusion with ProFusion", Susan Gauch and Guijun Wang, *WebNet '96: The First World Conference of the Web Society*, San Francisco, CA, October 1996.

This paper has been submitted to the Journal of Universal Computer Science.

1 Introduction

There are a huge number of documents on the World Wide Web, making it very difficult to locate information that is relevant to a user's interest. Search tools such as InfoSeek [InfoSeek 1996] and Lycos [Lycos 1996] index huge collections of Web documents, allowing users to search the World Wide Web via keyword-based queries. Given a query, such search tools search their individual index and present the user with a list of items that are potentially relevant, generally presented in ranked order. However large the indexes are, still each search tool indexes only a subset of all documents available on WWW. As more and more search tools become available, each covering a different, overlapping subset of Web documents, it becomes increasingly difficult to choose the right one to use for a specific information need. ProFusion has been developed to help users deal with this problem.

2 Related Work

There are several different approaches to managing the proliferation of Web search engines. One solution is to use a large Web page that lists several search engines and allows users to query one search engine at a time. One example of this approach is All-in-One Search Page [Cross, 1996], another is the Washington Researcher's Search Engine Reference Page [WR 1996]. Unfortunately, users still have to choose one search engine to which to submit their search.

Another approach is to use intelligent agents to bring back documents that are relevant to a user's interest. Such agents [Balabanovic et al. 1995][Knoblock et al. 1994] provide personal assistance to a user. For example, [Balabanovic et al. 1995] describes an adaptive agent that can bring back Web pages of a user's interest daily. The user gives relevance feedback to the agent by evaluating Web pages that were brought back. The agent them makes adjustment for future searches on relevant Web pages. However, these agents [Balabanovic et al. 1995][Knoblock et al. 1994] gather information from only their own search index, which may limit the amount of information they have access to.

A different approach is the meta search method which builds on top of other search engines. Queries are submitted to the meta search engine which in turn sends the query to multiple single search engines. When retrieved items are returned by the underlying search engines, it further processes these items and presents relevant items to the user. ProFusion [Gauch 1996], developed at the University of Kansas, is one such search engine.

The idea of using a single user interface for multiple distributed information retrieval systems is not new. Initially, this work concentrated on providing access to distributed, heterogeneous database management systems [Arens et al. 1993]. More recently, meta searchers for the WWW have been developed. For example, SavvySearch [Dreilinger 1996] selects the most promising search engines automatically and then sends the user's query to the selected search engines (usually 2 or 3) in parallel. SavvySearch does very little post-processing. For example, the resulting document lists are not merged. MetaCrawler [Selberg and Etzioni 1995][Selberg and Etzioni 1996], on the other hand, sends out user's query to all search engines it handles and collates search results from all search engines. What distinguishes ProFusion from others is that it uses sophisticated yet computationally efficient post-processing.

3 ProFusion

3.1 General Architecture

ProFusion accepts a single query from the user and sends it to multiple search engines in parallel. The current implementation of ProFusion supports the following search engines: Alta Vista [DEC 1996], Excite [Excite 1996], InfoSeek [InfoSeek 1996], Lycos [Lycos 1996], Open Text [Open Text 1996], and WebCrawler [GNN 1996]. By default, ProFusion will send a query to Alta Vista, Excite, and InfoSeek, but the user may select any or all of the supported search engines. If the user prefers, the system will analyze the user’s query, classifying it into a topic or multiple topics. Based on this analysis, the system will automatically pick the top three search engines that perform best on this topic or these topics. However the search engines are selected, the search results they return are then further processed by ProFusion. The post-processing includes merging the results to produce a single ranked list, removing duplicates and dead references, and pre-fetching documents for faster viewing and further analysis[2].

3.2 User Interface

ProFusion queries are simple to form; they are merely a few words describing a concept. Online help is available via a Help button that leads users to a page explaining the query syntax, including sample queries. Users need only enter a query and press the "Search" button, however there are several options available which give the user more control over their search. The first option specifies whether or not the user wants to have a short summary displayed for each retrieved item. The benefit of displaying retrieved items without a summary is that a user can more quickly scan retrieved items by title. The second option allows users to manually select the search engine(s) to which their query is sent, or to have the system choose automatically (described in Section 3.1). If the user is selecting the search engines, they may choose any number of search engines from one to all six. When “Automatic Pick Best 3” is selected, the system to selects the best three search engines based on the words in the query.

[2]Note: Some of the more computationally expensive features (e.g., pre-fetching and broken link removal) are only available through the private ProFusion interface. They may be added as options on the public page.

3.3 Duplicate Removal

Since the underlying search engines overlap in the Web pages they index, it is highly likely that they will return some of the same pages in response to a given query. ProFusion attempts to remove these duplicated pages, using a few simple rules. The simplest case is when the identical URL has been returned by multiple search engines. Clearly, if two items have exactly the same URL, they are duplicates. More complex rules are necessary to handle the case where the identical page is referenced by slight variations on the same address. For example, the URLs is "http://server/" and is "http://server/index.html" reference the identical page. Handling the previous two cases removes approximately 10 - 20% of the retrieved URLs. However, duplicates may also occur because multiple copies of the same page may exist at different locations. Thus, if two items have different URLs but the same title, they might be duplicates. In this case, we break a URL into three parts: protocol, server, and path. We then use n-gram method to test the similarity of two paths. If they are sufficiently similar, we consider them as duplicates. This appears to work very well in practice, removing an additional 10 - 20% of the URLs, but runs the risk that the URLs point to different versions of the same document, where one is more up-to-date than the other. To avoid this risk, we could retrieve the potential duplicates in whole or in part, and then compare the two documents. However, this would increase network traffic and might be substantially slower. This capability has been developed, but has not been added to the public version of ProFusion due to response time and network traffic concerns.

3.4 Merge Algorithms

How to best merge individual ranked lists is an open question in searching distributed information collections [Voorhees et al. 1994]. Callan [Callan et al. 1995] evaluated merging techniques based on rank order, raw scores, normalized statistics, and weighted scores. He found that the weighted score merge is computationally simple yet as effective as a more expensive normalized statistics merge. Therefore, in ProFusion, we use a weighted score merging algorithm which is based on two factors: the value of the query-document match reported by the search engine and the estimated accuracy of that search engine.

For a search engine *i*, we calculated its confidence factor, CF_i, by evaluating its performance on a set of over 25 queries. The CF_i reflects the number of total relevant documents in top 10 hits and the ranking accuracy for those relevant documents. Based on the results, the search engines were assigned CF_is ranging from 0.75 to 0.85. More work needs to be done to systematically calculate and update the CF_is, particularly developing CF_is which vary for a given search engine based on the domain of the query.

When a set of documents is returned by search engine *i*, we calculate the match factor for each document *d*, M_{di}, by normalizing all scores in the retrieval set to fall between 0 and 1. We do this by dividing all values by the match value reported for the top ranking document. If the match values reported by the search engine fall between 0 and 1, they are unchanged. Then, we calculate the relevance weight for each document *d*, R_{di}, by multiplying its match factor, M_{di}, by the search engines confidence factor, CF_i. The document's final rank is then determined by merging the sorted documents lists based on their relevance weights, R_{di}. Duplicates are identified during the merging process. When duplicates are removed, the surviving unique document's weight is set to the maximum R_{di} value of all the copies.

3.5 Search Result Presentation

The merge process described in the previous section yields a single sorted list of items, each composed of a URL, a title, a relevance weight, and a short summary. These items are then displayed to the user in sorted order, with or without the summary, depending on user's preference.

3.6 Other Implementation Details

ProFusion is written in Perl and is portable to any Unix platform. It contains one Perl module for each search engine (currently six) which forms syntactically correct queries and parses the search results to extract each item's information. Other modules handle the user interface, the document post-processing, and document fetching. Due to it's modular nature, it is easy to extend ProFusion to additional search engines.

ProFusion's main process creates multiple parallel sub-processes, and each sub-process sends a search request to one search engine and extracts information from the results returned by the search engine. The main process begins post-processing when all sub-processes terminate by returning their results or by timing out (60 seconds in the current prototype).

4 Intelligent Search Engine Selection

A recent extension to ProFusion allows it to automatically select the best three search engines for a given query, based on the query's domain. Different search engines perform differently in different topics, and users don't have the knowledge nor the time to know which search engines work best for specific queries. Our system automatically identifies the topic(s) in a user's query, selects the search engines which have been shown to do best on this/these topic(s), and submits the queries to those search engines. This operation is done quickly, with no user feedback required, and has been shown to improve the quality of the search results.

To do this, we created a knowledge base which: 1) contains a taxonomy which reflects the contents of the Web; 2) associates words with specific topics; 3) associates search engines with specific topics.

4.1 Selecting the Taxonomy

The first step in being able to classify queries is to select a taxonomy from which to build the knowledge base. This taxonomy must reflect the breadth of topics available on the Internet today. Since it is representative of users' interests, we used the list of newsgroups on the Internet for the classification process. The first task was to select, from the newsgroup names, a set of categories. To do this, we parsed the newsgroup names (e.g., rec.arts.movies.reviews) into individual tokens (rec, arts, movies, reviews) which form a hierarchy of topics and sub-topics. From this list of tokens, we extracted approximately 4000 unique terms. For each term, we calculated how many times it occurred in the list of newsgroup names. For instance, "science" appears 25 times, "engineering" 25 times, and "recreation" 73 times in the newsgroup list (i.e., they have 25, 25, 73 sub-newsgroups respectively). This data was combined with our everyday knowledge of the world to select the final categories. For instance, society, law, and government are closely related to each other. Hence, they are merged into one category. The resulting set of 13 categories was:

- Science and Engineering
- Computer Science
- Travel
- Medical and Biotechnology
- Business and Finance
- Social and Religion
- Society, Law and Government
- Animals and Environment
- History
- Recreation and Entertainment
- Art
- Music
- Food

4.2 Creating the Dictionary

For each category, it is necessary to have a set of associated terms. These terms are used to map from the user queries to the appropriate categorie(s). We used the 4,000 terms extracted from the newsgroup names above for this purpose, automatically associating the terms to the categories using the topic-sub-topic hierarchy. The current dictionary should be improved so that more words in more queries can be accurately assigned to categories. One approach would be to expand the number of words in the dictionary This can be done by including words from the actual newsgroup articles. Another approach would be to make better use of the existing dictionary words. This can be done by incorporating stemming on query words and dictionary words to increase the number of matches.

4.3 Calibrating the Search Engines

The next step in the process of building the knowlege base was to calibrate the effectiveness of each of the 6 search engines on queries in each of the 13 different categories. To do this, we submitted 48 different queries (approximately 4 per category) to each of the 6 underlying search engines. For each query, we examined the top 10 retrieved documents, classifying it as relevant or irrelevant. Then, we calculated the performance of each search engine on each query using the following formula:

$$\left(\frac{\sum_{i=1}^{10} N_i}{10} * \frac{R}{10} \right) \div 0.2929$$

where N_i = 0 if document i is irrelevant, 1/i otherwise.
R is the number of relevant documents in the set of 10.

This formula is the product of two factors, *rank order factor* and *precision*, normalized to yield numbers in [0,1]. The rank order factor takes into account the position in the retrieval set of the relevant documents, rewarding those which present the relevant documents near the top of the list. In contrast, precision is a measure of the number of relevant documents in the retrieved set of 10, regardless of position.

If the rank order factor is used alone, it is too heavily biased for position in the list over the total number of relevant documents in the set. For example, if the first document of the top 10 documents retrieved by a search engine is the only relevant document, then the rank order factor would be 1/10 = 0.10. In contrast, if 7 relevant documents are retrieved in positions 4 through 10, the rank order factor would be (0+0+0+1/4+...+1/10)/10 = 0.11, which seems to poorly reflect the relative quality of the two search results. However, precision alone fails to differentiate between two search engines which each retrieve 5 relevant documents, but the first places them in positions 1 through 5 whereas the second places them in positions 6 through 10. To balance both criteria (number and position of relevant documents), we use a measure which is the product of the rank order factor and precision.

Finally, an average score for search engine was calculated for each category by averaging the scores for each query in that category. This average score is the search engine's quality of result with regard to a query in this topic. These scores range from a high of 0.767 (Alta Vista on Food) to a low of 0.011 (OpenText on History). The same search engine did not consistently perform the best across all categories.

For example, the queries on Art did best with WebCrawler, InfoSeek and OpenText whereas the queries on Animals and Environment did best with Lycos, InfoSeek and Excite, and queries on Food do best when directed to AltaVista, Lycos and WebCrawler.

4.4 Selecting the Search Engines

A user's query is classified via a two-level dictionary look-up. The dictionary of words and the related categories used in the prototype were stored in two separate files. The dictionary file is used to map from words to the categories in which they belong. The category file is used to map from categories to the scores of the 6 search engines on those categories. These files contain sorted, fixed length records which are accessed via binary search.

Each query may be associated with multiple categories. The query may contain several words from different categories and/or the individual words may themselves be associated with multiple categories. To select the best search engines for a given query, the system keeps an accumulator for each of the six search engines. When a word's categories are identified, the six search engine performance scores for each category are added to the scores buckets respectively. When all words are handled, we select the three buckets with the highest accumulated scores. Their scores are normalized by dividing by the maximum of the three values. If no word in the query is in the dictionary, then we use a default set of three search engines, those which produce the best average results over all categories.

4.5 Merging Search Results

The three normalized scores for the goodness of match between the chosen search engines and the query, described in the previous section, are used during search result merging. The goal is to rank documents higher when they come from search engines that have, in the past, been particularly good for queries in this category. Normally, during merging, the relevance weight for a given document is the product of the (normalized) scores returned by the search engine and the confidence factor for the given search engine (see section 3.4). When AutoPick is used, this value is further multiplied by the goodness of match score between the query and the search engine. If the default three search engines are selected, weights of 1.0 are used for all three, which leaves the regular relevance weights unchanged.

5 Performance Evaluation

5.1 Manual Pick ProFusion

We invited every student in our Spring 1996 Information Retrieval class to select a query he/she was interested in. They then were asked to perform a search on that query using each of 9 search engines: the six underlying search engines used by

ProFusion (Alta Vista, Excite, InfoSeek, Lycos, Open Text, WebCrawler); ProFusion; and two other meta search engines (MetaCrawler and SavvySearch). Each participant provided relevance judgments for the top 20 retrieved items from each search engine, noting which were broken links and which were duplicates. The performance of each of the search engines was then compared by accumulating the information on the number of relevant documents, the number of irrelevant documents, the number of broken links, the number of duplicates, the number of unique relevance documents, and the precision. Here, precision = number of unique relevant documents divided by total number of documents retrieved (20 documents in this evaluation). The following is a summary of the results from the 12 independent queries, evaluated by the top 20 retrieved documents (total 240 documents evaluated for each search engine).

	Total	Total	Total	Total	Total	Average
Search Engines	number of relevant	number of irrelevant	Broken links	number of unique relevant	number of duplicate relevant	Precision number unique / 240
Single Search Engines						
Alta Vista	108	101	31	99	9	0.41
Excite	129	104	7	122	7	0.51
InfoSeek	99	125	16	87	12	0.36
Lycos	119	104	17	93	26	0.39
Open Text	72	136	32	54	18	0.23
WebCrawler	92	130	18	73	19	0.30
Meta Search Engines						
MetaCrawler	98	118	24	85	13	0.35
SavvySearch	127	84	29	112	15	0.47
ProFusion						
Manual Pick All 6	142	85	13	134	8	0.56
Auto Pick Best 3	170	51	19	166	4	0.69

Table 1: Performance Comparison

From this table, we see that the Manual Pick version of ProFusion outperformed all other search engines on the user selected queries. The only search engine to do better was the Automatic Pick version of ProFusion, which will be discussed in the next paragraph. Automatic Pick ProFusion returned the most relevant documents and hence the best average precision. We attribute this performance to our sophisticated yet efficient merging algorithm, combined with the removal of duplicates. When more of the documents in the top 20 are unique, there is a better chance that more of them are relevant. ProFusion did a better job in duplicate removal than SavvySearch and MetaCrawler. ProFusion has 8 duplicates among 142 relevant documents (5.6%), whereas SavvySearch has 15 duplicates among 127 relevant documents (11.8%) and MetaCrawler has 13 duplicates among 98 relevant documents (13.3%). Similar numbers were observed for duplicates among irrelevant retrieved documents. Surprisingly, the number of duplicates retrieved by many of the individual search engines was higher than those of the meta-search engines. Many times, the same document is accessible from multiple URLs and thus is multiply indexed by the individual search engines. The percentage of broken links retrieved by ProFusion was also lower than any system except Excite.

5.2 Automatic Pick ProFusion

Automatic Pick ProFusion did dramatically better than the Manual Pick version, with 170 relevant documents (by far the most of any system) and only 4 duplicates (by far the least of any system). One measure of quality, however, was poorer. The number of broken links increased from 13 (second best overall) to 19 (just better than the average of 20.6). Manual Pick always included Excite, which decreased the overall number of broken links in its merged set due to its low number of broken links retrieved from Excite. In contrast, the AutoPick version did not always select Excite, so it did not always benefit by averaging in Excite's superior broken link performance.

We believe that this small experiment demonstrates that merging results from the "best of the best" can do better than merging results from all possible contributors. Our student subjects in this experiment tended to query on science or computer science topics, and a more thorough experiment with queries from a wider range interests needs to be done to validate these results. However, we believe that the quality of the search engines varies by topic, making it desirable to select the best search engines for individual queries rather than sending a query to all search engines or a fixed subset. Selecting a subset of the search engines also places less demand on Internet resources.

6 Future Work

Enhancements that are underway include analyzing the retrieved documents to improve the ranking, improving the speed of broken link removal, and incorporating user preferences in the ranking process (e.g., do they prefer content-bearing pages which contain mostly text or summary pages which primarily contain links to further pages). The AutoPick process can be improved in two ways: 1) increase the number of words in the dictionary so that query categories are more accurately identified. 2) incorporate learning into the search engine evaluation on different topics. Search engines may change, and we need a way to automatically keep the our table of which search engines are best for which categories up to date.

New capabilities are also being planned. In particular, we plan to add the ability to automatically rerun searches on a periodic basis, presenting only new or updated URLs to the user. This will provide a personal search assistant/information filtering capability.

References

[Arens et al. 1993] Arens, Y., Chee, C., Hsu, C., Knoblock, A.: "Retrieving and Integrating Data From Multiple Information Sources"; Journal on Intelligent and Cooperative Information Systems, 2, 2 (1993), 127-158.

[Balabanovic et al. 1995] Balabanovic, M., Shoham, Y., Yun, Y.: "An Adaptive Agent for Automated Web Browsing"; Journal of Image Representation and Visual Communication 6, 4 (1995).

[Callan et al. 1995] Callan, J., Zhihong L., Croft, W.B.: "Searching Distributed Collections With Inference Networks"; 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, WA (1995), 21-28.

[Cross 1996] Cross, W.: All-in-One Home Page, URL: <http://www.albany.net/allinone/>, (1996).

[DEC 1996] Digital Equipment Corporation, Alta Vista Home Page, URL: <http://altavista.digital.com/>, (1996).

[Dreilinger 96] Dreilinger, D.: SavvySearch Home Page, URL: <http://www.cs.colostate.edu/~dreiling/smartform.html>, (1996).

[Excite 1996] Excite Inc., Excite Home Page, URL: <http://www.excite.com/>, (1996).

[Gauch 1996] Gauch, S.: ProFusion Home Page, URL: <http://www.designlab.ukans.edu/ProFusion.html>, (1996).

[GNN 1996] Global Network Navigator Inc., WebCrawler Home Page, URL: <http://www.webcrawler.com/>, (1996).

[InfoSeek] InfoSeek Corp., InfoSeek Home Page, URL: <http://www.infoseek.com/>, (1996).

[Knoblock et al. 1994] Knoblock, A., Arens, Y., Hsu, C.: "Cooperating Agents for Information Retrieval"; Proceedings of the Second International Conference on Cooperative Information Systems, Toronto, Canada (1994).

[Lycos 1996] Lycos Inc., Lycos Home Page, URL: <http://www.lycos.com/>, (1996).

[Open Text 1996] Open Text Inc., Open Text Home Page, URL: <http://www.opentext.com/omw/f-omw.html>, (1996).

[Selberg and Etzioni 1995] Selberg, E., Etzioni, O.: "Multi-Service Search and Comparison Using the MetaCrawler"; WWW4 Conference, Boston, MA (1995).

[Selberg and Etzioni 1996] MetaCrawler Home Page, URL: <http://www.cs.washington.edu:8080/>, (1996).

[Sun 1996] Sun Microsystems, Inc., Multithreaded Query Page, URL: <http://www.sun.com/cgi-bin/show?search/mtquery/index.body>, (1996).

[Voorhees et al. 1994] Voorhees, E.M., Gupta, N.K. and Johnson-Laird, B.: "The Collection Fusion Problem"; in The Third Text REtrieval Conference (TREC-3), Gaithersburg, MD (1994), 95-104.

[WR 1996] Washington Researchers Ltd., Washington Researcher's Search Engines Reference Page, URL: <http://www.researchers.com/pub/busintel/searcheng.html>, (1996).

Acknowledgments

This work was funded by the University of Kansas General Research Fund and National Science Foundation Awards CDA-9401021 and IRI-9409263.