The University of Sydney

Proceedings of ADCS '98

Third Australian Document Computing Symposium

TECHNICAL REPORT 518
20 August 1998

Editors

Judy Kay

Basser Department of Computer Science
University of Sydney

and

Maria Milosavljevic

CSIRO
Mathematical and Information Services

BASSER DEPARTMENT OF COMPUTER SCIENCE

University of Sydney
NSW 2006
AUSTRALIA

Message from the Programme Chairs

Welcome to the ADCS'98, the third Australian Document Computing Symposium. The ADCS series of symposia provide an opportunity for researchers and practitioners in document management and information retrieval to meet and present their work. The symposium aims to cover all aspects of Document Computing - issues ranging from the fundamentals of document architectures and standards for markup, through storage, management, retrieval, authentication and workflow, to active and virtual documents.

This symposium follows the two previous symposia held in Melbourne. We have slightly altered the format of the symposium to include demonstration sessions as well as full and short papers.

All papers submitted for this symposium were submitted as complete papers and these were reviewed by three reviewers. The final programme has nine full papers, three short papers and three demonstrations.

We wish to thank the reviewers who worked to a tight schedule in reviewing the papers, giving both an assessment of each paper and helpful feedback to the authors. They are Peter Bruza, Queensland University of Technology, Stephen Green, Macquarie University, John Shepherd, University of New South Wales, Paul Thistlewaite, Australian National University, Wilson Wen, Telstra Research Labs, Ross Wilkinson, CSIRO Mathematical and Information Sciences and Justin Zobel, Royal Melbourne Institute of Technology.

Our thanks to our Organising Committee, Nick Carr, Linda Romer and Nghi Cao Allette Systems, who ensured that many details of running the symposium were managed smoothly.

Programme Chairs,

Judy Kay, University of Sydney
Maria Milosavljevic, CSIRO Mathematical and Information Sciences.

Proceedings of ADCS'98
Third Australian Document Computing Symposium

Friday 21st August 1998
University of Sydney, Sydney, Australia

CONTENTS

A Proximity Measure for Ranked Text Retrieval

Arkadi Kosmynin

Research Data Network Co-operative Research Centre &
CSIRO Mathematical and Information Sciences
723 Swanston St, Carlton, Vic 3053 Australia

Arkadi.Kosmynin@vic.cmis.csiro.au

Abstract

In this paper we introduce a simple heuristic measure that gives higher scores to the documents where query terms co-occur in close proximity. This measure is aimed to increase performance of text retrieval by distinguishing dense regions of matching from a few matches scattered across a document. The ability to do this is important for large collections where document sizes vary significantly. We briefly discuss a few other techniques that make use of proximity information, then introduce our method and present results of its evaluation. This evaluation shows that the method gives a considerable advantage in comparison with the cosine similarity measure. We also have conducted additional experiments to prove that it works well in a combination with an automatic relevance feedback method.

Keywords Document databases, information retrieval.

1 Introduction

Ranked text retrieval methods [1] have received wide recognition in commercial and research systems. As the focus in information retrieval shifted from experiments with small collections of documents titles, abstracts and relatively short news articles to experiments with very large collections [2] of documents of various sizes, the need to take proximity into consideration became more apparent. Use of terms proximity information is one of the ways to improve performance of text retrieval systems on real life collections and therefore it is a promising direction of research.

A long document may contain many terms from the query under consideration. These terms may be randomly distributed in the document or they may be concentrated in a few paragraphs. In the first case the document is most likely irrelevant to the query. In the second case it is most likely relevant or, at least, a part of it is, and a good algorithm should distinguish these cases. There is strong evidence obtained experimentally that long documents may not be regarded as homogeneous objects [3].

There are working methods, like phrase search [4, 5, 6, 7] and passage retrieval [8, 9, 10], utilizing the proximity information. These methods are widely used in commercial systems [11, 12] and research prototypes [13, 14, 15, 16]. The disadvantage of most of these methods is that they are largely boolean. That means that there is no continual numeric value for the proximity component of the score. It is either "yes" if the terms appear close enough in the document, or "no" if they do not. This makes it hard to use these methods for ranked text retrieval.

Phrase search is a method that allows to identify phrases and use them for indexing and search. "The objective of phrase indexing is to identify groups of words that will enhance retrieval effectiveness when assigned as phrase descriptors to representations of documents and queries" [5]. Phrases can be identified using statistical or syntactic methods. Syntactic

Proceedings of the Third Australian Document Computing Symposium, Sydney, Australia, August 21, 1998.

methods use linguistic evidence to identify phrases. Words in text have to match certain (preset) patterns for phrases to be found. One of the limitations of syntactic methods is that they can only identify phrases for which a grammar has been given.

Statistical methods select words for phrases based on their co-occurrence in texts. All found phrases are kept in an automatically compiled thesaurus which is used to identify phrases in texts or queries.

In both of these methods, to be recognized as a phrase, words have to occur together within some unit of text. For syntactic methods this unit is determined by the patterns, for statistical methods this unit is a parameter. It may be a text, a paragraph, a sentence or a passage of text. This requirement is boolean by its nature: if two words are found within a range of, say, N positions from each other, they are recognized as a phrase; if there are $N+1$ positions between them, they are not a phrase any more and their contribution to the similarity score is 0.

Passage retrieval "provides convenient units of text to return to the user, avoids the difficulties of comparing documents of different length, and enables identification of short blocks of relevant material amongst otherwise irrelevant text" [8]. It is a step down the path of relaxing proximity requirements, as it just requires as many query terms as possible to occur within a single passage. This method splits documents into passages, ranks the passages according to their similarity to the query, and then assigns similarity scores to the documents based on the scores of a few of their top ranked passages. The method has been reported to produce considerable improvement in performance [8, 9, 10]. However, it has its share of problems. First, the similarity scores depend on the size of passages and the way the documents are divided into passages. There are solutions implemented in research systems, but they are not practical for real life applications. The second problem is in the way the similarity scores of passages are combined into the scores of documents. Is it better to rank documents higher if they have few, but highly relevant passages, or if they have many passages of lower relevance? A good description of various approaches to passage retrieval and their problems can be found in [8].

The essential difference between boolean and ranked queries is that boolean queries specify exactly which words must and which must not be in the documents to find. The result of a boolean query is an unordered list of documents. Ranked queries attempt to find documents that are "as similar as possible" to the query according to some similarity measure. They result in ranked lists of documents. To fit into ranked text retrieval, the use of a proximity measure in the resulting score should be fuzzy enough to benefit documents where query terms occur "as close as possible". It should not drop out the documents where the query terms do not occur close than some threshold, it should assign lower scores to such documents.

In the following sections of this paper we introduce a heuristic measure designed to transparently work in a fuzzy, non-boolean way and we describe experiments conducted to evaluate and optimize the measure and discuss the obtained results.

2 A Measure of Proximity

2.1 Quantifying Proximity

In our method, for every <document, query> pair, we build a reduced document vector. To construct the vector of a <document, query> pair, we take the words of the document appearing within a given range (we call this range "frame size" - FS) of the query terms in the document. The query terms in the document are also included. If the occurrence of the query terms in the document is accidental, they will tend to be spread and surrounded by random terms. This will result in a relatively long vector consisting of random terms with low weights and a low proportion of the query terms. If the document is relevant, the query terms will tend to appear together and be surrounded by similar terms (context terms) and the frames of the query terms will overlap. It will result in a shorter vector consisting of (context) terms with higher weights and a higher proportion of the query terms. We call such reduced document vectors "document context vectors" because they include terms of local context of the query terms in the document. Document context vectors can be used instead of the original document vectors in calculation of similarity scores. We will address this method

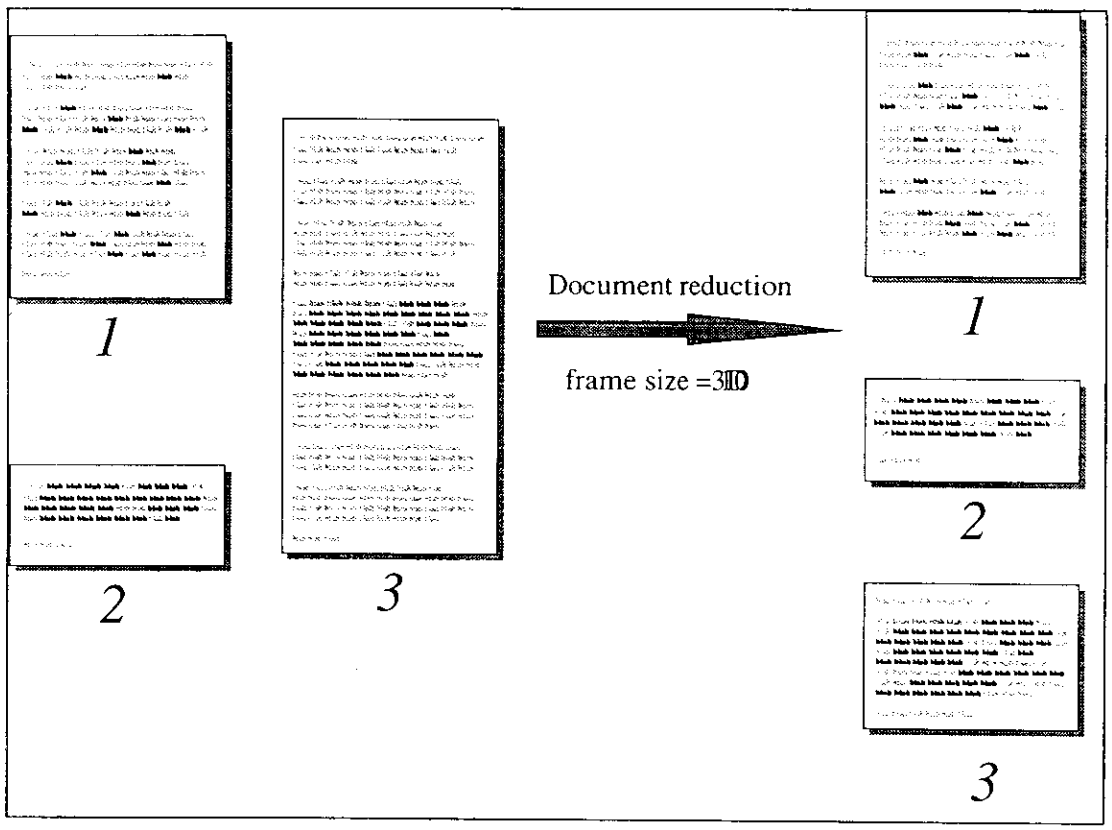


Figure 1: Example of document reduction

(and measure) below as method of Document-Query Context Vectors, or DQCV.

The process of reduction is illustrated in Figure 1. Bold parts of text are query terms. Document 1 is irrelevant to the query, but contains accidental query terms. Document 2 is relevant and short. Document 3 contains a relevant part, but is considerably longer than the other two and therefore would be assigned the lowest score of these three documents. However, the 3rd document will not be penalized for its length after document reduction. It may well be the case that the documents would be ranked in order 2,3,1.

The frame size is an important parameter for this method. With frame size 1, only the query terms would be included in the context vector. With increasing frame size, more terms are included and more frames are overlapping. Overlapping of frames takes place when two or more query terms occur close enough for their frames to overlap. Overlapping leads to fewer non-query terms included in the vector per same number of query terms included, thus, producing a shorter vector and a higher score (as a result of length normalization). With increasing frame size, the effect from overlapping becomes more apparent and thus, documents with query terms occurring in close groups receive more of an advantage. On the other hand, with increasing frame size, more documents with accidentally close query terms receive higher scores. We found that FS=3D gives the best results in most cases.

The following example illustrates our way of including a proximity component in the documents scores. Suppose we have a query consisting of two words: "**red fox**". Suppose we have two one sentence documents, nineteen words each, and we build their context vectors with FS=3D

1) Document "Hunters often use red flags to direct scared wolfs, foxes and some other wild animals into a trap or ambush" gives context vector (often, use, red, flags, to, scared, wolfs, foxes, and, some).
2) Document "A new Web searching tool announced by Red Fox Software

Corporation supports off line search based on user profile information" gives context vector (announced, by, red, fox, software, corporation).

Let weights of all words be 1 for simplicity. The classic cosine measure would rank the documents equally. However, if we use the same cosine measure to rank the context vectors, the scores will be $2/\sqrt{10}$ and $2/\sqrt{6}$ respectively. Thus, the method scores higher the documents where query terms appear closer and it does this in a fuzzy, non-boolean, way.

To build a DQCV for a <query, document> pair, we scan the document word by word. When we meet a query term, say in position i of the document, we take all the words in positions j, $max(\ i\text{-}FS,\ 1\ ) \le j \le min(\ L,\ i\text{+}FS\ )$, where L - number of words in the document. We add each of these words to the DQCV if we have not done it before (some words may occur in overlapping frames of several query terms). This way of building DQCVs is acceptable for a research system, but it is certainly not practical. In a practical system, the positional information from the index should be used to optimize the construction of DQCVs.

2.2 Emphasizing Important Query Terms

Words in queries are not equally important, especially in long natural language queries. We would like our method to give priority to the most important words in queries. So, to favor occurrences of important query terms, we give words from frames of important query terms greater weight when including them in the context vector. For every position i of the original document, the weight of term t_i in this position is increased in the context vector by the logarithm of the sum of the query weights of all query terms occurring within the frame of t_i.

$$dw_c(t_i) = \log\left(1 + \sum_{j \in frame(i)} w_q(t_j)\right) \qquad (1)$$

where $w_q(t_j)$ - query weight of term in position j of the original document.

3 Evaluation

3.1 Collections and Query Sets

For our experiments we used two TREC data sets. The first data set (D2N4) was built from texts available on the TREC CDs 2 and 4. It includes about 524000 documents, total size of the data is 1.7Gb. The second data set (D4N5) was built from texts available on the TREC CDs 4 and 5. It includes about 556000 documents, total size of data is 1.8Gb.

For queries, we used TREC topics 251-300 and 301-350. TREC test topics include a numeric identifier, a *domain* field, a *title* field, a *description* field and a *narrative* field. *Title* is a small group of essential words identifying the topic. For example, "International Terrorism" or "Hydroponics". *Description* is a more detailed explanation of the topic, and *narrative* adds even more details to the explanation.

By combining these fields or their parts, one can generate different queries from the same topic. Table 1 shows what parts of the topics we used to generate our sets of queries. For example, for query set q3td *title* and *description* fields of topics 301-350 were combined.

This process gave us 400 queries (50 in each set). We ran them on the two data sets described above. Queries of the sets q2tdn, q2t, q2d and q2td were ran on the data set D2N4, queries of the sets q3tdn, q3t, q3d and q3td were ran on the data set D4N5.

	Title, description and narrative	Title	Description	Title and description
Topics 251-300	q2tdn	q2t	q2d	q2td
Topics 301-350	q3tdn	q3t	q3d	q3td

Table 1: Used query sets and the topics from which they are derived

3.2 Comparison with Cosine Measure Ranking

We used two variations of the cosine similarity measure for the base runs. They weight terms differently.

$$cosine(Q,D) = \frac{1}{W^q W^d} \sum_{t=1}^{n} w_{q,t} \cdot w_{d,t} \quad (2)$$

$$W^d = \left(\sum_{t=1}^{n} w_{d,t}^2 \right)^{1/2} \quad (3)$$

$$W^q = \left(\sum_{t=1}^{n} w_{q,t}^2 \right)^{1/2} \quad (4)$$

weighting 1: $$w_{d,t} = f_{d,t} \cdot \log \frac{N+1}{f_t} \quad (5)$$

$$w_{q,t} = f_{q,t} \cdot \log \frac{N+1}{f_t}$$

weighting 2: $$w_{d,t} = 1 + \log(f_{d,t} + 1) \quad (6)$$

$$w_{q,t} = 1 + \log(f_{q,t} + 1) \cdot \log(N / f_t + 1)$$

Q - query;
D - document;
N- number of documents in the collection;

f_t - number of documents in which term t occurs (collection frequency of term t);

$f_{d,t}$ - frequency of term t in the document.

$f_{q,t}$ - frequency of term t in the query.

We tried both of these variations for base and DQCV runs. We chose the variation 2 (weighting formulae 6 above) for the base runs because it gave consistently better performance. In the DQCV computation of similarity scores, however, the best choice to use was variation 1 (weighting formulae 5 above).

Table 2 compares results of the base run with DQCV runs. DQCV run 1 was performed with FS =3E with no differentiation between query terms, i.e., documents terms were added to context vectors with weight 1 every time they occurred in a query term frame, or in overlapping frames of several query terms (we call it "flat weighting scheme"). The results in table 2 show that performance decreased only for two query sets (the longest queries), and average gain in average precision was 16.92%, compared to the base run.

The DQCV run 2 (see Table 2) was performed to test the heuristics that takes into account the relative importance of query terms. We used the weighting scheme described in section 2.2. No query sets showed a decrease of performance compared to the DQCV run 1. The average gain in precision achieved by the improvement of the weighting scheme was 13.69%. To calculate this gain, we took the difference between the precision of DQCV runs 2 and 1, and related it to the precision of the base run.

The total gain from using the DQCV measure was 30.62% on average, ranging from -4.14% to 53.35%.

3.3 Improving Results with Automatic Feedback

We have conducted some experiments with an automatic feedback method to investigate whether the DQCV technique can be used complementarily to it. We experimented with

Query set	Data set	Base run (av. pr., %)	DQCV run 1 (av. pr., %)	Gain 1 %	DQCV run 2 (av. pr., %)	Gain 2 %	Total gain, %
q2tdn	D2N4	**15.19**	12.79	-15.79	**14.56**	11.65	**-4.14**
q2d	D2N4	**9.20**	11.08	20.43	**11.51**	4.67	**25.10**
q2td	D2N4	**9.84**	11.33	15.14	**12.23**	9.14	**24.28**
q2t	D2N4	**7.31**	10.61	45.14	**11.21**	8.20	**53.35**
q3tdn	D4N5	**16.01**	13.94	-12.92	**18.75**	30.04	**17.11**
q3d	D4N5	**10.60**	13.68	29.05	**15.52**	17.35	**46.41**
q3td	D4N5	**13.36**	17.10	27.99	**19.71**	19.53	**47.52**
q3t	D4N5	**12.59**	15.91	26.37	**17.04**	8.97	**35.34**
Average		**11.76**	**13.30**	**16.92**	**15.06**	**13.69**	**30.62**

Table 2: Results of the base runs versus results of the DQCV runs.

Query set	Data set	Base run + feedback (av. pr., %)	Gain/ base run %	DQCV run + feedback (av. pr., %)	Gain / DQCV run 2 %	Gain / base run + feedback, %	Gain / base run %
q2tdn	D2N4	**18.51**	21.85	**16.33**	12.15	-11.77	7.50
q2d	D2N4	**12.06**	31.08	**12.74**	10.68	5.63	38.47
q2td	D2N4	**13.70**	39.22	**14.41**	17.82	5.18	46.44
q2t	D2N4	**10.04**	37.34	**12.28**	9.54	22.31	67.98
q3tdn	D4N5	**19.63**	22.61	**20.41**	8.85	3.97	27.47
q3d	D4N5	**12.01**	13.30	**16.42**	5.79	36.71	54.90
q3td	D4N5	**14.90**	11.52	**21.82**	10.70	46.44	63.32
q3t	D4N5	**12.30**	-2.30	**18.09**	6.16	47.07	43.68
Average		**14.14**	**21.83**	**16.56**	**10.21**	**19.44**	**43.72**

Table 3: Results of the feedback runs.

query expansion based on Local Context Analysis (LCA) [17]. LCA uses terms from the local context of the query terms in the top ranked documents for query expansion.

First, LCA runs the original query to get the n top ranked documents. Next, the terms from the local context (passages) of the query terms in these documents are extracted, ranked according to their co-occurrence, re-weighted and the top m of them are added to the original query.

We used the idea of LCA, with a simplistic implementation. Instead of using passages as local context, we used frames. We built DQCVs for the 4 top ranked documents using the flat weighting scheme with FS=30. Than we merged the DQCVs, re-ordered the terms and added the top 70 terms to the query with weight

$$w_i = 0.25 \cdot (1.0 - 0.9 \cdot (i - 1) / 70) \qquad (7)$$

We applied this technique to improve the results of the base run, as well as the results of the DQCV 2 run. Table 3 presents results of query expansion combined with the basic measure we used, and query expansion combined with the DQCV measure. The results show that our method works well with an automatic feedback technique. A query expansion allowed us to obtain considerably better results. The combination of DQCV and feedback outperformed combination of the basic cosine measure and feedback in all but one cases and by 19.44% on average.

4 Conclusions

This work shows that the DQCV method performs well for ranked text retrieval.

The experiments conducted showed an average improvement of 30% over the base method performance and a 19% improvement over the base method when both the DQCV and the base method were used with automatic feedback.

It can be noted that this method works well for shorter queries (q2t, q3t) which is important in the WWW context (the average length of queries submitted to the WWW search engines is between 1 and 2 words).

A possible reason why this method is performing better with short queries is that, due to small amount of words, they do not identify relevant documents as well as longer queries. Therefore, there is a room for improvement. On the other hand, long queries have more accidental words that may significantly reduce quality of produced DQCVs and thus, precision of the method.

Clearly, one of the disadvantages of the method is its substantial run-time cost that includes building DQCVs for every query. To make the method practical for very large collections, advanced optimization techniques must be developed.

We also believe that performance of the DQCV measure can be further improved by tuning the impact of co-occurrence of terms and by applying better techniques for identifying and weighting the important terms in queries.

5 Acknowledgments

The work reported in this paper has been funded in part by the Co-operative Research Centres Program through the Department of the Prime Minister and Cabinet of the Commonwealth Government of Australia.

References

[1] G. Salton. *Automatic text processing: the transformation, analysis and retrieval of information by computer.* Addison-Wesley, 1989.

[2] D. K. Harman. Overview of the First Text Retrieval Conference. *In D. K. Harman, editor, Proceedings TREC Text Retrieval Conference,* pages 1-20, Gaithersburg, Maryland, November 1992.

[3] R. Wilkinson. Effective retrieval of structured documents. *In Proceedings, SIGIR 94*, pages 311-317, Dublin, Ireland, 1994.

[4] W. B. Croft, H. R. Turtle and D. D. Lewis. The Use of Phrases and Structured Queries in Information Retrieval. *In Proceedings, SIGIR 91*, pages 32-45, Chicago, 1991.

[5] J. L . Fagan. Automatic Phrase Indexing for Document Retrieval: An Examination of Syntactic and Non-Syntactic Methods. *In Proceedings, SIGIR 87*, pages 91-101, New Orleans, Lousiana, 1987.

[6] D. D. Lewis, W. B. Croft. Term Clustering of Syntactic Phrases. *In Proceedings, SIGIR 90*, pages 385-404, Brussels, 1990.

[7] E. M. Keen. Term Position Ranking: Some New Test Results. *In Proceedings, SIGIR 92*, pages 66-76, Copenhagen, 1992.

[8] M. Kaszkiel, and J. Zobel. Passage Retrieval Revisited. *In Proceedings, SIGIR 97*, pages 178-185, Philadelphia, 1993.

[9] G. Salton, J. Allan, and C. Buckley. Approaches to passage retrieval in information systems. *In Proceedings, SIGIR 93*, pages 49-58, New York, 1993.

[10] J. P. Callan. Passage-level evidence in document retrieval. *In Proceedings, SIGIR 94*, pages 302-309, Dublin, Ireland, 1994.

[11] Alta-Vista internet search engine. http://www.altavista.digital.com/

[12] Dejanews usenet search engine. http://www.dejanews.com/

[13] C. Buckley, M. Mitra, J. Walz and C. Cardie. Using Clustering and SuperConcepts Within SMART: TREC 6. *In D. K. Harman, editor, Notebook Papers of the 6th Text REtrieval Coinference*, pages 1-15, National Institute of Standards and Technology, Gaithersburg, Maryland, 1997.

[14] J. Allan, J. Callan, W. B. Croft, L. Ballesteros, D. Byrd, R. Swan and J. Xu. INQUERY does battle with TREC-6. *In D. K. Harman, editor, Notebook Papers of the 6th Text REtrieval Coinference*, pages 61-85, National Institute of Standards and Technology, Gaithersburg, Maryland, 1997.

[15] J. O. Pedersen, C. Silverstain, C. C. Vogt. Verity at TREC-6: Out-of-the-Box and Beyond. *In D. K. Harman, editor, Notebook Papers of the 6th Text REtrieval Coinference*, pages 127-141, National Institute of Standards and Technology, Gaithersburg, Maryland, 1997.

[16] D. Hawking, P. Thistlewaite and N. Craswell. NAU/ACSys TREC-6 Experiments (Draft for Notebook). *In D. K. Harman, editor, Notebook Papers of the 6th Text REtrieval Conference*, pages 155-166, National Institute of Standards and Technology, Gaithersburg, Maryland, 1997.

[17] J. Xu and W. B. Croft. Query expansion using local and global document analysis. *In Proceedings, SIGIR 96*, pages 4-11, Zurich, 1996.

Conflation-based Comparison of Stemming Algorithms

Michael Fuller Justin Zobel
Department of Computer Science, RMIT
GPO 2476V, Melbourne 3001, Australia
{msf,jz}@cs.rmit.edu.au

Abstract

In text database systems, query terms are stemmed to allow them to be conflated with variant forms of the same word. On the one hand, stemming allows the query mechanism to find documents that would otherwise not contain matches to the query terms; on the other hand, automatic stemming is prone to error, and can lead to retrieval of inappropriate documents. In this paper we investigate several stemming algorithms, measuring their ability to correctly conflate terms from a large text collection. We show that stemming is indeed worthwhile, but that each of the stemming algorithms we consider has distinct advantages and disadvantages; choice of stemming algorithm affects the behaviour of the retrieval mechanism.

Keywords: information retrieval, document databases, digital libraries, word disambiguation.

1 Introduction

Queries to text database systems consist of words to be matched to the content of stored text [15]. A query can be Boolean, where the words are connected by Boolean operators and retrieved documents must satisfy the given condition; or ranked, where the documents with the greatest statistical similarity to the query are retrieved as answers. For both kinds of query there must be a mechanism for deciding whether a given query term corresponds to a given word in a document. The simplest correspondence method is to allow exact matching only; for example, "trains" would match itself only, and a document that contained "Trains" but not "trains" would not be recognised as a match. Another simple correspondence method is case-insensitive exact match.

A more general form of correspondence is known as *stemming*, in which query terms and document terms are reduced to a common root word by removal of affixes. Terms match if they have a common root. For example, "trains" might be stemmed to "train", which would match "train", "training", and of course "trains". Each word thus has a set of *conflations*, that is, words that have the same root. Several stemming algorithms have been proposed [8, 9, 10, 11, 13, 17, 18], based on different principles; each produces rather different sets of conflations.

Stemming has usually been measured by its impact on querying: since stemming changes the documents that are retrieved in response to a query, it has the potential to change the quality of the set of answers. Despite the fact that stemming—and choice of stemmer—can dramatically change the number of documents that match a query term, the results of information retrieval experiments have been mixed, ranging from negligible difference in average performance to definite but small improvements [5, 7, 10]. However, measuring average performance can mask significant changes in individual queries, and the metrics used for evaluating performance can allow the effects of stemming to seem small, even if the documents sets retrieved are changed significantly.

Instead of using variations in recall and precision as an indirect measure of stemming efficacy, it is possible to evaluate stemming algorithms by the accuracy of the conflations they produce. This was the approach of Paice [12] and of the work presented in this paper: to evaluate stemming, not by its impact on query evaluation, but by direct consideration of the set of terms that are determined by stemming to be equivalent. We investigate three well-known stemming methods, the S stemmer, Lovins's algorithm, and Porter's algorithm, and compare them to a new dictionary-based method that we have developed. Most existing algorithms use word-formation rules and a small lexicon of common prefixes and suffixes to reduce words to probable roots; our method is based on a large public-domain dictionary of words and affixes.

We show experimentally that these methods have rather different behaviour. Using the TREC

Proceedings of the Third Australian Document Computing Symposium, Sydney, Australia, August 21, 1998.

text database and queries, we show that Lovins's method and Porter's method are fairly error-prone, while the dictionary method does not discover as many correct conflations. However, our results indicate that for typical text retrieval applications stemming should be used, and that suffix removal with the Porter algorithm is unlikely to be significantly bettered.

2 Stemming and document retrieval

The task of query evaluation is to find the documents that match a query. For Boolean querying, matching is defined as a logical condition composed of query terms and Boolean connectives. For ranked querying, each document in the collection is scored against the query, and the documents with the highest scores are fetched as matches. As these querying methods only approximate the user's actual information need, they are measured by their ability to fetch relevant documents, that is, documents that a user has manually judged to be relevant.

Two metrics are commonly used to measure retrieval systems: recall, the proportion of the set of relevant documents that are retrieved; and precision, the proportion of the retrieved documents that are relevant. These are often combined into a single recall-precision figure, or effectiveness, by averaging precision at different levels of recall. Systems are tested by applying them to a standard test collection (consisting of documents, queries, and manual relevance assessments for each query) and computing an average effectiveness across all queries. A great many different scoring functions for ranking have been proposed [20], and, as initiatives such as the TREC experiments [6] have shown, different systems can be of similar effectiveness, yet retrieve very different sets of answers [19].

Most previous investigations of stemming have focused on the impact on effectiveness of changing the stemming mechanism [5, 7, 9, 10, 18]. Harman [5] evaluated the performance of the Lovins, Porter, and S stemmers on retrieval from the small Medlars, Cranfield, and CACM collections. Her conclusions were overall weakly positive: stemming produces at best a non-significant improvement. She tried three techniques to improve performance: using lower weights for expanded terms; only using stemming to broaden queries if the number of query terms was less than ten; and only stemming low-noise terms. None of these resulted in improved retrieval. Her results indicated that whilst stemming does improve the performance of some queries, it weakens that of others. In contrast to Harman's conclusions, Krovetz reported generally positive results from the use of stemming [10]; these experiments used a different set of test collections to Harman's work. However, results were not consistent across all collections.

Interestingly, evidence from other fields is also mixed. For example, Riloff reported that stemming had a deleterious effect on term classification [14]; whereas Sanderson's work on sense disambiguation [16] recorded that random ambiguation of terms had minimal impact on retrieval and that only very accurate re-disambiguation regained lost performance, suggesting that stemming should at worst have no effect and could potentially result in significant gains.

Intuitively these results are quite surprising. For example, the scoring functions used in ranking consider the frequency of the query terms, and give considerably more weight to terms that are relatively rare. Thus if a rare term becomes conflated with a common term, not only are many more documents allocated a score but the term's weight in each document is also greatly reduced. The true behaviour may be even more complex than this, however: the work of Church suggests that the correlation between terms (and therefore, from a retrieval perspective, the usefulness of conflation) can be observed to depend on both their frequency and their distribution [2].

However, assuming that the conflation is accurate and does indeed introduce additional relevant documents, with the standard measures of effectiveness these behaviours tend to cancel out. Such conflations increase recall but degrade precision, with small net impact on effectiveness. Moreover, the design of some test collections can obscure the effect of stemming: the large number of terms in long queries ameliorates stemming errors, and long relevant documents are likely to contain many of the conflations of each query term, reducing the need for stemming. These effects could well be the cause of the contrasting results of Harman [5] and Krovetz [10].

Nonetheless, there are clearly cases in which stemming is helpful, such as when one form of a word is mapped to another; and cases in which erroneous stemming is unhelpful, such as when a word is incorrectly mapped to a common word. In practice, perhaps only a small sample of queries would be significantly affected by changes in stemming; measurement of the effect of stemming on information retrieval should therefore investigate per-query changes rather than overall impact.

Paice assessed stemming performance against predefined concept groups in samples of words taken from the small CISI test collection and two

other small sources [12]. Treating stemming as the process of creating concept groups of related terms, a similar approach to that taken in this paper, he evaluated the performance of the Lovins, Porter, and the Paice-Husk stemmer. Paice limited his work to reporting on ideal stemming performance, choosing not to consider term frequency in his evaluation. His results confirm that Lovins is less accurate but more aggressive than Porter, and that the Paice-Husk stemmer, as well as being more accurate, is even more aggressive than Lovins.

3 Stemming algorithms

Stemming algorithms proceed by extracting each word from the text, and, considering that word in isolation, reducing it to a probable root word. Current stemming algorithms do not, for example, use context information to determine the correct sense of each word, and indeed such an approach is unlikely to be helpful. While knowledge of the sense of a word can help guide a stemmer—for example, in the context of income the term "living" should not be stemmed, but in the context of biology it should be stemmed to "live"—such cases are fairly rare. Moreover, given that for most words the vast majority of uses are of one sense only, errors due to inaccurate word-sense analysis are likely to overwhelm any gains that might be made by increased accuracy in stemming.

We now consider several stemming methods in detail, including existing methods and our proposed dictionary technique.

Elementary methods. The simplest form of term conflation is case-folding, thus allowing case-insensitive match. Case-folding is likely to be helpful in most cases, but it is worth noting that this elementary approach can cause errors, for example when the title "Major" is conflated with "major", or the acronym "AIDS" with the term "aids". That is, for even the simplest stemming technique there is potential for effectiveness to degrade for some queries.

Another simple method is the S stemmer [5], in which only a few common word endings are removed: "ies", "es", and "s" (with exceptions). Some practical systems use the S stemmer as, although it does not discover many conflations, it is conservative and rarely produces conflations that surprise the user.

Porter's method. The Porter stemmer [13] removes about 60 suffixes in a multi-step approach, successively removing short suffixes without exceptions. Each step results in the removal of a suffix or the transformation of the root.

Lovins's method. The Lovins stemmer [11] uses a longest match algorithm and exception list to remove over 260 different suffixes. It is the most aggressive of the three algorithmic approaches to stemming.

We used public-domain implementations of the S stemmer, the Porter stemmer, and the Lovins stemmer for the experiments described below.

The dictionary method. Stemming based on affix removal is straightforward to implement and requires minimal run-time resources, but has obvious disadvantages. In particular, a stemming operation that is valid in one case is invalid in another—it is valid to conflate "create" and "creation" but not "state" and "station".

To address this problem we have developed a dictionary-based stemmer. Some dictionaries hold affix information, showing the forms that can be derived from a root word. The public-domain spell checking utility `ispell` includes such a dictionary, with, for each entry, a list of legal affixes. To create a list of "legal" conflations, each affix rule was applied to its root to produce a dictionary of conflation–base term pairs. Where a base term was identical to the conflation of another base term, it was replaced by that conflation's base term. The expanded dictionary contains 41,709 distinct terms.

The dictionary-based approach does have some limitations. The `ispell` dictionary was developed for spelling checking, and includes some derivations that correspond to legal spelling but not the same word; for example, it derives the valid English word "stationer" by appending the affix "er" to the unrelated word "station". However, the number of such errors appears to be fairly small, and as we show below it allows much more accurate conflation than do the other stemming methods. Other problems are that, despite the size of the dictionary, some words are not present; for others, the spelling is given explicitly instead of as a derivation from a root, so that we cannot detect the conflation; and for others, directly related word forms have separate entries to allow different derivations. These problems could be addressed in a production system with a dictionary designed for stemming.

Other techniques. In addition to testing Porter and a Porter variant, Krovetz introduced a new dictionary-moderated inflectional stemmer and a new dictionary-moderated derivational stemmer (now known as KSTEM) [10]. The latter technique stemmed by removing derivational endings, with the requirement that resultant terms must themselves be present in the dictionary. Their experiments showed that this technique, which in principle is similar to Porter with restrictions,

yielded small but significant improvements in retrieval effectiveness.

Xu and Croft introduced a novel technique that combines the use of an aggressive initial stemmer (using trigram conflation or Porter) with term co-occurence data to create corpus-specific equivalence classes; they reported similar or improved retrieval performance compared with that when using KSTEM or Porter stemming [18]. Interestingly, this technique was directly applied to a Spanish corpus with a similar outcome, and could potentially be applied to corpora in other languages without requiring significant additional linguistic knowledge.

Last, Jacquemin has described a truncation-based technique for conflating similar phrases [8].

4 Test data

We chose to test the stemmers by examining the conflation sets they produce for a real text database. The database used was the second disk of TREC data [6], which is one gigabyte of text with a vocabulary of 363,553 distinct terms after case folding. Rather than examine the conflations of all terms, we restricted our attention to terms occurring in TREC queries, both to limit the resources required to complete this work and to obtain a realistic view of the effect of stemming: many of the terms occurring in the TREC data would never be used in a query. The queries used were numbers 211–235 and 251–300; after removal of stopwords (common words such as "the" and closed-class words such as "furthermore"), there were 1,093 distinct terms in total.

Using the TREC vocabulary, we applied each stemmer—the S stemmer, Lovins's, and Porter's stemmers, and our dictionary stemmer—to determine a conflation set for each of the query terms. These were then pooled, giving on average 13.5 conflations per term. We then manually judged the conflations, using the criterion that a conflation was correct if it and the query term could reasonably have been derived from the same root word. Overall, there was an average of 7.9 correct conflations per query term. Some of these "correct" conflations may not be appropriate in the context of particular query or document; our judgements were on the basis of the information that would be available to a stemmer, namely the word alone.

For some terms there were only a few conflations. At the other extreme, each of the terms "care", "cure", and "current" had 56 distinct case-folded conflations; those for "cure" are shown in Figure 1, of which 10 were judged correct. Interestingly, many of the false conflations are a consequence of typographic errors.

Stemmer	Attempted	Correct	Missed
No stemming	1.0	1.0	6.9
Perfect	7.9	7.9	0.0
S	1.7	1.7	6.1
Dictionary	4.1	3.8	4.1
Porter	6.3	5.2	2.6
Lovins	11.1	5.9	2.0

Table 1: *Performance of each stemmer, as measured by the number of conflations attempted, the number of these that are correct, and the number of correct conflations missed, averaged over all 1,093 query terms.*

Our manual judgements in effect provide us with an additional, *perfect* stemmer that finds all, and nothing but, correct conflations for the given query terms. Our evaluation of the perfect stemmer and the other stemmers is discussed in the next section.

5 Experiments and results

Our first experiment was a comparison of the impact of each stemmer on retrieval effectiveness. This experiment was inconclusive, with the different stemmers yielding small overall differences in performance, in line with earlier work [5, 10]. However, as Harman also noticed [5], the effect on performance varied from query to query. As a consequence of the results described below we will further explore the impact on effectiveness in ongoing work.

The second experiment was to directly measure the performance of the stemmers by examining their ability to find the correct conflations. There are two dimensions to these results: how many conflations were attempted for each query term, and how many were correct. Results are shown in Table 1; the "no stemming" data shows behaviour if stemming is not performed. As can be seen, failure to stem means that many reasonable conflations are overlooked, and the S stemmer, while highly reliable, only finds 0.7 additional terms per query term, just 22% of those possible. On the other hand, the Lovins stemmer is highly aggressive, but (in comparison to Porter) the 4.8 additional conflations per query term only contain 0.7 additional correct terms; Lovins has an accuracy of 53% while Porter has an accuracy of 83%. The dictionary stemmer is even more accurate, at 93%, but misses many terms found by the Porter and Lovins methods; these three stemmers found 48%, 67%, and 75% of possible conflations, respectively.

cur cura **curable** curate curated curates curation **curative curatives** curator **curators** **cure cured** curency curently **curer cures** curi curia curial curies **curing** curious curiously curity curium curless curly curr currant currants currator curre curren currencies currency currenly current currently currentness curried currier curries curro curry currying curs curse cursed curses cursing cursive curso cursor cury **uncured**

Figure 1: *Distinct conflations for the term "cure" due to the four stemmers. Conflations that were judged correct are shown in bold.*

Note that, since pooling was used to derive the set of possible conflations, the recall figure (proportion of the correct conflations found by each stemmer) may be inflated; other stemmers might find further correct conflations. There is a surprising lack of overlap between the conflations found by each stemmer.

Interestingly, visual inspection of the errors—that is, the incorrect conflations and the correct conflations that were missed—suggested that although the errors were numerous they might be unimportant, since many of them corresponded to rare or unusual forms of words. We thus conducted a third experiment, where we used the collection-wide statistics to count the number of word occurrences that conflated to each query term. For example, the term "train" occurs 4,890 times in the TREC data, "training" occurs 17,808 times, and "trains" occurs 1,368 times, giving a total of 24,066 occurrences for those conflations of "train".

Results are shown in Table 2. As can be seen, a rather different picture emerges. The S stemmer finds about 61% of the correct conflations by frequency, up from the 22% of our first experiment. Similar increases can be observed for each of the other stemmers, with the dictionary stemmer jumping to 84%, Porter 89%, and Lovins 90% of the correct conflations by term frequency. The accuracy of Lovins rises to 86%, while Porter is 97% and the dictionary stemmer is around 94%.

A distinctly new picture has emerged through consideration of term frequency. Even the simplistic S stemmer is surprisingly effective and the coverage of Porter is as good as the more aggressive Lovins. Accuracy is markedly better for Porter and Lovins, to the extent that the dictionary stemmer, by this measure, can be seen to be *less* reliable than Porter. Indeed, Porter's performance leaves little scope for practical improvement.

Stemmer	Attempted	Correct	Missed
No stemming	15,331	15,331	21,290
Perfect	36,621	36,621	0
S	22,492	22,470	14,151
Dictionary	32,540	30,649	5,972
Porter	33,584	32,652	3,969
Lovins	38,680	33,118	3,503

Table 2: *Performance of each stemmer, as measured by the count of word occurrences that match each query term. The reported results are the number of conflations attempted, the number of these that are correct, and the number of correct conflations missed, averaged over all 1,093 query terms.*

6 Conclusions

We have compared the performance of several stemmers on query terms, measured by their ability to find correct conflations and to find nothing but correct conflations, weighted by the in-collection frequency of the terms involved. The success of the Porter stemmer in achieving 97% accuracy at 90% coverage of potential conflations strongly suggests that stemming is worthwhile. Why previous research has been inconclusive is not clear; possible explanations include that stemming may introduce wrong senses of words, that it adversely affects term weighting, or that the recall-precision metric is in this case a poor method of measurement.

From our results, there seems little reason to use any stemmer other than Porter—it is reliable and finds most of the conflations. As noted above however, the dictionary used was not designed for this purpose, and, in practice, both better reliability and coverage could be expected. The work of Xu and Croft [18] and of Krovetz [10] suggests that truncation techniques moderated by reference to a machine-tractable dictionary could result in improved performance for the simple dictionary stemmer used here.

The next step is to assess the impact of stemming on term weighting. Future work will extend the frequency-based results reported here by using term weighting to examine the impact on similarity measure performance, and explore the use of residual IDF to control selective stemming, as suggested by Church [2].

Acknowledgements

This work was supported by the Australian Research Council.

References

[1] Nicholas J. Belkin, A. Desai Narasimhalu and Peter Willet (editors). *Proceedings of the 20th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, Philadelphia, Pennsylvania, U.S.A., 17–31 July 1997. ACM.

[2] Kennneth Ward Church. One term or two? In Fox et al. [4], pages 310–318.

[3] W. Bruce Croft and C. J. van Rijsbergen (editors). *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin University*, Dublin, Ireland, 3-6 July 1994. Springer-Verlag.

[4] Edward A. Fox, Peter Ingwerson and Raya Fidel (editors). *Proceedings of the 18th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, Seattle, Washington, USA, 9-13 July 1995. ACM.

[5] Donna Harman. How effective is suffixing? *Journal of the American Society for Information Science*, Volume 42, Number 1, pages 7–15, 1991.

[6] Donna Harman. Overview of the second text retrieval conference (TREC-2). *Information Processing & Management*, Volume 31, Number 3, pages 271–289, 1995.

[7] David Hull. Stemming algorithms: a case study for detailed evaluation. *Journal of the American Society for Information Science*, Volume 47, Number 1, pages 70–84, 1996.

[8] Christian Jacquemin. Guessing morphology from terms and corpora. In Belkin et al. [1], pages 156–165.

[9] Wessel Kraaij. Viewing stemming as recall enhancement. In Hans-Peter Frei, Donna Harman, Peter Schäuble and Ross Wilkinson (editors), *Proceedings of the 19th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 40–48, Zurich, Switzerland, 18-22 August 1996. ACM.

[10] Robert Krovetz. Viewing morphology as an inference process. In Robert Korfhage, Edie Rasmussen and Peter Willett (editors), *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 191–202, Pittsburg, U.S.A., June 27 – July 1 1993. ACM.

[11] J. B. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computation*, Volume 11, Number 1-2, pages 22–31, 1968.

[12] Chris D. Paice. An evaluation method for stemming algorithms. In Croft and van Rijsbergen [3], pages 42–50.

[13] M. Porter. An algorithm for suffix stripping. *Program*, Volume 14, Number 3, pages 130–137, 1980.

[14] Ellen Riloff. Little words can make a big difference for text classification. In Fox et al. [4], pages 130–136.

[15] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, Reading, MA, 1989.

[16] Mark Sanderson. Word sense disambiguation and information retrieval. In Croft and van Rijsbergen [3], pages 142–151.

[17] Evelyne Tzoukermann, Judith L. Klavans and Christian Jacquemin. Effective use of natural language processing techniques for automatic conflation of multi-word terms: the role of derivational morphology, part of speech tagging, and shallow parsing. In Belkin et al. [1], pages 148–155.

[18] Jinxi Xu and W. Bruce Croft. Corpus-based stemming using co-occurence of word variants. *ACM Transactions on Information Systems*, Volume 16, Number 1, pages 61–81, January 1998.

[19] Justin Zobel. How reliable are large-scale information retrieval experiments? In *Proceedings of the 21st International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 308–315, Melbourne, Australia, August 1998.

[20] Justin Zobel and Alistair Moffat. Exploring the similarity space. *SIGIR Forum*, Volume 32, Number 1, pages 18–34, 1998.

Evaluation of Indexing Methods for Clustering

Mingfang Wu

Department of Computer Science,
Royal Melbourne Institute of Technology
GPO Box 2476V, Melbourne 3001,
Australia
ming@cs.rmit.edu.au

Ross Wilkinson

CSIRO, Division of Mathematical
and Information Science,
723 Swanston St., Carlton VIC 3053,
Australia
Ross.Wilkinson@cmis.csiro.au

Abstract

In order to synthesize a better answer based on the retrieved documents, we are exploring the use of clustering methods. In this paper, we present an evaluation of some popular indexing methods used for term selection and term weighting. The aim of indexing here is to represent documents, to relate documents with similar topics, and distinguish documents with different topics from each other.

Experiments have been conducted to examine how the clustering results are influenced by some index term selection methods, such as the term selection based on the document frequency, and some term weighting methods, such as the inverted document frequency weight, the signal-noise ratio, and the term discrimination value, will influence the result of clustering. Based on our experiments, we recommend the use of the discrimination value weighting method together with a suitable set of indexing terms for the purpose of clustering the retrieved documents.

Keywords Index, Term Selection, Term Weighting, Clustering.

1 Introduction

Clustering methods have attracted attention in efforts to categorise ranked documents and help users explore the answer space. The aim is to reorganize the information from a list of N top ranked documents into k ($k << N$) groups. This may give users a more concise and understandable answer space. In order to cluster a set of ranked documents, we will need to find suitable techniques to represent each document, and then to cluster the documents based on their similarity.

The vector space model [9] provided a way to represent documents. In this model, each document in the document collection is described by a set of unique content terms. Each document is then represented as a term vector in a vector space of dimension M, where M is the number of unique terms in the collection.

Using the vector space model, a document can be represented using a set of weighted terms,

$$D_i = (w_{i1}, w_{i2}, \ldots, w_{iM}) \quad (1)$$

where w_{ij} represents the weight of term j in the vector corresponding to the i th document. Thus, for any two documents, their similarity can be measured by the similarity of their respective term vectors.

There are two major tasks in the process of representing documents using the vector space model: to identify the terms that are capable of representing a document content, and to assign each of these terms a weight that represents the importance of that term in the document. The first task is usually referred to as *term selection*, the second one as *term weighting*, and the process of representing documents is referred to as indexing.

The term selection and the term weighting could be done either manually or automatically. This paper is focused on the automatic methods, i.e. automatic indexing.

The aim of term selection is to find out a set of terms that can act as a content identification of a document. Luhn [5] found that the occurrence frequency of a given term can be used to judge the importance of that term.

Define the term frequency as the occurrence of a given term in a document, and the collection frequency of the term as the sum of the term frequency in all documents of the collection, then, according to Zipf's law, the terms with a medium collection frequency tend to have higher resolving power, while the terms with either a high or low collection frequency tend to be less significant. Therefore, the terms with medium collection frequency are usually regarded as good content identifiers, and can serve as good candidates for index terms. However, there is no simple method to determine suitable thresholds to pick up the useful medium-frequency terms from the whole term set. A practical approach is to remove those high frequency function words (i.e.

Proceedings of the Third Australian Document Computing Symposium, Sydney, Australia, August 21, 1998.

those words appeared in the stop list), and merge the words with equivalent stems.

As content identifiers, selected index terms may have different importance to different documents. A weighting method is usually employed to evaluate the importance of the index terms for each document, and more importantly, to differentiate a document from the rest of the collection. A very basic weighting method is to give each index term a weight directly proportional to its frequency of occurrence in each individual document. This basic weighting method gives preference to terms with a high frequency in the document. But these terms are not necessarily good at discriminating one document from all other documents in the collection. Some relative frequency weighting methods have been proposed as improvement, which identify those terms occurring with substantial frequencies in some individual documents, but only with a relatively low overall collection frequency.

The term selection and term weighting methods should be chosen so that they are relevant to the type of clusters being sought. For the task of organizing retrieved documents under certain topic through clustering, we want to index documents so that documents with similar topics can be related to each other, and documents with different topics will be distinguished from each other. In the rest of this paper, we will present our investigation and experiments on how the term selection and term weighting methods influence the results of clustering.

The rest of this paper is organized as follows. Section 2 will introduce the term selection and weighting methods we tested. Section 3 will describe experiments conducted for the evaluation of different term selection and weighting methods, and Section 4 will summarize the findings as the conclusion.

2 Term Selection and Term Weighting

2.1 Term Selection

Current indexing term selection method for document clustering is to use a whole set of unique terms in the collection, i.e. the union of all (non-stop word) terms derived from the documents. The resulting indexing term set is usually large, which leads to high dimension sparse document vectors, and may affect the effectiveness and efficiency of clustering. This is because any extra dimensions added into the document vector, especially those terms containing no relevant information, may dilute the useful information provided by other terms and make the cluster less apparent.

For the purpose of clustering, index terms are selected not only for representing a document, but also for relating documents. So the terms with very low document frequency may not be helpful in relating one document to another, and the terms of very high document frequency, such as those terms appeared in almost every document of the collection, do not contribute much to distinguishing documents on different topics.

The selection of a right set of terms for clustering the top ranked retrieved documents is a difficult task, because those top ranked documents are similar to each other in one way or another. Some query terms tend to appearing frequently in the dataset. For example, if we issue a query "As a result of DNA testing, are more defendants being absolved or convicted of crimes?", it may get other documents containing some of the query terms but are about different topics, such as "As the result of polygraph test, are more defendants being convicted or absolved of crimes?", or "Blood tests and defendants being convicted or absolved of crimes for drunk driving" [14]. In this case, the words "result, test, crime" tend to be frequent words in top ranked documents, but these words are not helpful to distinquish documents of different topics. These non-informative terms could be removed.

Salton, Yang and Yu [7] found that the best document space for retrieval purpose is one which maximizes the average separation between documents in the document space. Their experiments showed that the document frequency and the goodness of a term as index term are well correlated. If N represents the number of documents in a collection, df_j is the document frequency, i.e. the number of documents in which term j appears, then terms may be classified based on the document frequency: high frequency terms, $df_j > N/10$, are poor discriminators; low frequency terms, $df_j < N/100$, are indifferent discrimators; medium frequency terms, $N/10 \leq df_j \leq N/100$, are good discriminators. In this paper, we will investigate on the influences on the clustering result caused by the removing of high or low document frequency terms.

2.2 Term Weighting

After selecting a set of terms, we will need to represent each document from the collection using these terms, i.e., giving each term a weight that reflects its importance in each document. Many term weighting methods have been described in the literature, among them the term frequency is a simple, commonly used measure. In this measure, index terms are weighted in proportional to their term frequency in each document. In this way, this basic measure can relate to the information content, but it may not be able to distinquish one document from other documents in the collection.

Many methods have been proposed in order to give a term a discrimination value, to identify terms occurring with substantial frequency in some individual documents, but with a relatively low overall collection frequency. Inverse document frequency weight, signal-noise ratio, and the term discrimination value are three of the most popular methods.

2.2.1 The Inverse Document Frequency Weight

The inverse document frequency weight method assumes that term importance is proportional to the term frequency (*tf*) and inversely proportional to the document frequency of the term (*idf*). Thus, two indicators are very often multiplied together to form the "$tf \cdot idf$" weight:

$$w_{ij} = tf_{ij} \cdot idf_j \tag{2}$$

There are many formula proposed for calculating tf_{ij} and idf_j [15]. Here we adopted:

$$w_{ij} = tf_{ij} \cdot log(N/df_j + 1) \tag{3}$$

where N is the total number of documents in the collection.

2.2.2 The Signal-Noise Ratio

The signal-noise ratio method suggests the use of information theory considerations to construct a measure of term importance [8, 10]. For a collection of N documents, the noise of term j is defined as:

$$NOISE_j = \sum_{i=1}^{n} \frac{tf_{ij}}{TF_j} log \frac{TF_j}{tf_{ij}} \tag{4}$$

where

$$TF_j = \sum_{i=1}^{n} tf_{ij} \tag{5}$$

The measure of noise shows the "concentration" of a term in the document collection. That is, for the perfect even distributions, when a term occurs in identical number of times in every document of the collection, the noise is maximized. On the other hand, for perfectly concentrated distributions, when a term appears in only one document, the noise is zero.

An inverse function of the noise, SIGNAL, is then defined as follows:

$$SIGNAL_j = log(TF_j) - NOISE_j \tag{6}$$

thus

$$w_{ij} = tf_{ij} \cdot SIGNAL_j \tag{7}$$

2.2.3 The Term Discrimination Value

Salton, Yang and Yu [7] introduced the idea of the discrimination value of indexing term. This model ranks the term in accordance with how well they are able to discriminate a document from each other in the collection; that is, the value of a term depends on how much the average separation between individual documents changes when the given term is assigned for content identification. Let

$$AD = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{k=1,k\neq i}^{N} sim(d_i, d_k) \tag{8}$$

AD represents the average density of the document space. To determine the effect of removing a particular term from the collection, the density of the document space are recomputed. Let d_i^j represent the document vector d_i with term j removed. Then D_i, the density of the document space with term j removed from the collection, is

$$AD_j = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{k=1,k\neq i}^{N} sim(d_i^j, d_k^j) \tag{9}$$

and the discrimination value of term j , DV_j, will be calculated for each term j by the equation

$$DV_j = AD_j - AD \tag{10}$$

and so

$$w_{ij} = tf_{ij} \cdot DV_j \tag{11}$$

The discrimination value involves a lot of calculation. El-Hamdouchi [3] and Crouch [2] proposed some algorithms to simplify the calculation of discrimination value. Their experiments shown that their methods have greatly reduced the time required to calculate discrimination value. Here, we will adopt Crouch's exact centroid algorithm. It was reported that exact centroid algorithm [2] takes less time and is highly compatible with those produced by formula (11).

3 Experiments

The aim of our experiments is the investigation of the influence of the different term selection thresholds and different term weighting methods on the clustering results – we want to examine whether those documents with similar topics could be grouped together.

3.1 Experiment Set-up

We experimented over the TREC5 [4] collection which consists largely of newswire and magazine articles. Associated with this collection is a set of topic descriptions (refer to interchangeably as queries here) with matched relevance assessments. We used 50 queries in our experiments. For each query, the top 300 ranked documents were used to form a document collection for clustering.

We examined the effects on term section caused by different cut-offs (thresholds) at the high and low document frequencies. Table 1 shows a summary of the document frequency distribution. As we can derive from the table, for a collection of 300 documents, there are about 55% of terms with a quite low document frequency range from 2 to 5. However, there is much lower percentage of terms with a relatively high document frequency: only 5% of terms which have a document frequency between 61 to 300, 10% of terms with a document frequency between 31 to 300, and merely 20% of terms within a quite broad range of document frequency between 16 to 300. The term set with a term document frequency between 2 to N (i.e. 2 to 300 in our case) is referred to as the base term set. This set is usually used for clustering the retrieved documents.

Range of document frequency	Average number of terms per collection
[2, N]	4619.52
[N/30, N]	1445.1
[N/50, N]	2102.68
[N/100, N]	3405.16
[N/100, N/5]	3202.2
[N/100, N/10]	2887.72
[N/100, N/15]	2625.48
[N/100, N/20]	2406.66

Table 1: The term sets and their corresponding document frequency ($N = 300$).

3.2 Clustering Algorithm and Evaluation

A suitable clustering method needed to be selected for the experiment. There are a wide variety of document clustering algorithms [1, 12, 9] to choose from. These algorithms can be broadly classified into two categories: the hierarchical and the non-hierarchical clustering methods. Because our purpose of clustering was to group documents with similar topics, owing to the difficulty in exactly judging whether the content of a documents being "about" a certain topic, the hierarchically structuring seemed inappropriate in our case [6]. We decided to use a non-hierarchical single-pass algorithm. The cosine measure was used to calculate the similarity between any two documents, that is:

$$cos(d_i, d_k) = \frac{\sum_{j=1}^{M}(w_{ij} \cdot w_{kj})}{\sqrt{(\sum_{j=1}^{M} w_{ij}^2) \cdot (\sum_{j=1}^{M} w_{kj}^2)}} \quad (12)$$

The evaluation on the effectiveness of the clustering of retrieved documents is limited by lacking of related information. For the collection we used, we have the information about whether a document is relevant to a query, but we do not have the information about what topics the document is about. Current evaluation method on clustering results continues to use the recall/precision [13, 11]. That is, a one dimensional list of documents is first constructed from a list of two dimensional clusters, then, the recall/precision for the constructed list is calculated.

The recall/precision for the constructed list largely depends on the way how the list is ordered. To reflect the real use of clusters, different strategies of reordering the list were proposed [13]. These include: taking all documents from one cluster before another cluster, taking documents from each cluster in turn, and taking documents from each cluster according to a ratio. In this paper, we present the precision of the list which is obtained by taking all documents from one cluster before another, as the precision for this list is the highest among the three strategies in this experiment.

3.3 Experiment Results and Discussion

3.3.1 Clustering Performance

We implemented, tested and compared 3 term weighting methods (cf. formula (3), (7) and (11)) for the 8 term sets (cf. Table 1). For each term set and each term weighting method, we clustered 50 document collections of 300 top ranked documents. Table 2 shows the average precision of 50 constructed lists for each combination of 3 term weighting methods and 8 term sets. From this table, we observed the following findings:

- When the cut-off for lower document frequency terms was increased from N/100 to N/30, the inverse document frequency weight and the signal-noise ratio reached their highest precision point at the cut-off of N/50, while the discrimination value reached its highest precision point at N/100.
- All weighting methods fell to the lowest precision point at N/30, this may be caused by the loss of too many low document frequency terms.
- If the cut-off for lower document frequency was fixed at N/100, and only the cut-off for

Term sets	Weighting Methods	Precision at each cut-off						Average number of clusters
		5	10	15	20	30	50	
[2, N]	Inverse	0.436	0.360	0.307	0.278	0.235	0.199	7.7
	Discrim	0.472	0.368	0.332	0.298	0.267	0.226	10.1
	Signal	0.416	0.340	0.300	0.270	0.237	0.197	5.3
[N/30, N]	Inverse	0.420	0.352	0.308	0.280	0.242	0.204	5.8
	Discrim	0.460	0.366	0.329	0.293	0.263	0.225	5.9
	Signal	0.428	0.340	0.300	0.273	0.239	0.203	5.4
[N/50, N]	Inverse	0.440	0.372	0.320	0.292	0.259	0.224	6.4
	Discrim	0.480	0.366	0.335	0.305	0.259	0.214	8.4
	Signal	0.444	0.340	0.311	0.274	0.240	0.203	5.4
[N/100, N]	Inverse	0.432	0.344	0.301	0.273	0.239	0.203	7.5
	Discrim	0.484	0.374	0.331	0.299	0.269	0.228	7.4
	Signal	0.436	0.350	0.304	0.272	0.238	0.197	5.3
[N/100, N/5]	Inverse	0.476	0.376	0.319	0.291	0.259	0.219	7.9
	Discrim	0.484	0.390	0.343	0.309	0.270	0.234	7.4
	Signal	0.420	0.344	0.301	0.265	0.233	0.199	6.6
[N/100, N/10]	Inverse	0.444	0.362	0.312	0.272	0.233	0.195	8.8
	Discrim	0.460	0.358	0.313	0.283	0.253	0.230	8.5
	Signal	0.380	0.312	0.268	0.232	0.199	0.164	8.1
[N/100, N/15]	Inverse	0.468	0.350	0.319	0.288	0.245	0.202	8.0
	Discrim	0.480	0.374	0.337	0.308	0.277	0.233	8.8
	Signal	0.448	0.366	0.320	0.283	0.235	0.187	10.3
[N/100, N/20]	Inverse	0.480	0.378	0.329	0.300	0.246	0.220	8.2
	Discrim	0.480	0.372	0.332	0.298	0.265	0.216	9.1
	Signal	0.376	0.288	0.245	0.211	0.191	0.170	12.4

Table 2: Evaluation of three term weighting methods for each term set.

high document frequency was changed, the inverse document frequency weight, the discrimination value and the signal-noise ratio will reach their highest precision points at N/20, N/5 and N/15 respectively.

From this table, we can summarize that:

- the discrimination value method can achieve the highest precision for almost all term sets;
- the signal-noise ratio method shown the worst performance;
- the discrimination value for term set [N/100, N/5] is about 11% better than the inverse document frequency weight for term set [2, N]; and 29% better than the worst signal-noise ratio;
- the performance of the discrimination value method is least sensitive to the changes in the term set size. This may be due to the fact that the discrimination value was not directly related to the document frequency.

From this table, we also observed that the average number of clusters per collection increased when the cut-offs for either low or high document frequency terms were decreased. This indicates that the removal of high document frequency terms may cause documents less related to each other, while the removal of low document frequency terms may cause documents less distinguishable from each other.

3.3.2 Efficiency

The efficiency of the clustering is also important, as the clustering method will be used after searching to organize top ranked documents dynamically. Table 3 shows average CPU times and average number of clusters per query for the all combinations of term sets and term weighting methods. (All experiments were run on Sun Sparc-10.) From the table, we can see that discrimination value method takes less time among the three term weighting methods for all term sets.

4 Conclusion and Future Work

We explored the application of term selection and term weighting methods on clustering. The experiments indicated that the discrimination value weighting method performed best for almost all tested term sets.

Different weighting methods achieved their best performance in different term sets. The best weighting method in the best term set was 29% better than the worst.

Term sets	Term weighting methods		
	Inverse	Discrim	Signal
[2, N]	44	25	154
[N/30, N]	25	24	60
[N/50, N]	30	23	80
[N/100, N]	38	22	118
[N/100, N/5]	55	19	41
[N/100, N/10]	79	15	66
[N/100, N/15]	85	12	75
[N/100, N/20]	85	12	77

Table 3: Average CPU times in seconds

Compared with the widely used inverse document frequency weighting methods on the base term set of [2, N], the discrimination value for term set [N/100, N/5] is about 11% better at top 5 documents, and 57% faster.

For the purpose of synthesizing retrieved documents using clustering technology, we recommend the use of the discrimination value weighting method together with an indexing term document frequency from N/100 to N/5.

The results reported in this paper were obtained based on only one dataset, we are planning to work on more datasets and more term sets in order to get a more representative result.

Acknowledgements

The work reported in this paper has been partially funded by the Cooperative Research Centers Program through the Department of the Prime Minister and Cabinet of Australia, and by the Australian Research Council.

References

[1] William B. Frakes and Ricardo Baeza-Yates. *Information Retrieval: Data Structures and Algorithms.* Prentice-Hall, 1992.

[2] Carolyn J. Crouch. An analysis of approximate versus exact discrimination values. *Information Processing and Management*, Volume 24, Number 1, pages 5–16, 1988.

[3] Abdelmoula El-Hamdouchi and Peter Willett. An improved algorithm for the calculation of exact term discrimination values. *Information Processing and Management*, Volume 24, Number 1, pages 17–22, 1988.

[4] D. Harman and E. Vorhees (editors). *Proceedings of the Fifth Text Retrieval Conference*, 500-238 in NIST Special Publication, Gaithersburg, Maryland, 1996. Department of Commerce, National Institute of Standards and Technology.

[5] H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, Volume 2, pages 159–165, 1958.

[6] Daniel E. Rose, Richard Mander, Tim Oren, Dulce B. Ponceleon, Gitta Salomon and Yin Yin Wong. Content awareness in a file system interface: Implementing the 'pile' metaphor for organizing information. In *ACM-SIGIR*, pages 260–269, Pittburgh, PA, USA, 1993. has a review.

[7] G. Salton, C. S. Yang and C. T. Yu. A theory of term importance in automatic text analysis. *Journal of American Society for Information Science*, pages 33–44, January-February 1975.

[8] Gerard Salton. *Dynamic Information and Library Processing.* Prentice-Hall, 1975.

[9] Gerard Salton. *Automatic Text Processing: the transformation, analysis, and retrieval of information by computer.* Addison-Wesley, Reading, MA, 1989.

[10] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[11] Hinrich Schutze and Craig Silverstein. Projections for efficient document clustering. In *Proceedings of the 20th Annual International ACM SIGIR Conference*, pages 74–81, 1997. Using LSI before clustering.

[12] C.J. van Rijsbergen. *Information Retrieval.* Butter-worths, London, second edition edition, 1979.

[13] Mingfang Wu and Ross Wilkinson. Using document relationships for better answers. In *Processing of the Workshop on Principles of Digital Document Processing*, St. Malo, France, March 1998.

[14] Jinxi Xu. *Solving the Word Mismatch Problem through Automatic Text Analysis.* Ph.D. thesis, Department of Computer Science, University of Massachusetts Amherst, 1997.

[15] Justin Zobel and Alistair Moffat. Exploring the similarity space. *To appear in SIGIR'98*, 1998.

Using Natural Language Generation Techniques to Produce Virtual Documents

Robert Dale†, Stephen J Green†, Maria Milosavljevic‡, Cécile Paris‡, Cornelia Verspoor† and Sandra Williams†

†MRI Language Technology Group
Macquarie University
Sydney NSW 2109 Australia
{rdale,sjgreen,kversp,swilliam}@mri.mq.edu.au

‡CSIRO Mathematical and Information Sciences
Locked Bag 17
North Ryde NSW 1670 Australia
{Maria.Milosavljevic,Cecile.Paris}@cmis.csiro.au

Abstract

With the increasing importance of Web publishing, there has been considerable interest in the production of virtual documents on demand. The bulk of this work has used existing documents annotated with meta-data as a source. We suggest that more flexibility and functionality can be obtained if virtual documents are generated instead from raw data. This capability can be achieved by using natural language generation techniques. In this paper, we describe a project concerned with automatically generating natural language descriptions of museum artefacts directly from a museum's Collection Information System.

Keywords Natural Language Generation, Databases

1 Introduction

Natural language generation (NLG) is concerned with the development of techniques for producing linguistic output, whether written or spoken, from some underlying information source. While some of the aims of NLG can be seen as similar to those of the work in document computing and management (e.g., the production of virtual documents on demand), the process and resources it uses are different. In document computing, the emphasis has been on controlling the authoring of a text in order to annotate it with the meta-data which will enable a system to later reason about that text, potentially creating a new text from it (e.g., [9]). Document computing thus deals with *existing* texts, reasoning about its meta-data to form a new text on demand. In contrast, an NLG system reasons about some *information* to be conveyed to the reader, the purpose for which it needs to be conveyed and the intended reader (listener) to construct a text from scratch. It does so by employing *linguistic resources*, which capture principles of communication, information about how to refer to some concept, and how to form sentences and paragraphs. The input to a generation system is not a *text* or *paragraphs*, but some "underlying" information, considered to be of a more conceptual or semantic nature. The linguistic resources then allow for the mapping of this semantic information to a text. Unlike a parser or a translation system, an NLG system does not need to analyse the meaning of a text. Rather, it must reason about how to express information linguistically. By starting from information as opposed to text, and by constructing the text from the underlying information, NLG technology offers a number of important benefits, including:[1]

- description of internal data: given the appropriate linguistic resources, an NLG system can automatically produce text to describe a wide range of internal data. For example, an NLG system can be employed to explain the contents of a database, to produce reports describing numbers (e.g., stock reports, weather reports, etc.), to explain the reasoning of an expert system, or to provide documentation for a program.
- contextual tailoring: the generation process can make use of information only available at the point of use (such as characteristics of the particular reader, or information about

Proceedings of the Third Australian Document Computing Symposium, Sydney, Australia, August 21, 1998.

[1] Of course, it is important to recognise that NLG techniques are only appropriate when underlying information is available. If all that is available are existing texts, the technology may not be applicable.

the content of recent interactions the user has had with the system) to create texts that are tailored to specific requirements. Since the process is not constrained by existing text fragments, the range of texts that can be produced is potentially very large and not producible by other means.

- up-to-date reporting and documentation: if descriptions of the information source are created automatically and dynamically, there is no requirement to update such descriptions manually, with the attendant problems of errors and time lag.
- multilinguality: if the underlying information source is not expressed in terms of a particular natural language, then it is possible to generate descriptions of the same information in different languages automatically.

In NLG, a great deal of research has been carried out to explore the technical requirements that need to be met to provide these capabilities. It is clear from that research that it is possible to construct the appropriate engines and linguistic resources to allow for the automatic production of virtual documents, on demand, providing the benefits mentioned above. Little work, however, has been done to ensure that the input required by a generation system was available (when it is not simply a set of numbers). Indeed, typical NLG systems take as input an AI-style knowledge base, and, in much of the work, proof-of-concept prototypes were built using small knowledge base samples often manually constructed for experimental purposes—see, for example, [1, 3, and 9].[2]

Most digitally encoded information is not, however, available in such richly structured and annotated form. If this technology is to make a significant impact, and if we want to be able to exploit it to produce virtual documents, then we need ways of using it in conjunction with *existing* sources of information. In particular, it must be possible to exploit existing databases. This is the issue we address in our work. In particular, this paper presents some results from an experiments we have been pursuing in using a real database as a source of information for the production of virtual documents on demand. Our particular goal is the automatic description of the contents of a museum Collection Information System (CIS).

This paper is structured as follows. We first briefly describe in Section 2 the architecture of our NLG system, Power, and show how we use NLG to produce virtual documents on the fly. In Section 3, we present our system, PowerTNG, which generates texts from a knowledge source derived completely automatically from a museum's database. Finally, in Section 4, we draw some conclusions.

[2] However, see [6] and [7] which also focus on the automatic acquisition of the knowledge required to produce texts, in order to make language generation technology more practical. In particular, the authors present a system capable of obtaining the knowledge required to generate software documentation directly from the software specifications for that application.

2 Generating virtual documents

Figure 1(a) shows the architecture of a conventional NLG system, whilst figure 1(b) is our architecture for generating virtual documents on the Web. Our system, Power, begins with a *discourse goal*, which captures the purpose of the text to be generated. In our scenario, the discourse goal is a user request either to describe a single museum object or to compare two museum objects. Based on this discourse goal, the system selects from its plan library (which is part of the system's linguistic resources) a discourse plan to employ to satisfy the goal. These discourse plans are based on the notion of discourse schemas introduced by [4], but modified for use in a hypertext environment, by indicating in which circumstances a hyperlink is appropriate, and what discourse goal is to be given to the system when clicking on that link.

After selecting a discourse plan, the text planning component instantiates it with *facts* from the knowledge base. These facts constitute the information which will be contained in the resulting text. For our first experiment, the knowledge base was hand-constructed as is common in other NLG systems.

A user model stores specialised information about particular users. This allows the tailoring of texts to the requirements of individual users. In our current system, the user model is exploited in two ways: to provide two different views of the object hierarchy for naïve and expert users, and to modify how descriptions and comparisons are presented.

A record of the discourse is also maintained for each user, and is exploited in combination with the user model in order to improve the conceptual and textual coherence of descriptions. For example, if the user has knowledge of an entity (as recorded in the user model) or has been told about an entity (as recorded in the discourse history), then the entity can be referred to in later descriptions where a comparison can be made with that entity [5]; Figure 2 shows a comparison between the Difference Engine and the Analytical Engine produced by this mechanism. It is important to note that this text does not exist as text anywhere, but was generated on the fly from the underlying semantic knowledge.

Once the text planning component has pulled together all the information about the entity (or entities) to be described in the document according to the user model and the discourse history,

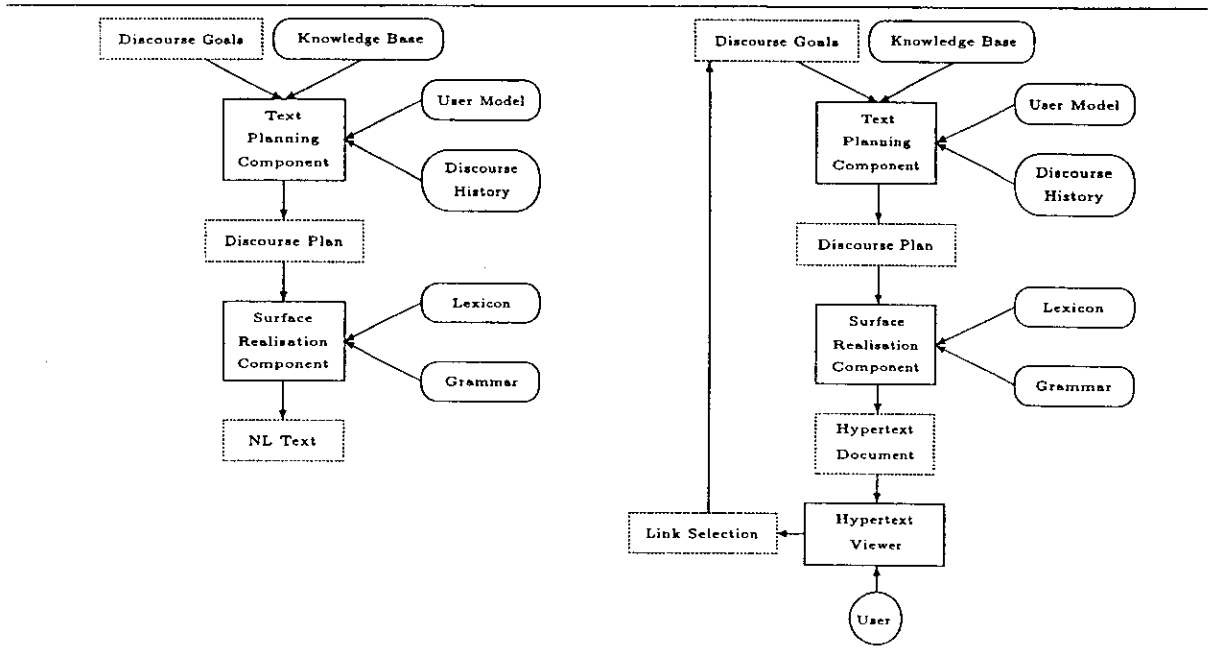


Figure 1: System architectures: (a) traditional NLG; (b) dynamic hypertext.

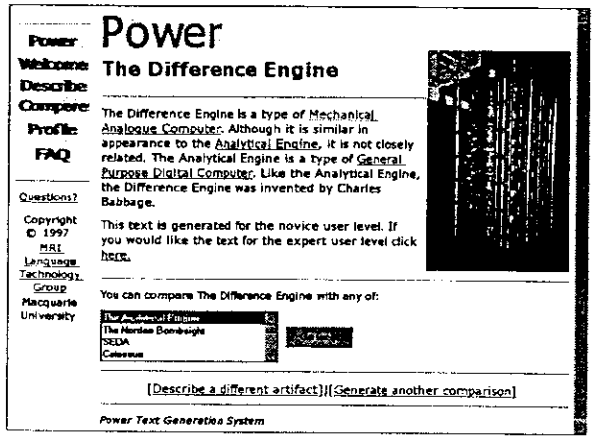


Figure 2: A text generated by Power

the filled discourse plan is passed to the surface realisation component. Here, the discourse plan is realised as natural language sentences, and HTML tags are positioned within the text to allow the user to request follow-up questions by selecting them. When the user selects a hypertext link within the description, a new discourse goal is posted to the text planning component, and the cycle repeats.

We can readily see from this scenario how NLG allows for the production of virtual documents on demand, and the advantages it may offer compared to the creation of new documents from existing ones: by creating the text from raw data according to linguistic principles, and by making use of a user model and discourse history, it can ensure the generation of an appropriate text for the specific user and situation, and the smooth transition from one description to the next. This functionality thus provides a more natural discourse between the user and the system [1, 5]). This idea is the one we explored when building the PEBA-II system, a system which provide interactive dialogues with databases using the Web as a delivery vehicle [1, 2, 5]. Finally, NLG offers the capability of constructing the text in a different language, given the appropriate linguistic resources (e.g., grammar and vocabulary).

Having demonstrated the idea of using NLG to produce virtual documents on the Web, we then examined the issue of using an existing database, as opposed to our carefully hand-crafted knowledge base in which we encoded precisely the kinds of information we needed to generate the texts we were aiming for. Our aim here was two-fold. On the one hand, we want to see the types of texts that can be generated when using a real database. On the other hand, we also want to draw some conclusions as to how the databases should be constructed in order to take advantage of what NLG can offer in terms of the production of virtual documents on demand.

The next section describes PowerTNG, our implementation of the same basic NLG architecture, but this time using the Powerhouse museum's existing database.

```
<rec num=3D12798 id=3D"H4448-513">
OID: H4448-513
INT: Part
LOC: TH2.STEP.6A
LOD: 27/11/1997
OBN: Boots
OBS: Balmoral boots, elastic sided, pair, women's,
patent/kid/leather/elasticised fabric/wood,/brass prize work,
[Gundry & Sons], England, c.1861; 1862-1869. DES: Balmoral boots,
elastic sided, pair, women's, patent / kid / leather /
elasticised fabric / wood /brass, prize work, [Gundry & Sons],
England, c.1861; 1862-1869. Pair of women's elastic sided
boots (Balmoral), with wooden filler, of welted construction with
rounded toes featuring peaked caps and stacked heels. The
uppers consist of a patent golosh, seamed at the back, glace kid
leg, seamed at front and back, and elastic sides extending to the
golosh. The uppers are decorated with oval stitching at the edge
of caps and scallops at the throat of golosh. The leather heel
is fine wheeled, featuring a top piece with brass nailed edge.
The black leather sole features a sueded forepart with brass
nails, as well as an internal clump and brass hinged section for
extra strength and a brown polished ridged waist with black edge.
Reputed to have been made by Gundry & Sons. (See object file for
specialist report by June Swann)
MDE: Gundry & Sons; London, England
MAN: 1965 list says "made by Gundry & Sons, Soho Square." Swann
says hinged device to increase flexibility is unusual. Similar
across on H4448-515. Note hinged sole in 1862 exhibition. She
finds no information about Box in information she has about the
1851 exhibition, though William Walsh is mentioned in connection
with a pair of shoes. Patent 558, 5 March 1861, granted to J.H.
Carter, a similar sole with 2 cuts across the tread and 4 rows
of screws "for soldiers, riflemen, sportsmen. The inner sole is
whole and contains pitch." It is not possible to confirm whether
these boots contain pitch.
DAT: c 1851 - 1869
MAR: Interior obscured by last, no marks on exterior
DIM: Length 248 mm Height 31 mm Overall Height 160 mm Width 58 mm
</rec>
```

Figure 3: A database record

3 Generating from a Real Database

We are concerned with producing descriptions of the Powerhouse artefacts on the Web, starting with their Collection Information System (CIS). The Powerhouse Museum's CIS is a database of the 200,000 objects the museum owns, although for our pilot study we narrowed our focus to the approximately 5,000 objects that are actually on display on the museum floor (as with many museums, most of the collection is in storage). We then supplemented these 5.000 objects with any objects that are part of a display object, and any objects that have a display object as a part. Our current system contains information on 15,483 objects.

Figure 3 shows a typical record from the database. This record contains information about the Balmoral boots.

As this figure shows, a great deal of the information contained in this record is of a *textual* nature. As mentioned in the introduction, an NLG system does not start with text as input, as it is not capable of understanding the meaning of a text fragment. An NLG system needs *facts* as input, as *semantic* units of information. Given our set of 15,483 records like the one shown in the figure, we processed these to build a hierarchical semantic knowledge base.[3] This can then be used by the generation module. We thus studied how much semantic information could be automatically acquired. Figure 4 shows a text generated from

[3]While some of this processing may be considered as *parsing*, we do not claim that the system *understands* the text fragments.

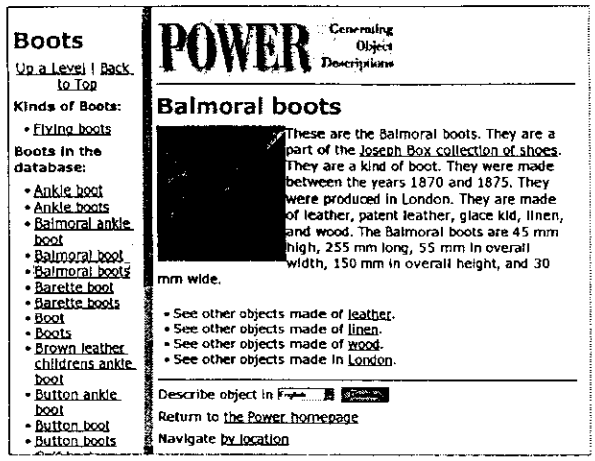


Figure 4: A text generated by PowerTNG

'Balmoral boots'
Estos son 'balmoral boots'. Fueron hechos entre los años 1870 y 1875. Fue produdido en London, Inglaterra. Estàn hechos de cuero, 'patent leather', 'glace kid', lino y madera. Los 'Balmoral boots' tienen 45 mm de altura, 255 mm de largo, 55 mm de anchura total, 150 mm de altura total y 30 mm de ancho.

'Balmoral boots'
Ces objects sont des 'Balmoral boots'. Ils ont été fabriqués en entre 1870 et 1875. Ils ont été fabriqués à 'London'. Ils sont en 'leather', 'patent leather', 'glace kid', 'linen' et 'wood'. Les 'Balmoral boots' ont 45 mm de hauteur, 255 mm de longeur, 55 mm de largeur totale, 150 mm de hauteur totale et 30 mm de largeur.

Figure 5: Texts in different languages: Spanish and French

the information obtained automatically from the above database record. It is clearly of a less sophisticated nature than the text shown in Figure 2. This is largely because the knowledge base created automatically from the database record is not as sophisticated or rich as the knowledge base created by hand for the purpose of generating descriptions. Yet, even with less information, some of the benefits of producing texts automatically on demand using NLG techniques can be seen. For example, the text can be generated in different languages on demand, as shown in Figure 5, provided the appropriate linguistic resources are available.[4]

We now look in more detail at how we extracted information automatically from the database. The Powerhouse museum provided us with a dump of their database in ASCII format with the fields in the

[4]In this figure, the words in single quotes indicate that the word is in English, as the appropriate lexical item for the language of the generated text has not yet been provided.

database records indicated by tags at the beginning of each field. They also provided us with a thesaurus of object types. This is the only information we had available to build a structured knowledge base from which to produce text. To be able to obtain a knowledge base that can serve as input to the generation process, we processed the data file as follows:

1. normalisation of the database: This is to ensure that each record is surrounded by an SGML-style `rec` tag, and that each field of an entry is on a single line.
2. extraction of dimensions: In this step, a Perl script extracts the dimensions of the objects. This information resides in easily identifiable fields (e.g., the DIM field in Figure 3) and the information in that field is structured and can be decomposed into its subfields (e.g., length, height).
3. extraction of thesaurus categories: This step involves trying to identify the thesaurus category that applies to each of the objects in the database. This is normally found in the OBN (Object Name) field and corresponds to an entry in the Powerhouse's thesaurus.
4. extraction of names, materials, makers, locations, and dates of construction: This involves extracting information from the textual information contained in the database records. Most of our work here so far has focussed on the OBS (Object Statement) field. This field is supposed to include information encoded in a standardised and rigorous way. However, in practice, not all the information that is supposed to be included is present, or it is present in a different order, or format, from the norm. Yet, with the help of information from the thesaurus, we were able to identify information such as date of manufacturing or purchasing, materials and location.
5. extraction of PART-OF and A-KIND-OF information: We use the OID (Object ID) field to determine the PART-OF hierarchy for the database. For example, in the database record shown in Figure 3, the OID H4448-513 indicates that this object is the 513^{th} part of the object with OID H4448 (in this case the Balmoral boots are part of a large collection of footwear). According to the database specifications, an object may have parts, sub-parts, and sub-sub-parts.

The result of this processing is the construction of an information record such as the one shown in Figure 6, from which our knowledge base containing 15,483 objects used by the PowerTNG NLG system was constructed.

```
OBS.original: Balmoral boots, elastic sided, pair, women's,
patent/kid/leather/elasticised fabric/wood,/brass prize work,
[Gundry & Sons], England, c.1851; 1862-1869.
OBS.object: Balmoral boots
OBS.object.number: plural
OBS.material.1: patent
OBS.material.2: kid
OBS.material.3: leather
OBS.material.4: elasticised fabric
OBS.material.5: wood
OBS.production.country: England
OBS.create: 1851
OBS.create.inexact: 1
```

Figure 6: Data extracted from an Object Statement field.

4 Discussion

From our experiments with the systems Power and PowerTNG, we conclude that NLG techniques are appropriate to produce virtual documents on demand, and that in fact they offer a number of advantages over techniques dealing with already existing documents, in specific situations where underlying information is available. In order for this technology to be effective, however, we would like to make some recommendations on issues related to the source of information. In our experiments with an existing database which was not built for this specific purpose, we learnt some valuable lessons. To obtain sufficient information for the production of sophisticated texts, special care must be taken when the database is designed and populated. Lack of structure and consistency in the database, and the inclusion of large amount of unstructured textual information in the fields restricted the amount of high quality information we were able to extract. Yet, we believe these obstacles are not insurmountable in many cases. Indeed, the features required (e.g., consistency and structure) are important for well-designed databases. It is important to note that complying with them would not necessarily impose constraints on the end-users, given appropriate interfaces and tools. In fact the database would be greatly improved for all.

It is also quite possible that there will be fewer problems of this kind in the future: as application programs become more sophisticated, it is likely that their underlying representations will have the characteristics required and that their content will move closer to the kinds of rich symbolic structures expected in AI systems. It is also possible that increasingly sophisticated data input tools will be developed to enable the construction of such knowledge bases (see for example, [6, 7]), so that database entry clerks do not have to acquire the skills of knowledge engineers in order to do their jobs.

5 Acknowledgments

Thanks are due to Matthew Connell and Kevin Sumption from the Powerhouse Museum for their

enthusiasm in supporting this project and to Des Beechey for supplying the Powerhouse Museum's databases.

References

[1] Robert Dale and Maria Milosavljevic [1996] Authoring on Demand: Natural Language Generation of Hypermedia Documents. In *Proceedings of the First Australian Document Computing Symposium (ADCS'96)*. Melbourne, Australia.

[2] Robert Dale, Jon Oberlander, Maria Milosavljevic and Alistair Knott [in press] Integrating natural language generation and hypertext to produce dynamic documents. *Interacting with Computers.*

[3] Leila Kosseim and Guy Lapalme [1994] Content and Rhetorical Status Selection in Instructional Texts. In *Proceedings of The International Workshop on Natural Language Generation*, pp. 53–60.

[4] Kathy McKeown [1985] Discourse strategies for generating natural-language text. *Artificial Intelligence* **27**:1–41.

[5] Maria Milosavljevic [1997] Augmenting the User's Knowledge via Comparison. In *Proceedings of the 6th International Conference on User Modelling*. Sardinia.

[6] Cécile Paris and Keith Vander Linden [1996] Building Knowledge Bases for the Generation of Software Documentation. In *Proceedings of the 1996 Meeting of the International Association for Computational Linguistics (COLING-96).*

[7] Cécile Paris, Keith Vander Linden and Shijian Lu [1998] Automatic Document Creation from Software Specification. In *Proceedings of the Third Australian Document Computing Symposium (ASDC'98)*, Sydney, Australia, August 21, 1998.

[8] Dietmar Rösner and Manfred Stede [1992] TECHDOC: A system for the automatic production of multilingual technical documents. In *Proceedings of KONVENS-92.* Springer. Berlin. Also available as technical report FAW-TR-92021, FAW, Ulm, Germany.

[9] Anne-Marie Vercoustre, Jon Dell'Oro, Brendan Hills [1997] Reuse of Information through Virtual Documents. In *Proceedings of the Second Australian Document Computing Symposium*, Melbourne Australia, pp55-64.

Automatic Document Creation from Software Specifications

Cécile Paris†, Keith Vander Linden‡
and Shijian Lu†

†CSIRO Mathematical and Information Sciences
Locked Bag 17
North Ryde NSW 1670 Australia
{Cecile.Paris,shijian.lu}@cmis.csiro.au

‡Calvin College
Department of Computer Science
Grand Rapids, MI 49546, USA
kvlinden@calvin.edu

Abstract

Software documentation, and in particular, on-line help is a crucial aspect of a software system. Producing and maintaining it, however, is both labor-intensive and tedious, making it a candidate for automation. This paper presents our work on automatically generating hypertext based on-line help, starting from software specifications. Our approach is motivated by practical considerations, such as the impossibility to construct by hand the semantic knowledge base typically required by a generation system.

Keywords Natural Language Generation, Software documentation, hypertext

1 Introduction

It is widely accepted that software documentation, and in particular, on-line help, is a crucial aspect of any software system, and that writing and maintaining it is labor-intensive and tedious. It is thus desirable to automate its production. While much work has been done in this area, e.g., [14, 6, 12], there are limitations to current language technology (LT) that prevent us exploiting it in realistic settings:

- Full language generation is knowledge intensive, and systems tend to rely on deep semantic models, usually complex and manually built. This manual construction is clearly impractical in real applications.
- Text generation requires a set of linguistic resources, once again often created by hand. It may not always be possible to provide these resources in such a way that no extensions will be required when the system is deployed. This is especially true of lexical information.

This paper describes our work on developing Isolde[1], an authoring tool that exploits natural language generation (NLG) techniques while addressing the limitations above. Isolde is designed to support the production of hypertext based on-line help. It proposes to re-use, for documentation purposes, as much information as possible from the software specifications of the system to be documented.

The paper is structured as follows. In section 2, we discuss our choice of target text, hypertext-based on-line help. In section 3, we examine how parts of the semantic models might be obtained automatically and how more "shallow" generation resources might be used. We present Isolde's architecture and an example in section 4. Finally, we conclude in section 5.

2 Why procedural hypertext based on-line help

In our work, we are concerned with end-user documentation, as opposed to documentation aimed at programmers, as in [4]. Typical end-user documentation comprises a variety of information, including procedural help, which enumerates the series of steps required to perform a user goal. Procedural help can be seen as an answer to the question "how to", and is typically delivered on-line, in a hypertext form.

Because procedural help describes system functions in terms of user actions on the user interface, it is highly structured and heavily based on the

[1] An Integrated Software and On-Line Documentation Environment.

Proceedings of the Third Australian Document Computing Symposium, Sydney, Australia, August 21, 1998.

programmed behavior of the system. It thus seems realistic to automate its production. This has indeed been pursued in other work, e.g., [16, 12].

From the technical writers' perspective, while writing such help is the easiest part of their task, it is also the most routine and tedious one [11]: it requires them to explore the possible commands to perform a task, in a systematic fashion which does not require much personal input or creativity.

Another important characteristics of on-line help (or any documentation) is that it must change when the underlying software changes. Automating the production of documentation and linking it to the software specifications would thus provide an important benefit over current manual production: it would allow for consistency and better document management.

3 Our Approach

In building the ISOLDE system, we have addressed the limitations of NLG by integrating three approaches:

- the adoption of the human-in-the-loop approach, recognising that it is not always feasible or even desirable to automate the process completely, and always generate text from first principles;
- the automatic acquisition of as much of the semantic and linguistic input as possible;
- the use of shallow semantic and linguistic models.

These approaches have affected several elements of our system. In this section, we discuss these elements and show how they take advantage of some of the characteristics of our application domain. It is important to note that each element in isolation would not be enough, but that their integration gives us the possibility of constructing a realistic and practical system.

Human-in-the-loop approach. While we aim at generating the on-line help automatically, we recognise that this may not be entirely practical at this point in time. Isolde is thus an *authoring tool* for technical writers. The tool produces as much as possible of the on-line help automatically, but also allows for input from the technical writers. In particular, we recognise that:

1. It may not be possible to obtain automatically all the information required to generate on-line help. The tool thus provides an interface to the technical writers to augment the information that was obtained automatically. The interface employs both a graphical tool and a controlled natural language to help the writers.
2. It may not be possible or desirable to generate everything from first principles, i.e., from a fine-grained knowledge base, using full NLG, because of the difficulties involved both in representing formally all the information required, and in acquiring it, whether manually or automatically. For example, it is unclear how one would represent or obtain the information contained in the **Notes** often included in on-line help. In light of this, we allow for the smooth integration of text generated with full NLG and canned text, as in [5].

Since we adopt a human-in-the-loop approach, we must ensure that the input to the generation system be understandable to technical writers, as they may need to modify or augment it.

To generate procedural help, a system needs a representation of the actions that can be performed and the relationships amongst them [12]. We note that this information is essentially that contained in some of the models (*task models*) defined and exploited in human computer interaction (HCI) to perform user requirements analysis, interface design and interface evaluations. Some of these task models are specifically designed to be usable by non-computer specialists. We thus adopt such a formalism, Diane+ [15], for the input representation of Isolde. We have performed an experiment with technical writers validating its claims of readability and usability [10]. We have also designed and implemented a Diane+ task model viewer/editor to serve as an interface to the technical writers.

The automatic acquisition of the knowledge required. An NLG system requires a knowledge base from which text is constructed. In our case, Isolde generates text from a task model, as mentioned above. It must thus obtain this task model. To this end, we turned to Computer Aided Software Engineering (CASE) tools. Studying the knowledge embodied in a typical CASE tool, we noted that standard object-oriented models include both class structure, i.e., information about objects, and system behavior models, i.e., information about actions and the relationships amongst them. System behavior models can form a starting point for constructing the procedural model required to generate instructions. We thus include in our system the ability to construct an initial model (objects, actions, *and* procedures) from the information contained in a CASE tool [7]. While automatic knowledge acquisition has already been employed in some generation systems, approaches put forth have concerned exclusively the acquisition of objects from databases [3], as opposed to processes as in Isolde.

Use of a "shallow" semantic model and its automatic acquisition. The objects and pro-

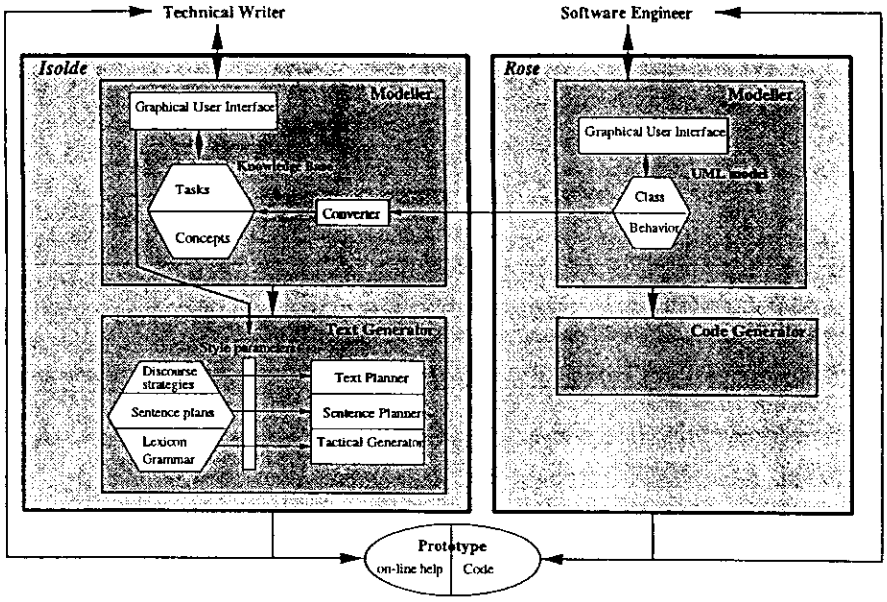


Figure 1: The Isolde Architecture

cesses that form each proposition of the task model must also be available, and they must be appropriately subordinated under the ontology employed by the grammar to guide their realisation. We take advantage of the regularity of our domain and observe from corpus analysis that objects and processes are essentially of one type (using the vocabulary of our ontology [1], objects are `decomposable-objects` and processes are `material-actions`). Isolde thus does not need a *deep* semantic model for its knowledge base to still perform full NLG. It only needs the appropriate links into the ontology. This regularity allows us to avoid constructing an *a priori* semantic model. Instead, the system can create objects on the fly and subordinate them to the ontology at time of creation. This result in a shallow semantic model, but one that can be obtained automatically and is still adequate for the generation of instructions.

Given that concepts and processes can be acquired on the fly, we also need the ability to construct their corresponding lexical items dynamically. This is done through the controlled natural language interface. We are able to do this again because of the regularity of our target text.

4 Isolde: Architecture and example

4.1 Architecture

Isolde combines a software engineering CASE tool with an NLG tool. These tools, shown in Figure 1, are used together to produce a prototype software system (including its user interface) along with portions of its user documentation. The CASE tool, in our case Rational Rose, contains two basic components:

- The *Modeller* allows the software engineer to design the application by building an object-oriented model including two interrelated elements: the *class structure* of the application (objects and methods), and the *behavior* of the application (potential user goals and sequences of method applications). The model is built with Rose's Unified Modelling Language (UML) [13];
- The *Code Generator* automatically generates some of the code. The engineer then codes the rest by hand.

Isolde then includes two basic components:

- The *Modeller*, written in Java, allows the technical writer to build and modify the *domain model* (actions and objects mentioned in the instructions), and the *task model*, in Diane+ [15] (the procedural relations between the actions in the domain model).
  The modeller includes a *graphical user interface* and a *controlled natural language interface* to support the technical writers, and a *converter* to automatically much of the information in the two models mentioned above from the class structure and the specification of the behavior of the system, as specified in the CASE tool. This is important as it provides the technical writers with an initial draft of the task model. The converter is written within Rose using RoseScript. The modeller also allows the technical writer to enter some

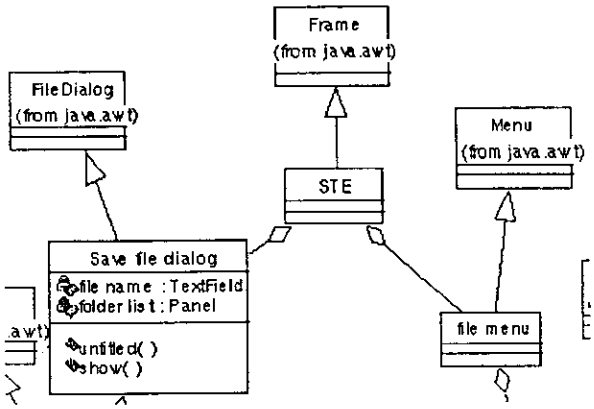


Figure 2: The UML Class Structure

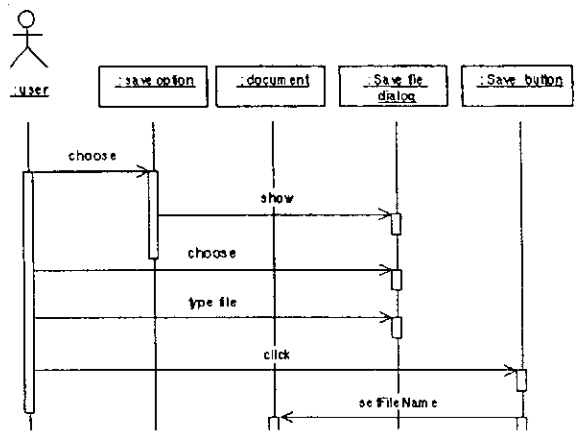


Figure 3: The specification of the behavior in UML

free text to be included in the instructions as canned text if necessary.

- The *Text Generator*, written in Lisp, takes the domain and task models as input, and plans text and sentence structures, using a version of the Moore and Paris' Text Planner [9] extended for sentence planning. It generates hypertext instructions in English, using KPML [2]. The plans specify where hyperlinks are to be included.

The application code and the documentation can then be combined to produce a full prototype which may be evaluated.

4.2 Process and Example

We use STE, a Simple Text Editor written in Java,[2] for our example. The design process starts with software designers developing a UML model of the class and behavioral structures of the intended system. Portions of these models are shown in Figures 2 and 3. Although they are built for the purpose of software engineering, they also contain much information useful for text generation.

The converter in ISOLDE takes the portions of the UML model useful for text generation and creates the initial domain and task models automatically. The modeller then allows the technical writer to modify and extend it. The result is shown in Figure 4. In this example, most of the model was derived automatically, with the writer making the following modifications:

- Two actions performed by the system were removed. The converter uses heuristics to filter out actions not relevant to the user, but this process cannot be guaranteed accurate. This is why a human-in-the-loop approach is important.
- The names of the actions derived from the software specification were modified by the technical writer to conform to the syntax of the CNL. Some of the original names used software engineering jargon and were thus not appropriate to use in instructions.
- The writer added a note to be included in the final text.

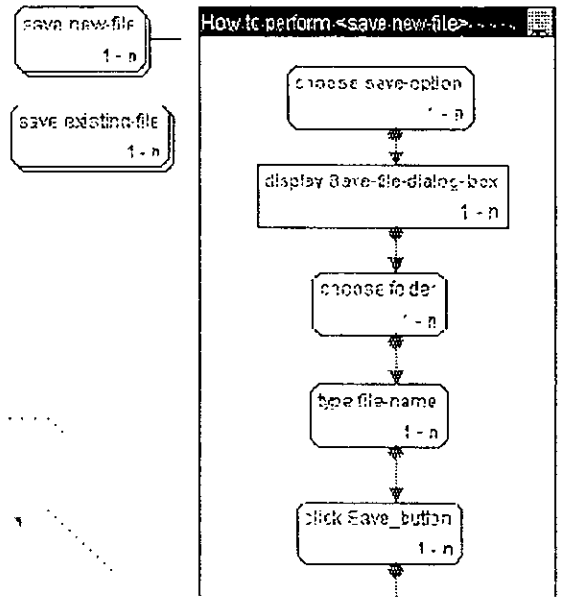


Figure 4: The task model in Diane+

The text/sentence planner builds a single text structure based on RST [8], extended to include hypertext links. The tactical generator then uses this structure to produce the 3 texts shown

[2] STE is pre-existing freeware; We have reverse engineered a UML model for it using Rose.

To use ste.
What would you like to do?

- Create a file.
  How?
- Save a file.
  How?
- Open a file.
  How?
- Print a file.
  How?
- Close a file.
  How?

To save a file.
What would you like to do?

- Save a new file.
  How?
- Save an existing file.
  How?

To save a new file.

1. Choose the save option from the file menu.
   The system will display the save file dialog box.
2. Choose the folder.
3. Type the file name.
4. Click the save button.
   The system will save the document.

Note If you would like to create a new folder in which to save your document, you can click on the **New Folder** button on the top of the dialog box.

Figure 5: The Generated On-line Help

in Figure 5. The underline text represents a hypertext link.

5 Conclusions

In this paper we have presented Isolde, a system for generating on-line procedural help in hypertext form. The system architecture attempts to address the fundamental difficulties of building semantic and linguistic resources by taking input from the HCI and software engineering processes, and by operating with shallow resources. The resulting system provides a practical approach to the generation of on-line, procedural help.

Acknowledgements

This work is partially supported by the Office of Naval Research (ONR) – Grant N00014096-1-0465. We gratefully acknowledge the participation of the members of the team, S. Balbo, and N. Ozkan, and are thankful to V. Anciaux and C. Plier for their implementation of the task model editor. We also thank the technical writing team of IBM Global Services Australia.

References

[1] John A. Bateman. Upper modeling: Organizing knowledge for natural language processing. In Kathleen R. McKeown, Johanna D. Moore and Sergei Nirenburg (editors), *Proceedings of the Fifth International Workshop on Natural Language Generation,* June 3–6, Dawson, PA, 1990.

[2] John A. Bateman. KPML Development Environment. Technical report, Institut für Integrierte Publikations- und Informationssysteme (IPSI), GMD, Darmstadt, January 1997. Release 1.1.

[3] Robert Dale, Stephen J Green, Maria Milosavljevic, Cécile Paris, Cornelia Verspoor and Sandra Williams. Using Natural Language Generation Techniques to Produce Virtual Documents. In *Proceedings of the 9th International Conference and Workshop on Database and Expert Systems Applications*, 1999.

[4] L. Johnson. Dynamic (Re)Generation of Software Documentation. In *Proc. of the 4th Systems Reengineering Technology Workshop, John Hopkins University*, pages 57–77, 1994.

[5] A. Knott, C. Mellish and M. O'Donnell. Sources of Flexibility in Dynamic Hypertext Generation. In *Proc. of the 8th International Workshop on Natural Language Generation*, 1996.

[6] Leila Kosseim and Guy Lapalme. Content and rhetorical status selection in instructional texts. In *Proceedings of the Seventh International Workshop on Natural Language Generation*, Kennebunkport, ME, 21–24 June 1994, pages 53–60, 1994.

[7] Shijian Lu, Cécile Paris and Keith Vander Linden. Towards Automatic Construction of Task Models from Object-Oriented Diagrams. In *To appear in the Proceedings of the IFIP Working Conference on Engineering for Human-Computer Interaction 1998*, 1998.

[8] William C. Mann and Sandra A. Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text: An Interdisciplinary Journal for the Study of Text*, Volume 8, Number 2, pages 243–281, 1988.

[9] Johanna D. Moore and Cécile L. Paris. Planning text for advisory dialogues: Capturing

intentional and rhetorical information. *Computational Linguistics*, Volume 19, Number 4, pages 651–694, 1993.

[10] Nadine Ozkan, Cécile Paris and Sandrine Balbo. Understanding a Task Model: An Experiment. In *Proceedings of Human Computer Interaction'98*, 1998.

[11] Cécile Paris, Nadine Ozkan and Flor Bonifacio. The Design of New Technology for Writing On-Line Help. In *Proceedings of Human Computer Interaction'98*, 1998.

[12] Cécile Paris and Keith Vander Linden. Drafter: An interactive support tool for writing multilingual instructions. *IEEE Computer*, Volume 29, Number 7, pages 49–56, July 1996. (Special Issue on Interactive Natural Language Processing).

[13] Rational Software Corporation. Unified Modelling Language. Notation Guide, v.1.0, January, 1997.

[14] Dietmar Rösner and Manfred Stede. TECHDOC: A system for the automatic production of multilingual technical documents. In *Proceedings of KONVENS-92*, Berlin, 1992. Springer. Also available as technical report FAW-TR-92021, FAW, Ulm, Germany.

[15] Jean-CLaude Tarby and Marie-France Barthet. The DIANE+ Method. In *Proc. of the 2nd International Workshop on Computer-Aided Design of User Interfaces*, 1995.

[16] H. Thimbleby and M. Addison. Intelligent adaptive assistance and its automatic generation. *Interacting with Computers*, Volume 8, Number 1, pages 51–68, 1996.

User-Mediated Word Shape Tokens for Querying Document Images

Alan F. Smeaton and Jerh O'Connor

School of Computer Applications
Dublin City University
Glasnevin, Dublin 9, IRELAND

E-mail asmeaton@compapp.dcu.ie

Abstract

Word Shape Tokens (WSTs) are tokens used to represent words based on the overall shape or contour of a word as it appears in printed text. A character shape code (CSC) mapping function is used to aggregate similarly shaped letters such as "g" and "y" into one single code to represent those letters. The rationale behind this is that it is far easier and more accurate to map a scanned image of a word or letter into its WST representation than it is to map into full ASCII. WSTs were initially applied to the task of language recognition and have proved useful in implementing a computationally lightweight form of OCR. In previous work, we have applied WST representations to information retrieval based on automatically deriving query WSTs from topic descriptions. In the work reported here we extend this to allow a user to judiciously select WSTs as search terms based on the number of surface forms of words which share that WST. We also factor into our experiments for the first time, the WST recognition errors found from an implementation of the WST recognition process. Our results encourage us to further develop the idea of using WSTs for retrieving scanned images of text documents.

Keywords Document management; Retrieval of document images;

1 Introduction

Information retrieval is a task typically performed on machine-readable documents where words can be used to index and represent document content. If the documents are not in machine-readable form but are printed then the typical approach to providing content-based is to apply an OCR process to determine document content and then to index that OCR representation.

Word shape tokens (WSTs) are a representation of words where similarly shaped characters are grouped together using a mapping function called a character shape coding (CSC). WST recognition is much more accurate than OCR because the CSC mapping functions are chosen in such a way as to greatly reduce the need to separate letters which appear to be similar in the first place.

Our previous results on using WSTs for IR have shown that turning a user's topic statement into its WST equivalent and using these search WSTs to match against WSTs derived from document texts leads to poor IR performance because of the number of surface forms of words sharing the same WST. We have extended our earlier retrieval approach to allow a user to judiciously choose WSTs as search terms based on the number of surface forms of words that share those WSTs plus what those surface form words are and how they impact on the overall query. Furthermore, we now also factor into our experiments the type and the rate of WST recognition errors that are likely to arise in a full-scale implementation. While we acknowledge that this appears to be piling one source of fuzziness (word shapes) on top of another (document content retrieval) we believe that there are enough cases of real-world requirements for retrieval on such documents to merit this exploration.

2 Word Shape Tokens

Most of the CSC mapping functions used recently are based on determining the baseline of a text string in a document image, determining the line marking the top of the "x" characters (called the x-line) and identifying characters in the image as being units of connected components. For each of these character components we determine whether they have ascenders which rise above the x-line such as the letters l, t, f, h, b, etc., whether they have descenders such as the letters g, j, q, y, etc. which drop below the baseline and whether they have a disconnected component, namely the dot that goes over the lowercase letters i and j.

Using these basic attributes of each character in a document image, several CSC mapping functions have been developed which are summarized by Spitz [7]. For our experiments we have chosen

Proceedings of the Third Australian Document Computing Symposium, Sydney, Australia, August 21, 1998.

to use one of the simplest CSC mapping functions developed by Spitz and known as V_0 [6] and we choose this because it is the simplest CSC mapping to implement and the most accurate. V_0 has only five codes in its alphabet, which we represent with the letters, A, x, g, i and j and we use the following CSC mapping:

A ← A-Z b d f h k l t 0-9
x ← a c e m n o r s u v w x z
g ← g p q y
i ← i
j ← j

WSTs have had two main areas of application, language recognition and as a pre-process to OCR. For language recognition the WSTs in a document can be measured against the distribution of WSTs for target languages while in OCR WSTs have been used to narrow the set of ASCII characters into which each character token from a document image can be mapped.

3 Word Shape Tokens for Information Retrieval

The idea of using WSTs as the representation for documents when the documents are scanned images rather than ASCII texts, creates some problems for the matching operation. This is because a WST in a document representation cannot be normalized either to handle morphological variations of words (plurals, verb endings, etc.) or variations due to the case of letters (case, Case, CASE). This is because an individual WST may be a representation for more than one word's surface form occurrence and this requires us to handle the word normalization problem in some other way.

Our approach is to generate valid search term variations at query time and to incorporate these variations into a user's query. Word variations due to word morphology could be handled by some linguistic process but our approach is to reverse-stem each search term. Similarly, surface form variations due to the use of upper and lower case letters could be generated algorithmically.

To address both sources of word variation we stemmed a large corpus of document texts (about 80 Mbytes) and recorded, for each generated word stem, the surface forms of words that yielded that stem. At query time we stem each search term and for each we look up the surface forms of words which yielded that stem in our preprocessing, identifying their WST forms as candidate search terms.

In our earlier work in TREC [3] we used this approach to perform retrieval but we found the resulting effectiveness to be quite poor due to having incorporated many search terms which were turned into WSTs for matching against document WSTs where those WSTs could have been used as a representation for many word form occurrences. Thus if a user's original query contained the term "card" then the variants generated for that would have included card, cards, Card, Cards and so on. Incorporating the WST xxxA into the query effectively meant that the words card, cord, rush, cost, cent, most, such, west, rest, and so on, 207 words in all, would all have been incorporated into the query. The effect of this is similar to the effects of the use of pseudo-words in experiments by Sanderson where pseudo-words are simulated words used to replace all occurrences of a set of real words in documents and in queries [5]. Sanderson's experiments showed that the use of pseudo-words that grouped together 5 or 10 actual word occurrences led to a relatively small decrease in retrieval effectiveness. In our WST-based retrieval, however, we see that some of the search terms would correspond to pseudo-words of sizes of the order of hundreds of real world occurrences and Sanderson has not reported experiments of this nature. We speculate that such large sized pseudo-words would degrade retrieval performance even for long queries and this would explain the very poor initial performance of WST-based retrieval.

In the work reported here we have extended this further by developing a system which allows a user to judiciously choose WSTs (or their word equivalents) from a set of WSTs automatically derived from an initial topic description by reverse-stemming and lookup in our lexicon, and we present this system and evaluate its retrieval effectiveness. We have also incorporated noise into the process that turns documents into WSTs by factoring in noise parameters determined from an actual implementation of the WST recognition process [8]. The diagram in Fig. 1 summarizes how documents have been indexed in our approach and how our retrieval system operates. The next section presents results from our experiments.

4 Experiments

4.1 Documents, Queries and Evaluation

The documents we used in our experiments were taken from the TREC data-set, specifically the set of documents used in category B of TREC-5 [10]. The queries and relevance judgments we used were derived from the topic statements for TREC-5 with an average of 21 relevant documents per topic. Evaluation is performed by calculating precision at the 11 standard recall points as used in TREC [10]. In all the retrieval experiments we report here, retrieval is based on scoring each document in the collection based on the sum of the $tf \times IDF$ weights of search terms occurring in each document and

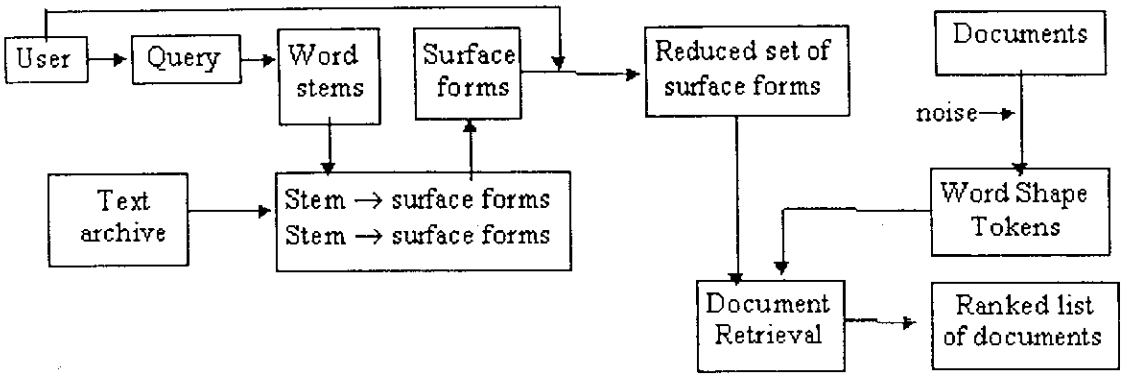


Figure 1: Schematic Outline of our WST-based Retrieval System.

ranking documents based on these scores. Search terms in this case are WSTs.

4.2 Errors in the WST Recognition Process

Although it is computationally easier than OCR, WST recognition will still have errors and these are likely to have some impact on the quality of retrieval. To factor this parameter into our experiments we turned the original TREC document texts from ASCII into their WST equivalents using the V_0 CSC mapping but we introduced CSC mapping errors. The rate at which errors were introduced, and their type, was based on an implementation of WST recognition by Spitz [8] and the parameters for this noise were generated by comparing the recognised text against the ground truth for a segment of the University of Washington English Document Image Database [1].

Like OCR, the WST recognition process has errors of three kinds, deletion of characters that should have been recognised, insertion of spurious characters and the substitution of one character for another and these errors can occur in combination. In the ground truth documents used by Spitz [8], there were 49625 characters of which 9539 were spaces or punctuation marks. When mapped to their CSC codes, the remaining 40086 characters were distributed with a frequency as shown in the second column of Table 1.

The number of spurious insertions of CSC-coded characters was negligible so the error type we are interested in modeling, in addition to deletions, is the incorrect substitution of one CSC character for another the parameters for which are shown in Table 2.

We note from Tables 1 and 2 that the chief type of recognition error in WST recognition is caused by the high number of incorrect deletions of characters from the ground truth in the WST text and the number of "A" characters recognized as "x".

CSC	Ground Truth Totals	Deletions or recognised as punctuation
A	11208	683 (6.1%)
g	2356	78 (3.3%)
i	2897	62 (2.1%)
j	44	2 (4.5%)
x	23581	1524 (6.4%)
Total	40086	2349 (5.8%)

Table 1: Distribution of Ground Truth CSC Characters and Number of Deletions for Each.

Ground Truth	Recognised as				
CSC	A	g	i	j	x
A	9978	1	15	1	530
g	13	2242	1	1	21
i	0	0	2796	9	30
j	0	0	0	42	0
x	14	1	24	2	22016
Total	10005	2244	2836	55	22597

Table 2: Substitution Matric for CSC Codes.

From Spitz's data, the overall recognition accuracy of character shape coded characters is 92.5% and the number of correctly recognized WSTs would be smaller. This low figure is attributable to the poor quality of the sample documents and because the WST recognition process has received relatively little development compared to commercial OCR. Better performance figures can be expected in the future.

4.3 First Experimental Results

The first set of experiments we ran were to index documents by the "noisy" WST recognition with errors as described in section 4.2. We extracted non stop-words from topic descriptions, reverse stemmed and generated all morphological and surface form variations, and then selectively pruned search terms (WSTs) automatically from the query

depending on the number of surface word forms from the lexicon that share each WST search term. Results of pruning WST search terms sharing word forms for more than 20, 15, 10, 5, 3, 2 and uniquely occurring WSTs are presented in Fig. 2

To put these performance figures into context we ran an experiment to compute what would be the upper-bound in retrieval effectiveness for the indexing and retrieval strategies we use. Shown in Fig. 2 as an upperbound is retrieval effectiveness where ASCII documents (as opposed to WST-recognised documents) and topics had stop-words removed, were stemmed and then indexed by these word stems. Comparing the best of the WST-based retrievals with their word-based upper-bound using the same retrieval strategy as shown in Fig. 2 we see WST-based retrieval is still far short of word-based retrieval.

4.4 User Mediated WST Selection

The process of user mediated WST selection involves the user inputting a query and being presented with candidate WSTs from which to choose those to be used as search terms rather than in the previous experiments where the set of search term WSTs was derived automatically from the user's query. This allows a user to judiciously choose whether to incorporate certain WSTs based on the number of surface word forms sharing that WST but also what those shared surface forms are and what other search terms have been incorporated into the query at that point.

For these experiments we developed a Java applet which takes a user's input query, reverse stems it as described earlier and displays the list of all surface word forms derivable from each non-stopword in the input query, except those that share their WST with 100 or more other surface forms (we believe using such WSTs as search terms would be useless). Furthermore, a user is able to view the actual surface forms of each word that shares its WST with a candidate search term, and to view the number of documents in the database containing that WST. Our single user, who is well versed in IR and the use of WSTs, spent an average of 2 to 3 minutes inputting the initial query (which was the TREC topic) and then manually validating the search WSTs to be used. To allow fairer comparison with earlier experiments we allowed the user to use only search terms derived from the topic descriptors, so the user could not bring in any domain knowledge into the process and the choice of search terms was based solely on the search term and the relative frequencies of WSTs. A screendump of the interface is shown in Fig. 3.

The performance of WST-based retrieval with user-validated selection of search terms is shown in Fig. 4. These results show that manually validating WST search terms does indeed improve retrieval effectiveness however it is still not great considering the human effort made,and certainly still far short of word-based retrieval efffectiveness.

Our results demonstrate that WST-based retrieval of scanned document images does appear to have some utility on average but what the results hide is the variability in performance across the 50 topics. Some topics perform reasonably while others are only as good as random retrieval of documents. Fig. 5 shows the average precision for each of the 50 topics for retrieval based on our manually validated WSTs, the best of the automatic frequency-based pruning of WSTs and a retrieval run submitted by Cornell University as part of TREC-5 Category B. We choose to include this Cornell run to show roughly how difficult each of the top topics are.

Fig. 5 shows that for some topics (10 of the 50), manual WST-based retrieval is actually better than the conventional word-based Cornell run but for most topics, word-based retrieval is better. However, there is apparently little correlation between whether topics perform well for WST-based and for word-based retrieval with some topics giving almost zero performance for either of our WST-based approaches. It is these zero-performance queries which deflate the overall average performance of WST-based retrieval. Such queries would appear to be built around words that in all their surface forms share their WST with many other noisy terms and these queries will always be difficult for single-word WST-based retrieval.

5 Information Retrieval from Scanned Document Images

Most of the work to date on content retrieval from paper documents has been based on performing full OCR and attempting to handle, in some way, the resulting OCR errors. Intuitively one would expect that errors due to inaccurate OCR would have a detrimental effect on the performance of subsequent retrieval. A surprising result reported by Taghva [9] showed that if the number of documents and their lengths are large then this can be cancelled out. Subsequent work reported by Mittendorf [4] on a collection of library card catalogues developed a probabilistic term weighting function which had a built-in component to handle OCR error types and this was shown to be very effective in retrieving very short documents.

The most recent work of this type is reported by Harding [2] where retrieval of OCR-degraded text is based on n-gram matching within a probabilistic framework using 2-, 3-, 4- and 5-grams and is shown to yield better retrieval than a word-based approach. Instead of generating all possible n-grams from search terms, they choose a combi-

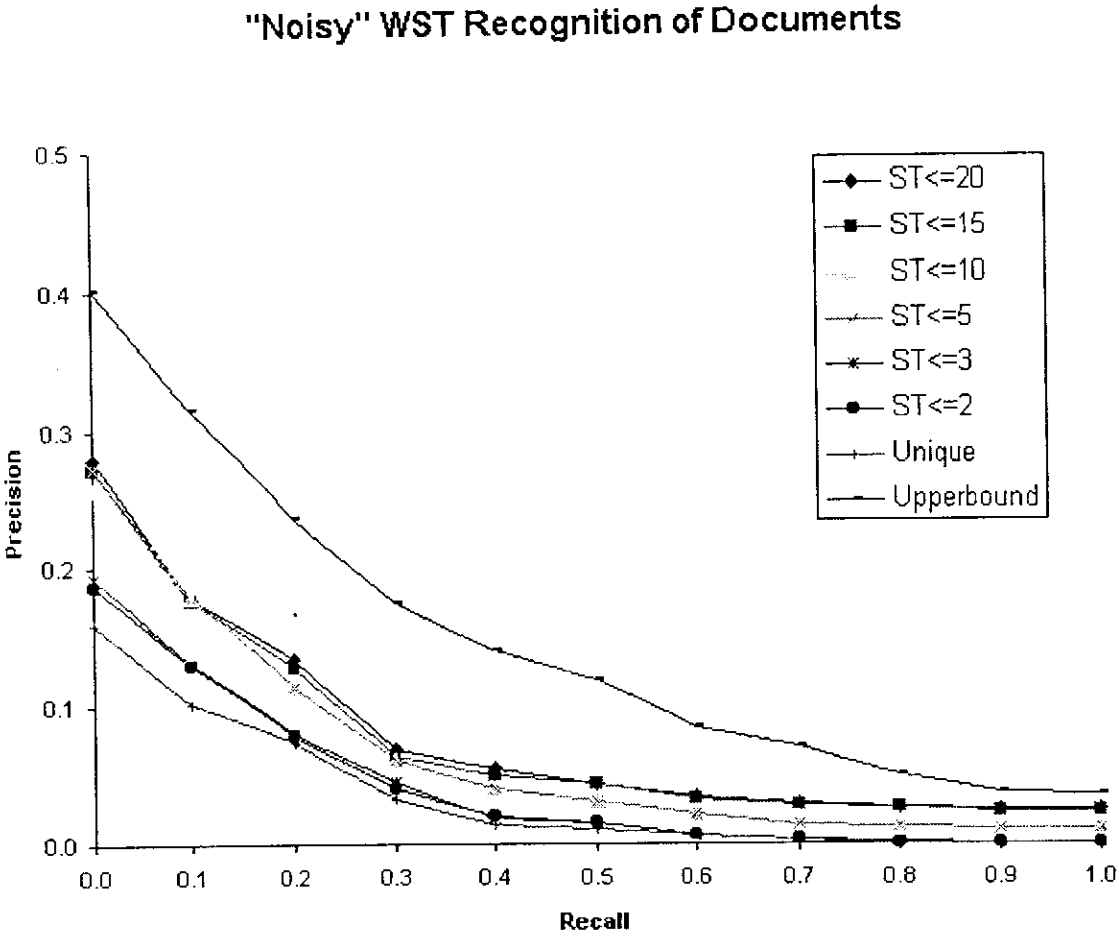


Figure 2: Performance figures for frequency-based WST pruning from topics on "Noisy" CSC-recognized documents.

nation of 2-, 3-, 4- and 5-grams based on position within search terms and choosing only 8 n-grams from any given word. Also described in [2] is a method for retrieval from OCRed text based on identifying appropriate matching and near matching terms for query expansion. This is based on finding search terms "close" to the original search terms from the OCRed text lexicon that will include many OCR errors. Retrieval performance on the small test collections in terms of retrieval effectiveness yielded a big improvement over word-based approaches because of these special treatments to handle errors due to the OCR process. To be strictly fair we should consider comparing our results against retrieval of OCRed data but this was not possible at this time.

Given that these successful approaches to IR on OCR text have incorporated some handling for OCR errors, we should now ask whether our WST-based IR could be extended to specifically handle WST recognition errors. Before we embark on such a line of research we believe we should concentrate on developing a more accurate WST recognition, though the ratio of different error types is likely to be as we have used here.

6 Conclusions

We have presented a series of experimental results for IR from scanned images of document texts that have been represented by WSTs rather than performing full-scale optical character recognition. Retrieval has been shown to be surprisingly effective for some queries given the hugely reduced alphabet size used in the WSTs and pruning such tokens from the user's query, either automatically or with user validation, improves performance. Factoring in WST recognition errors from a real implementation is also included to make our experiments closer to reality.

Our WST-based retrieval is still poor on average though individual queries can perform well. This variability across queries suggests WST-based retrieval has potential if single-token WST recognition can be improved or multiple-token WSTs (phrases) can be introduced.

The word shape recognition task is computationally easier than that of full-scale OCR because of the reduced alphabet size and the choice of letters in the CSC mapping. However, even in scanning and recognizing cleaner document pages and with good quality WST recognition, better than we have used here, WST-based IR would not perform as well as retrieval based on a poor OCR because of the numbers of surface word forms sharing many WSTs. Furthermore, OCR techniques also have the potential for further improvements [7]. We believe that it is only where the quality of the original document images is poor and OCR very noisy, as in the case of some faxes or older legacy documents, that WST-based IR has a role and we are further encouraged by the results presented here to develop further research.

Acknowledgements

The authors gratefully acknowledge the help of Larry Spitz in performing this work.

References

[1] S. Chen, M.Y. Jaisimha, J. Ha, R.M. Haralick & I.T. Phillips. *Reference Manual for UW English Document Image Database I.* The University of Washington, 1993.

[2] S. M. Harding, W. B. Croft & C. Weir Probabilistic Retrieval of OCR Degraded Text Using N-Grams. In *Research and Advanced Technology for Digital Libraries: First European Conference, ECDL'97*, Pisa, Italy, Springer Lecture Notes in Computer Science No. 1324 September, 1997.

[3] F. Kelledy & A. F. Smeaton. TREC-5 Experiments at Dublin City University: Query Space Reduction, Spanish Stemming and Character Shape Coding. In *The Fifth Text Retrieval Conference (TREC-5)*, NIST Special Publication 500-238, 1997.

[4] E. Mittendorf, P Schäuble & P. Sheridan. Applying Probabilistic Term Weighting to OCR Text in the Case of a Large Alphabetic Library Catalogue. In *Proceedings of SIGIR Conference*, Zürich, Switzerland, ACM Press, August, 1996.

[5] M. Sanderson & C. J. van Rijsbergen. The Impact on Retrieval Effectiveness of the Skewed Frequency Distribution of a Word's Senses. *Submitted for publication*, 1997.

[6] A. L. Spitz. Generalized Line, Word, and Character Finding *Progress in Image Analysis and Processing III. S. Impedovo (Ed.)*, pages 377–383, World Scientific, 1993.

[7] A. L. Spitz. Moby Dick Meets GEOCR: Lexical Considerations in Word Recognition. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, Ulm, Germany, IEEE Press, August, 1997.

[8] A. L. Spitz, *Personal Communication*, 1997

[9] K. Taghva, J. Borsack, & A. Condit. Results of Applying Probabilistic IR to OCR Text. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 202–212, 1994.

[10] E. Voorhees & D. Harman. Overview of the Fifth Text Retrieval Conference (TREC-5). In *The Fifth Text Retrieval Conference (TREC-5)*, NIST Special Publication 500-238, pages 1–28, 1997.

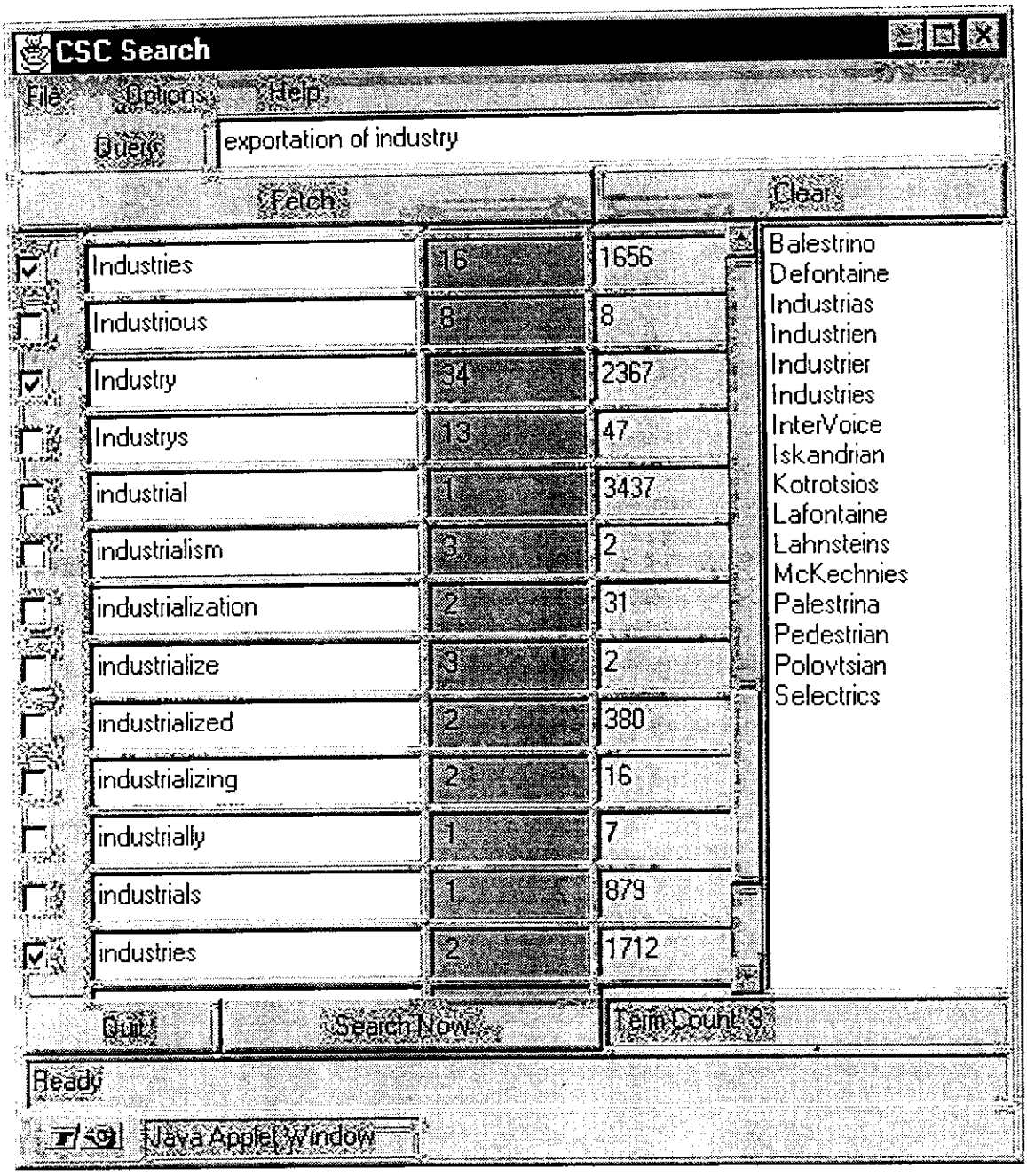


Figure 3: Interface to Interactive WST Selection.

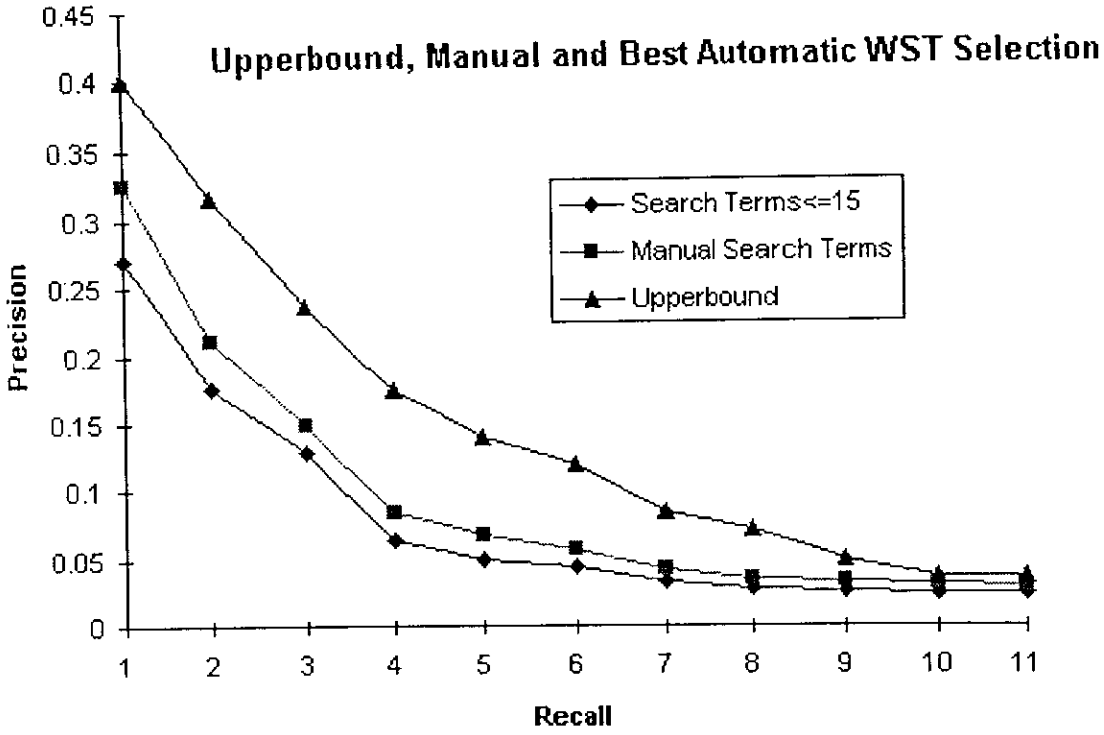


Figure 4: Performance of Manual vs. Best Automatic vs. Upper-bound.

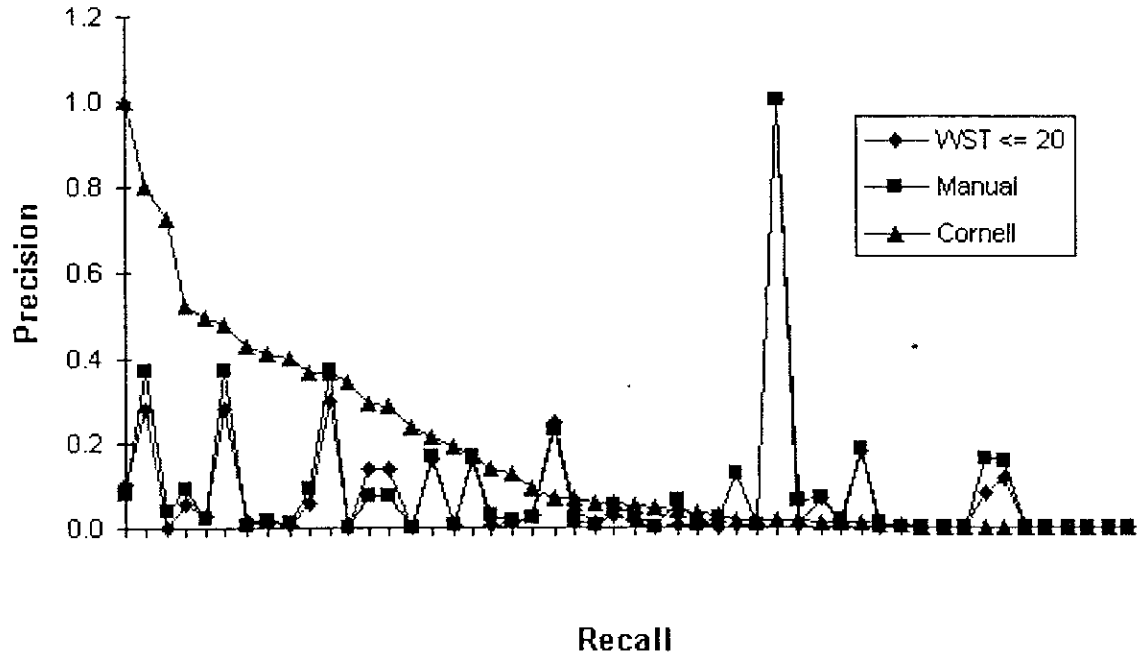


Figure 5: Performance per Query for Cornell, WST Manual and Automatic WST Selection (ordered L to R by decreasing Average Precision).

Essential Drug Informatics

Vincent H Guerrini

Department of Computer Science
and Electrical Engineering
University of Queensland
Q 4072
Australia

csvguerr@mailbox.uq.edu.au

Robert M Colomb

Department of Computer Science
and Electrical Engineering
University of Queensland
Q 4072
Australia

colomb@csee.uq.edu.au

Lucio J Filippich

School of Veterinary Science
and Animal Production
University of Queensland
Q 4072
Australia

l.filippich@mailbox.uq.edu.au

Abstract

At work and in emergencies, Precise Drug information (PDI) is required rather than General Drug Information (GDI). Most electronic search models return GDI even though drug terms are unique and limited (800). Drug literature searches and surveys were commissioned to determine users needs for an electronically indexed drug site. Results showed that 82% of respondents preferred PDI over GDI. Drug information was mostly retrieved from textbooks (79%), computers, (45%) and colleagues (33%) whereas only 6% was retrieved from libraries. On computers, 6% used local databases, 62% the internet and 27% programs. Respondents (67%) expressed a need for PDI. To provide PDI, HTML interfaces were linked to uniquely structured drug file indexes. Users were able to pre-select interest level(s) through "justabout", "moreabout" and "latestabout" search paths. PDI was retrieved () through drug groups-(species) and (drug group). Moreabout drug information was retrieved one link further from justabout. Files were uniquely identified by characters at position 1 (species), 2 (about), and 3 to 6 (drug name). The model may be accessed through AOLPress miniweb and changed on an HTML editor. The proposed model provides a means for specialized users to retrieve specialized information electronically.

Keywords: Resource, Discovery, Drugs, Informatics

1 Introduction

Electronic retrieval of drug information remains the domain of standard search query or list methods even though terms are limited to about 800 [1-2]. It is difficult to retrieve PDI in short search steps or further compilation. PDI comprises essential information about drugs for emergency treatments, such as surgery, drug overdoses and interactions [2].

Results showed that veterinarians and allied medical professions preferred electronic interfaces to retrieve PDI. Various user profiles and heuristic sources were used to design the interface. The purpose of the study was to provide a drug retrieval model based on user heuristics

2 Drug indexation

Drug information is indexed pharmacologically as in "pharmacodynamics", "pharmacokinetics" and therapeutics", by treatment related to disease such as "cytotoxic" or by drug names listed alphabetically as in pharmacopeias [3-4]. Key drug terms include chemistry, pharmacology, uses/indications, pharmacokinetics, contraindications, precautions, adverse effects, warnings, overdose, interactions, doses, routes of administration, dosage form and preparation [2].

Indexing is text or record based and manual or automatic. For PDI, a text search is not suitable since too few or general keywords in a search could result in a large answer set of low precision, and too many will result in lost data or low recall[5]. Electronic catalogues comprise lists, articles, abstracts and textbooks or interactive forms linked to a databases[6-12]

For PDI, manual indexing makes use of the UoD improving the probability for correct indexing [13]. Drug terminology is unique and relatively static. By using a controlled vocabulary, as in manual indexing, the problem of predicting keywords is avoided as well. PDI terms include "name", "uses", "form", "route", "dose", "contraindications", "adverse-effects", "overdose/antidote", "interactions", and "preparation".

To retrieve PDI, the proposed model makes use of key-terms in the search of documents which may not be possible to implement in an automated index. In the proposed model the main interface is tightly coupled with the subsequent links to extract information relevant to the original analysis. As PDI is specific, the vocabulary of the initial search and PDI documents will be similar different which makes a full-text retrieval approach unfeasible. Also, the nature of the basic problem situation is such that the user will

frequently not really know what information to look for initially.

The model reduces the effort required to maintain and search the index, and place low requirements on infrastructure and future systems.

3 User surveys and heuristics

Qualitative drug search design is based on information about the age, gender, education, training and motivation of a user [14-15]. Users with a high level of interest include practitioners, toxicologists, pharmacists and drug regulators and moderate interest, academics and alternative medicine practitioners. PDI is intended to save time in emergencies or when information is unavailable by conventional means. The motivation and goals of users unlikely to differ or change. The interface should be user friendly with familiar heuristics.

A survey was commissioned to determine user profiles, heuristics, need and frequency of access to drug information, type of information (general, precise), mode of access (traditional, or electronic), levels of satisfaction with present electronic retrieval methods , preferred methods of electronic retrieval (query, links or direct) and suggestions for a site. Respondents scored the replies from 1 to 6. The level of satisfaction with current electronic sites was also measured. Computer use bias was offset by surveying non-computer users. One survey was sent to the VETPLUS list [16] and the other was given to private and university practitioners in the Brisbane area, Queensland, Australia. The summarized questions and answers from the 54 respondents, follow:

- Regular need to access drug information: Reply: yes 100%, no 0%.
- Type of information.
  Reply: General 43%, Precise 82%,
- How is information obtained:
  Reply: Books 79%, colleagues 33%, library 6%, computer 45% and company 29%.
- From computers:
  Local database 6%, internet 62%,programs 27%.
- Preferred search methods:
  Reply:Query 74%, direct 64%
- Satisfaction with electronic retrieval: (60% response rate).
  Reply: dissatisfied 66%.

The results suggest that respondents regularly required drug information. However, 82% respondents expressed a need for PDI compared with 45% for GDI. Drug information was mostly retrieved from books and computers. Library retrieval was surprisingly low; computer use may have influenced this finding. There was also significant dissatisfaction with internet drug sites. Possibly, respondents expressed a need for PDI because drug interfaces did not provide this information.

4 Site Design and Maintenance

To address the objectives and maintenance reqirements for PDI, a number of design decisions were made. The general scheme is shown in Figure 1

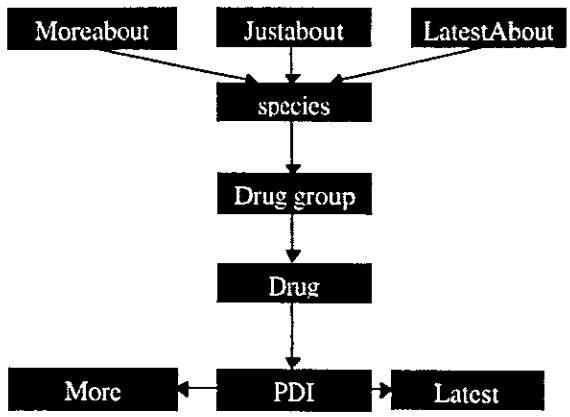


Figure 1- PDI retrieval

Users pre-select interest by the "justabout", "moreabout" and "latestabout" options on the main interface. The "justabout" pathway is shown on Figure 2. A user seeking essential PDI for emergencies may select "justabout" and/or "latest"

and one with more time may choose "moreabout". The "about" interfaces were indexed by drug groups (link 1) (ie:antibiotics and species (dog or cat)), drug

group (link 2) and a drug in a species (PDI). The "moreabout" interface followed the same path as justabout with one further path.

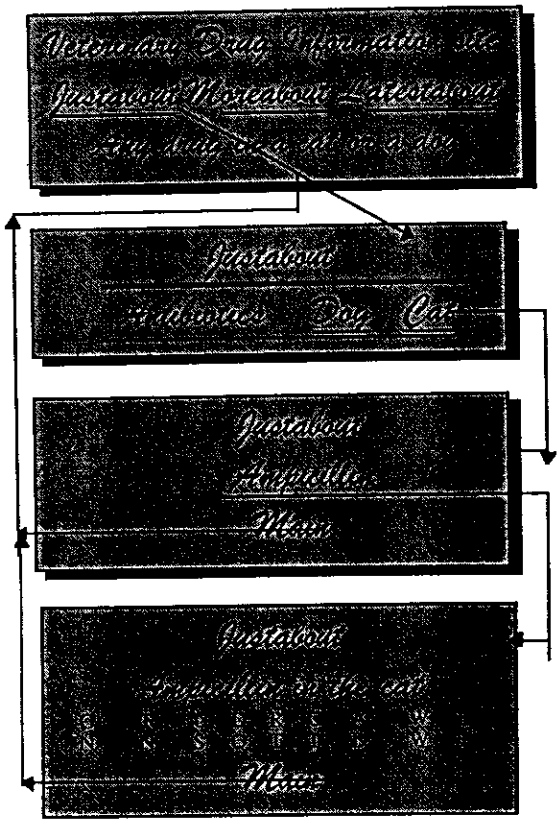


Figure 2. Drug interface design

The "latest" option was linked through drug groups, group and a compedium of latest about drugs linked to Medline at http://www.healthgate.com. The site provided "latest" "about" any drug in any species in the last two years. As shown on Figure 2, the last page of the justabout option provides drug generic (GN) and trade (TN) names, uses (US), dose (DS), route (RT), contraindication (CT), adverse effects and antidote (AD), World Wide Web (WW) and latest (LT) and . The interfaces link to main for re-formulation.

Filenames were structured uniquely to each record. The file djamoxi represents (d)og (j)ustabout (amoxi)cillin whereas cjantibi represents (c)at (j)ustabout (antibi)otics. The files may be upgrade and maintained using AOLPress miniweb ® wherein filenames and links are opened and visualized on one window. An example of the "justabout analgesics (list) in a cat" miniweb is shown on Figure 3.

Figure 3. AOLPress® miniweb

By clicking on a record, each file is opened and modified. Adding a species is achieved by saving an existing file and changing the first character "species". Existing records are changed by direct editing on an HTML browser such as AOLPress® or Netscape composer®.

5 Discussion and Conclusions

Presently, electronic retrieval models make it difficult to retrieve PDI mainly due to interface design and drug indexing. In many cases, extensive searches by query or listing are required and in most cases, the user has to review the data. These findings were in part supported by user surveys demonstrating a 2 to 1 preference for PDI. The proposed model interface pre-determined drug interest levels by "justabout", "moreabout" and "latestabout" search steps. The filenames were uniquely structured for retrieval, maintenance and upgrade. The proposed model returned PDI in 3 search steps and may prove to be a useful system for the retrieval of information by specialized users.

Acknowledgements

This papers was partially supported by a grant from the School of Veterinary Science, University of Queensland.

References

[1] D.C. Plumb Drug Monographs In Veterinary Drug Handbook. Iowa Stae University Press, 1995.

[2] L. S. Goodman and A.G. Gilman. The Pharmacological basis of Therapeutics 7th New York, Macmillan, 1985.

[3] B. Tenant Small animal formulary CDROM 1995.

[4] J.D. Kuiper and H.J. Kuiper. Vetbase 2.1 for WINDOWS Manual. A database with veterinary dosages for non-antibiotic drugs, 1997.

[5] Adriaans & Zantige: In Data Mining, Addison-Wesley 1996

[6] Vet Cabweb. http://vet.cabweb.org/ Index Veterinarius. 1998

[7] Agricola. http://www.nal.usda.gov/ general_info /agricola/agricola.html, 1998

[8] FDA Green Book. http://www.cvm.fda.gov/ iHoundDir/searchtips.html. Searching the greenbook and the FDA center for Veterinary Medicine Web Sites 1998.
[9] G Coppoc Veterinary Pharmacology Course. http://www.vet.purdue.edu/bms/ courses/bms513/index.htm, 1998
[10] Iowa Drug information Network, http://idin.idis.uiowa.edu/
[11] Tierarzneimittelkompendium der Scheiz http://www.kompen.org.ch, 1998
[12] Derwent Drug File http://www.krinfo.ch/ www/rs/DS /OLD/ DDFU.HTML
[13] Goodchild A. An overview of catalogue design problems in resource discovery. Internet Research: *Electronic* Networking *Applications and Policy* Vol 6, No. 1 (1996) pp.33-43.
[14] N.J.C Belkin and B.W. Croft Information filtering and information retrieval:Two sides of the same coin? Communications to the ACM 35:12 29-38, 1992.
[15]. B. Bates. Query languages, retrieval strategies KO. Ch 8 p18-20, 1989.
[16] VETPLUS vetplus-l@u-washington.edu, 1998.

Scholarly Web Sites: A General Method for Structuring Online Research Notebooks

Paul Martin

School of Mathematical and Computing Sciences
Victoria University of Wellington
New Zealand

pmar@mcs.vuw.ac.nz

Abstract

The researcher's notebook is a familiar tool in many disciplines and has several important uses: as a place to record detailed facts, as a place to store reminders of sources of useful information, and as a place to develop new ideas. In this paper we propose a general method for turning the research notebook into an online resource – what we call a 'Scholarly Web Site' – using a novel combination of directories and hypertext.

Keywords Hypertext, electronic notebooks, research methodology, WWW.

1 Introduction

The researcher's diary or notebook is a familiar tool in many disciplines. For our purposes, this paper-based friend has several important uses (Blaxter [2]). First, notebooks are used to look up specific facts, e.g. the page numbers of an article to be cited. Second, notebooks are used indirectly to find (or re-find) other material, e.g. the library's call number for an important text may be recorded in the notebook. Third, notebooks are used as a repository of information out of which new relationships and conclusions may be discovered. Fourth, notebooks – especially those in the form of loose-leaf folders or index cards – are used as a practice ground for marshaling and re-marshaling ideas until a logical disposition is found. Notebooks also have an important role in the verification of research (Zobel [6]) but this topic is beyond the scope of this paper.

The computer can extend the traditional paper-based notebook in two important ways, replacing the pen and paper to facilitate the production of a neat and tidy record, and providing a fast and effective search tool for locating information. The computer can also provide different 'views' of the material in the electronic notebook, as well as greatly speeding the re-marshaling of material.

Proceedings of the Third Australian Document Computing Symposium, Sydney, Australia, August 21, 1998.

2 A general method for structuring electronic notebooks

Our approach to producing online research notebooks uses standard WWW technology to structure hypertext nodes (Conklin [3]) into a 'Scholarly Web Site' (SWS). The researcher/author puts the contents of the notebook into a series of HTML files which are made available as Web pages using any conventional Web server. In our future work section we describe how these basic tools may be usefully augmented by some new tools.

The pages of a SWS are of two kinds: directory and content nodes. Since SWSs are not intended to exist in isolation but are intended to be part of the WWW, we also talk about 'external' nodes, meaning nodes not owned by the author of the SWS but directly linked to the SWS. The SWS has three kinds of link, directory, content and external, named according to the kind of destination node.

2.1 Directory nodes

The directory nodes are the main navigational device in a SWS and function in a similar way to the 'landmarks' described by Balasubramanian [1]. There are only a small number of directory nodes in any particular SWS and they are nodes which soon become familiar to the reader and help prevent her becoming 'lost in hyperspace' (Nielsen [4]). Directory nodes also function as familiar 'jumping off points' for the rest of the SWS.

Internally, a directory node is an ordered list of objects of the same type. For example, a SWS may have a directory of 'conference' type objects where each object represents knowledge acquired by the researcher about a particular conference. The type of an object determines the attributes of the object – a conference object might have 'title', 'acronym' and 'location' attributes. The intention is that the directory entry for a particular object will represent (directly or via links) *all* information known to the researcher about that object. Further, each object will have a rich network of directory or content links to every node in the SWS where it is mentioned, and each mention will have a (directory) link back

to the object. A fragment of an example directory node for conferences showing three objects is given in Figure 1.

Adaptive Hypertext and Hypermedia, Workshop on...

- AH '98, second, Pittsburgh, PA, June, in conjunction with Hypertext '98, {WWW}Home page
- AH '94, first, Cape Cod, Massachusetts, August, in conjunction with {CONFERENCE}UM '94

Education Multimedia and Hypermedia, World Conf. on...

- Ed-Media '99, Seattle, Washington, June, {WWW}Announcement

Figure 1: Directory node fragment

Directory links allow directory or content nodes to point to objects in directories and are annotated by parenthetically prefixing the hyperlink with the name of the directory in upper case. This shows that they are directory links (and not content or external links) and the type of the object at the link destination.

Figure 1 contains one directory link from the AH '94 conference object to the UM '94 conference object in the same directory. Figure 2 (below) contains two directory links into the 'ARTICLE' directory and one directory link into the 'PERSON' directory.

2.2 Content nodes

Content nodes contain largely unstructured information that doesn't fit naturally in any directory node. Examples of content nodes include descriptions and discussions of research topics, critiques of published articles, proposals for new work, and archives of work published in non Web-based forums. Figure 2 gives an example of a content node which explains the concept of a hypertext guided tour.

Guided tours The guided tour metaphor has been attributed to {ARTICLE}(Hammond 1987) but {ARTICLE}(Marshall 1989) also talks about guided tours, including the problems that authors have in selecting which nodes to include in each tour. For an online demonstration, see {PERSON}Watter's {WWW}Guided Tours Page. See also {hypertext concepts}trails, {hypertext concepts}paths, and {hypertext concepts}footprints.

Figure 2: Content node fragment

Content links are placed from content or directory nodes to content nodes and are annotated by parenthetically prefixing the hyperlink with some descriptive text in lower case to show that it is a content link and to explain the purpose of the link. Figure 2 contains three content links, all annotated with 'hypertext concepts'.

2.3 External nodes

External nodes are important since they often contain constantly updated or copyright information that can not be included by the researcher in the SWS. External links connect nodes within the SWS to nodes outside the SWS and are annotated with '{WWW}' to show that they are external and to warn the reader that the author accepts little responsibility for what is found at the destination.

2.4 An example SWS

The author has constructed, and has been using on a daily basis since 1997, a SWS for his research. This SWS contains 149 content nodes and just 6 directory nodes. This SWS has been a useful resource in its own right and a valuable testbed for development of the prototype tools described below. The cost of constructing the SWS has been somewhat greater than that of keeping a paper-based notebook, principally because of the need to maintain consistency in presentational format and because of the effort in creating and maintaining the hyperlinks. However, the SWS is considerably easier to browse than a (linear) paper notebook could ever be and the recent development of prototype tools for maintaining links has considerably reduced the effort of adding new information.

3 Evaluation of the SWS

Diverse kinds of information The traditional research notebook contains material of diverse kinds, for example raw data, lists, and draft papers. Being Web-based, the SWS can include inline any file format support by current browsers, plug-ins and helper applications. Further, as contemporary desktop applications are becoming increasingly 'Web aware' word processing documents, spreadsheets, and databases can be almost seamlessly integrated into the SWS.

Specific queries We noted earlier that researchers often need to look up specific facts. The SWS as currently proposed does not explicitly suppose the presence of a query-matching facility, although we note that most Web browsers allow the user to search for text in the current page and that a number of HTML authoring tools include a search engine for searching the pages in the author's Web site. Instead, our proposal anticipates that in most instances the SWS user will not try to formulate their query precisely but will navigate by browsing to find the information required.

Browsing Browsing is well supported in the SWS by means of the directory nodes, which act as 'jumping off' points, and rich interlinking (supported by tools we describe next), which ensures that every node is reachable by multiple, short paths. Further, our system for annotating links (also tool supported) helps users of the SWS interpret link function accurately and thus navigate efficiently.

Cognitive overhead and disorientation The consistency of the link annotations works to reduce 'cognitive overhead' (Wright [5]) while user disorientation is addressed by use of the directories as landmarks. Directories work well as landmarks because they are small in number, contain objects of the same type, and are largely independent of the application domain.

4 Future work

There are two key areas in which work on the SWS proposal still needs to be done. The first area involves evaluating the usability of the SWS with researchers in the field to see whether there are any criteria for effective online research notebooks that we have not anticipated.

The second area involves developing and evaluating software tools to automatically create and maintain the high degree of interlinking. Our prototype tools work by using headings in the text to identify and name objects in the directory nodes. To create directory links, the author need only plant a hint about the link required and the tools will substitute the correct HTML. For example, the tool will use string matching to deduce that the hint 'conf adaptive hypertext' refers to the 'Adaptive Hypertext and Hypermedia, Workshop on...' object in the 'CONFERENCE' directory.

We are also investigating automatic inference of new links – we know of several kinds of automatic inference that are possible, although we have yet to establish which are useful to the SWS author.

5 Conclusions

In this paper we have described the uses to which the traditional researcher's notebook is put and proposed a general method for structuring online versions of notebooks which we call Scholarly Web Sites. The unique characteristic of the SWS is its use of richly-linked directories as familiar 'landmarks' from which users browse the content.

References

[1] V. Balasubramanian. State of the art review on hypermedia issues and applications. Available from http://www.isg.sfu.ca / ~duchier / misc / hypertext_review, December 1993.

[2] Loraine Blaxter, Christina Hughes and Malcolm Tight. *How To Research.* Open University Press, Buckingham, United Kingdom, 1996.

[3] Jeff Conklin. Hypertext: An introduction and survey. *IEEE Computer*, Volume 20, pages 17–41, 1987.

[4] Jakob Nielsen. The art of navigating through hypertext. *Communications of the ACM*, Volume 33, Number 3, pages 296–310, 1990.

[5] Patricia Wright. Cognitive overheads and prostheses: Some issues in evaluating hypertexts. In *Proceedings of Hypertext*, pages 1–12. ACM Press, New York, 1991.

[6] J. Zobel. Reliable research: Towards experimental standards for computer science. In *21st Australasian Computer Science Conference*, pages 217–229, Perth, Australia, February 1997.

Information Retrieval in Documents Using Semantic Criteria

B. Mills, S. Venkatesh, M. Kumar

School of Computing
Curtin University of Technology
GPO Box U 1987 Perth WA 6845
(brucem,kumar,svetha)@cs.curtin.edu.au

L. Narasimhan

Information Technology Division
Defence Science and Technology Organisation
P O Box 1500 Salisbury SA 5108
lakshmi.narasimhan@dsto.defence.gov.au

Abstract

The central problem of automatic retrieval from unformatted text is that computational devices are not adequately trained to look for associated information. However for complete understanding and information retrieval, a complete artificial intelligence would have to be built. This paper describes a method for achieving significant information retrieval by using a semantic search engine. The underlying semantic information is stored in a network of clarified words, linked by logical connections. We employ simple scoring techniques on collections of paths in this network to establish a degree of relevance between a document and a clarified search criterion. This technique has been applied with success to test examples and can be easily scaled up to search large documents.

Keywords : Semantic network, relevance, parsing wordnet, shortest path.

1 Introduction

Given the state of contemporary culture and technology there is an increasing need for the ability to find information in large amounts of unstructured text. In most cases the unstructured text does not have central index. Attempts to perform this task automatically are hampered by the lack of any general understanding of the text by a computer. A true human designed index is a complex object that incorporates much understanding of what a human will be looking for. However, not only is the index a fading art, but the sheer volume of material to be handled prohibits use of indexing techniques . The amount of information in existence is increasing at a mind-boggling rate, whereas the ratio of useful information to total information is decreasing[6]. This paper describes a method to find a practical resolution to the task of extracting useful information.

During the last few years several parsers [3,4], inference engines [2], and text retrieval systems[5] have been built to aid information processing. Maruyama pioneered constraint- based parsers [4] that use constraint dependency grammar (CDG) to process sentences. Harper et al. employed CDG technique in their PARSEC parser that can process word graphs containing multiple sentences on a massively parallel computer[3]. Harabagiu and Maldovan [2] present a parallel inference engine that uses markers for extracting inference from text. However, the emphasis of this research work is to determine the degree of similarity (or closeness) of a given document to a given set of criteria. The primary goal of this paper is to develop techniques to select documents relevant to semantic criteria in a manner acceptable to humans. This is fundamentally an empirical goal. The secondary goal is a precise theoretical model for use in formal discussions about the device.

A set of sentence fragments is provided as search criteria. Then, the sense of meaning of each word in the search criteria is clarified interactively. Further the user indicates his/her preference for the direction the search should take, generalization or specialization for example. We consider this essential for refining the process of search, as without this form of clarification the user's intentions remain unknown. Each sentence in every given document is parsed for part of speech and sense of meaning and then appraised for relevance to each criteria, and a total score indicating computed relevance is given.

This paper firstly describes in broad terms the nature of the understanding that the system has of the text and how this is used to establish relevance. A precise definition of the structure used for the semantic information is then given and explained. The method of extraction of the graph of connections between a criterion and a sentence is indicated, and the scoring technique expounded. The results of sample tests on the method are also included.

2 The Problem

2.1 Problem Statement

Given search criteria entered by a human user including semantic content and the sense of meaning of the words, determine which of a large collection of documents is most relevant to these criteria.

2.2 The Basic Approach

A clarified word is a written form; for example the clarified word for "dogs", comprises the *part of speech* and sense indication, and is represented by "dog (n1)". Each criterion is a sentence fragment made up from clarified words, for example, "dog (n1) eat (v1)". To match a criterion against a parsed sentence, the strength of the relevance between each word in the criterion is computed for each word in the sentence. These scores are combined into a total score as explained in Section 4. The total score, also known as the parallel sum, was arrived at by the idea that irrelevance is the inverse of relevance, and that the irrelevance of two links in series should be the sum of the two individual irrelevancies. A criterion such as "dog(n1) eat(v1)" would score well against "canines consume meat" because of the strong connection between "dog(n1)" and "canines" and between "eat(v1)" and "consume".

Each logical connection between words is given a numerical irrelevance rating. The connection between two words is found as the least irrelevant path between the two. Irrelevance of a path is the sum of the irrelevancies of the links. The relevance is the inverse of the irrelevance. For example, "canine" is relevant to "dog" because a dog is a type of canine. In practice a limit is set to the length of the paths searched, above a small number of links the path is considered too obscure to be of interest to the person.

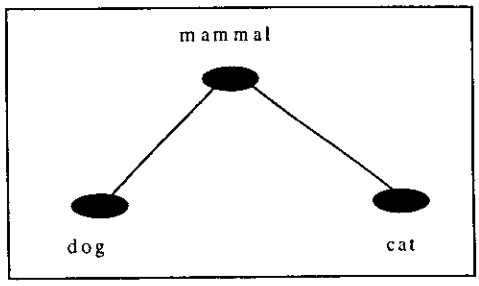


Figure 1: Connection between words

2.3 Semantic Information

The semantic information about English is a network in which the nodes are clarified words and the links indicate primitive logical connections between them. Each link has a type, indicating the nature of the connection, and a positive integral weight, indicating the strength of the connection.

holonym .... "has a part"
hypernym .... "is more general"
hyponym .... "is more specific"

A path in the semantic network can be taken as indicating a compound semantic connection. Thus a dog is a type of mammal, and a cat is an instance of a mammal. So the path from dog to mammal to cat, indicates that a dog is the same type of animal as is a cat. This is a compound connection between the two meanings.

2.4 The Semantic Connection

Given two sets S_1 and S_2 of clarified words, and a positive integral length n, the set of paths with sources in S_1 and destinations in S_2 can be viewed as a compound logical statement describing the connection between the sets. Together with annotation as to which words occur in both sets, this forms the first stage in the computed relevance between two sentences. Due to the overlap of paths it is efficient to compute these paths simultaneously. However this makes the computation of shortest paths more difficult, and so the exact balance for best efficiency needs to be determined. In this implementation we employ a variation of Dijkstra's single source shortest path algorithm.

The resultant graph consists of connected components; connected components of the computed graph have meaning. For example relating "dogs eat" to "canines consume meat", the words "dogs' and "canines" are grouped together as are "eat" and "consume", indicating that they play similar roles in each sentence.

2.5 The problem of ambiguity

Since the input to the process is plain text it is required that each word be clarified for its meaning. Sometimes the meaning can only be narrowed down to several possibilities rather than just one. In such a

case it is possible to include all the selected meanings in the search in order to ensure inclusion of the relevant one. This may however result in spurious matches. If the meaning of the word after all available clarification is still ambiguous, then it may be best to simply exclude this word from consideration since the potential correct match would be swamped by the many spurious ones. This is a problem with handling the documents but not with the criteria, since for the criteria the process has the opportunity to interrogate the user if there is any ambiguity .

2.6 The problem of generality

In practice it was found that some words such as the word "be(v1)" had too many connections to other words; we felt it was best to drop such words from the search entirely. Other words with the same level of generality should also be dropped from the search. Words such as "a","the" and "of" are not in the database, and so are never noticed by the process anyway.

The problems of ambiguity and generality are really very close, on the one hand the meaning of the word has been narrowed down to several options, which together are fairly broad. On the other hand the accepted meaning of the word is too broad. It is possible that the second is just a disguised version of the first, and that the meaning of the general word should be clarified further.

3 Theoretical Background

In general usage the term "graph" or "network" means a set of nodes connected by links. In this paper the term will mean a weighted graph in which there may be multiple connections between two nodes. The nodes represent clarified words. In the following, we detail the structure of the semantic network, the nature of the data structures used in the computation, the algorithm used to produce the path graph connecting two sets of meanings and the scoring technique to produce the final relevance rating.

In the remainder of this paper, a simple graph G is an ordered pair (N,F) where N is an otherwise undefined set of nodes, $F : N \rightarrow Pow(N)$ is the neighbor function, where $Pow(N)$ is the collection of all subsets of N and $F(x_I)$ for a given node x_I is the set of all neighbors of x_I that can be reached by traversing one link in the network.

A *graph* also referred to as the weighted graph or network, is defined as an ordered pair $(N,(F_1,F_2,...,F_m))$ of a set of nodes, together with a collection of neighbor functions. Each neighbor function corresponds to one weight. So $F_i(x)$ is the set of all nodes that can be reached from x via one link of type i.

3.1 Abstract definition of the semantic network

We have an alphabet A, A^* is the set of all finite length strings of elements of A. Let N be the set of nonnegative integers. Let parts of speech, p = {a, n, v, r}, represent adjective, noun, verb and adverb respectively. Thus in particular, A might be {a ... z} and so A^* is {a,aa,.., ab, abb, ..} being the set of all finite length strings of alphabetical characters, regardless of whether they have any sense or meaning in the English language or not. W is a set of triplets comprising a word form (A^*), a part of speech (p) and a sense number(N). Thus $W \in A^* \times p \times N$ and elements of W are represented by $a(bc)$, where $a \in A^*$, $b \in p$, and $c \in N$. Thus W is the set of words being used in the semantic network. For example, W might be {$cat(n1)$, $dog(n2)$, $run(v1)$ ...}. Each word being a textual form together with the part of speech and sense number to clarify the meaning of the word. The elements of the underlying set are clarified words such as "$dog(n1)$". The nodes in the graph are the maximal sets of synonyms of clarified words. The links are logical relations between words, specifically "hypernym", "hyponym", "meronym" and "holonym".

The form partition F is deducible from W, with $a = \underline{b}(\text{mod } F)$ iff $a_I = b_I$. The property of one written form having multiple meanings is called "polysemy". Thus the types distinguished in the form partition are essentially polysemy types. A partition is a collection of disjoint subsets of a set that together includes all the elements in the set. Thus here, it is a way of classifying all the words into types.

The synonym partition S, in which words are grouped according to meaning, is not deducible from W. That is, all elements of a synonym set should be semantically equivalent. This does not imply the ability to replace one by another in every English sentence since problems of syntax, and of common word usage would cause finer distinctions to be made. The full notion of "meaning" as used in this project is combinatorial. No attempt is made to construct (for example) a model based meaning for the words. The meaning of the word is considered informally to be the set of synonyms of that word, together with its location in the semantic network.

3.2 Path Finding Algorithm

To determine the connection between two sentences S_A and S_B, firstly the set of interesting words I_A and I_B such that $I_A \subset S_A$ and $I_B \subset S_B$ are extracted. Effectively, the inputs to the algorithm include, the neighbor function N, the source node set I_A, the destination node set I_B, and the path length limit L. Then the graph composed of all the paths from I_A to I_B up to a given length is computed. We call this the *path graph* from I_A to I_B. The output of the algorithm is a graph containing the source and destination sets. The extra information added is the combination of all the paths of length less than L. Implementation would be complicated if the node sets were not disjoint, but intuitively it makes little difference to the outcome of the path finding algorithm. Hence, we assume disjoint node sets. All the paths up to a given length from a node can be found by a breadth first search from that node. All cross-links, back to a node already on the graph, will be recorded, but not explored any further. If the breadth first search is performed from two nodes, one in the source set and the other in the destination set at the same time to half the depth, then the amount of work required is reduced by a factor depending on the average number of neighbors of a given node.

The semantic network nodes store the index of the node in the path graph structure. All other information is kept in search data structures. The actual graph computed may contain extra links, which could be trimmed if required. The main operations on the path graph are the addition of a node and the addition of a link between nodes already on the graph.

The algorithm maintains two queues; one for the set I_A and the other for the set I_B. Each queue contains nodes already registered with the graph. A node on the queue implies that there is further processing required of the links originating at that node. When a new node is placed onto either queue it has exactly one known link, either a parent or child link.

```
for each node in source do push(node, source_queue)
for each node in destn do  push(node, destn_queue)
  active ← pop(source_queue)
  for each child in forward(active)
    child.parents ← {active}
    active.children ← active.children + {child}
if not marked(child) then push(source,child)
  active  ← pop(destn_queue)
  for each parent in reverse(active)
    parent.children ← {active}
    active.parents ← active.parents + {parent}
        if not marked(parent) then push(destn,parent)
```

Nodes in the source and destination sets are placed in their respective queues. Then, during every step, a node designated as the *active node* is dequeued and each of the active node's neighboring nodes is explored to check whether it is a parent node (already in the queue) or a child node (not in the queue) and appropriate action is taken as shown in the algorithm.

3.3 Path Finding Efficiency

A path from I_A to I_B can be found by doing a depth first search from the nodes in I_A looking for the nodes in I_B . Alternatively a forward search could be done from I_A and a reverse search from I_B and the two results combined. To estimate the work required by each method, let us suppose that the graphs are regular, that the indegree (r) and outdegree of each node are the same.

In the first case if a path is of length l then it requires l^r nodes to be added to the search tree to find the path. In the second case if the trees are expanded at the same rate, then $(l/2)^r$ nodes must be added to each tree producing a total of $\{l^r\ [1/(2^{r-1})]\}$nodes, for any $r>1$ this is an improvement, and for $r=1$ it is no worse.

4 Scoring Technique

Each path graph obtained must have a score allocated. Beyond the intuitive notion that extending a path should not increase its contribution to the score there are few hard and fast rules. In practice, totaling the scores over the paths does not give desired results since the number of paths increases with the richness of the meaning of a word, especially when ambiguity is involved. However taking the maximum score maintains the monotonicity without letting multiplicity take over. This is in the style of a fuzzy logic conjunction, and thus encapsulates the idea that the collection of paths represents a conjunction of logical assertions about the two words.

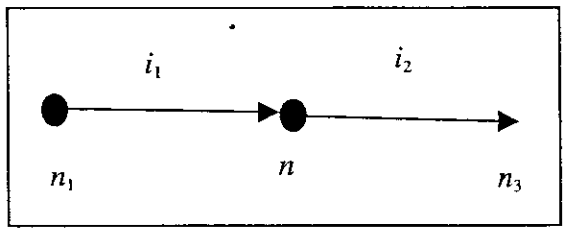


Figure 2: Computing relevance of a path

Consider the graph with three nodes (or words), n_1, n_2, and n_3 shown in Fig. 2. The user specifies the irrelevance for each link; this is a positive integer less than ∞. In this case the irrelevancies between nodes n_1 and n_2 and between n_2 and n_3 are i_1 and i_2 respectively. Now, the irrelevance between n_1 and n_3

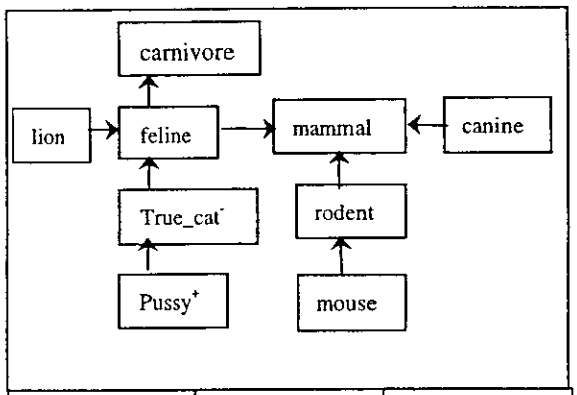
is (i_1+i_2). The effective relevance of the path from n_1 to n_3 is given by, $\{100*[1/(i_1+i_2)]\}$. The relevance between two nodes (or words) is also the strength of the connection between the two nodes (or words). If there are two parallel paths between n_1 and n_3, then we consider the path of maximum relevance. The scoring of the individual word pairs creates a matrix of scores. This matrix represents the relation between the two sets of words, and hence the relation between the two. We illustrate this with the help of an example:

Consider the following two sentences,

S1: The cat ate the mouse.

S2: The pussy played with the rodent.

In an actual run of the algorithm, the following (edited for clarity) graph was produced by the search for a connection between S1 and S2. Given a maximum path length of 4 and common weighting for each link of 1, the



	pussy	rodent
true_cat	1	3
mouse	∞	1

Figure 3: Computing relevance between two sentences

following irrelevance matrix is obtained. In this matrix ∞ indicates that no relevance is to be given to the connection due to the shortest path between the two words over the given limit.

The total relevance indicated by this matrix is the sum of the inverses of the elements in the matrix, 1/1+ 1/3+ 0 + 1/1= 7/3 ≈ 2.33. So the matrix scored about 2.33 out of a maximum possible (given positive integer weights) four. The final score of the matrix, the strength of the connection between the two sentences is 233.

5 Implementation

Some words in the English language change their form in a semi regular manner depending on tense and gender. These aspects are stored in the Wordnet database [6] via a morphological function that can produce a list of options given a word form and its part of speech.

5.1 The Program

There are five basic sections to the required structure. These are the grammatical sense parser, the lexical database, the search algorithm and the search criteria.

The grammatical sense parser determines for each word in a sentence, the part of speech for that word, noun, verb etc. This was obtained as a simple modification of a rule based parser obtained from [1]. The lexical database can determine what other words are related to a word and in what manner, given a word and its sense. The basic lexical data was obtained from the Wordnet lexical database [6]. Functions written for this project perform the tasks of determining which words to be extracted from the database, incorporating those words and links into a linked data structure designed for the purpose and location of words within this structure in order to find the starting point for each search. The search algorithm is a mechanism for finding short paths between the words of two sentence fragments by hunting through the connections determined by the lexical database. The search criteria are a collection of clarified sentence fragments against which the documents will be examined. The routines written include an interactive process for determining the exact meaning of words in the search criteria, so that the search may be narrowed down for speed and reducing the number of spurious matches. The semantic scorer accepts a network of connections between two sets of words and produces a single number that indicates the strength of the semantic association between sets.

5. 2 Semantic Network Creation

The actual data for the semantic network was extracted from the Wordnet database. Although Wordnet did come with software for doing searches along types of links, it threw away a lot of information, specifically about sense numbers, which was not available in the returned structures, hence it had to be laboriously reconstructed from the index files. Thus a new data structure was designed, and search routines developed in order to perform the search faster. The main components provided by Wordnet being, firstly the actual semantic information about the links, and secondly the morphological functions. In practice, the information contained in the Wordnet database has been seen to be incomplete and misleading at times, and it is envisaged that this material should ultimately be found through some other source.

5.3 Results

We tested the semantic network on a number of criteria and document statements. We present one of the test results in detail. The test criteria consisted of the following sentences,

The semantic network is expected to match each document with the corresponding criterion: α for 1, β for 2 and so on. The results show that the strongest match for each criterion is the correct document sentence. We also notice that the strength (or score) of the strongest matching is different for different pairs, strongest for pair 1-α and weakest for pair 1-δ. This result is easily explainable because α is mere restatement of 1 and as such the semantic network indicates strong connection as expected. On the other hand, criterion 4 is about motor vehicles whereas sentence δ is about cheetahs, and as such the semantic network indicates a relatively weak connection between 4 and δ. The matching score (of 100) for statement α with sentence 1 is reasonable because *Gods are willing to help says* something about state of the gods, and *The feline rested on the covering* says the feline is in a state of rest. We also noted a distinct problem related to the scoring of *piranha* against *fish*; we expected stronger connection between 3 and χ. However, the nature of the Wordnet database is such that hypernyms are placed in sequence; piranha is a characin, which is cypiniform, which is soft-finned, which is a teleost, which is a bony fish, which is a fish - resulting in a long path between fish and piranha. The intermediate connections are really not necessary for a non-technical search. This problem can be fixed by training the semantic database.

6 Conclusions and Future Extensions

In this paper we presented a semantic search engine with tagged parts of speech that drops irrelevant material. The words in the documents are tagged for sense and meaning, hence the semantic search engine picks up relevant material unlike a keyword search which would also pick up all those references to synonyms of all senses of the words in the search criteria. For example, if we were looking for dog(n1), then a keyword search would pick up dog(n2) and dog(v1). The parsed semantic search would find dog(n2) but not dog(v1). The latter would be dropped because it is the wrong part of speech.

Thus whether this type of semantic search is better than a keyword search depends on exactly what sort of search criteria are entered and how one measures the improvement. There is a strong interest in tagging for sense number as well as parts of speech. This can be done by parsing for sense number and part of speech using a trainable stochastic and/or rule based tagger and training it on text that has been processed by a human to clarify meaning as well as parts of speech.

An extension to include some parsing for meaning of the document could be done by finding related meanings for different senses of a word, thus producing a consistent interpretation of the meaning of the words. This could take in the whole structure of the document in a primitive manner by incrementally establishing a context of meanings selected on the basis of being often close to some meaning of the words in the document. It would be advantageous if some structural parsing such as grouping into noun phrases and so forth could be performed and handle cases where two sentences having the same set of words may have different meanings.

A closely related task is the one of summarizing a document. Given a collection of search criteria, each paragraph of a document could be classified according to a criterion. A list of the most relevant criteria to each section of the document would give a summary of the content of the document. It is envisaged that a refinement of this process, perhaps also involving frames, would produce a useful brief description of the document.

References

[1] E Brill Parts of speech tagger written at the Department of Computer and Information Science, University of Pennsylvania, and the Spoken Language Systems Group, Laboratory for Computer Science, MIT.

[2] S.M. Harabigiu and D.I. Moldovan, Parallel Inference on a Linguistic Knowledge Base, International Parallel Processing Symposium, Geneva, April, 1997, pp. 204- 208.

[3] M. P. Harper, L.H. Jamieson, C.B. Zoltowski, and R.A. Helzerman, Semantics and Constraint Parsing of Word Graphs, Technical Report, Purdue University.

[4] H. Maruyama, Structural disambiguation with constraint propagation, Proc. of the Annual Meeting of the Association for Computational Linguistics, 1990, pp. 31-38.

[5] M. L. Mauldin, Conceptual Information Retrieval : A Case Study in Adaptive Partial Parsing, Kluwer Academic Publishers, 1991.

[6] G.A. Miller, Wordnet : A Lexical Database, Communications of the ACM, vol. 38, No. 11, Nov. 1995, pp. 39-41.

Criteria	Raw sentence	Processed Sentence
1	The cat sat on the mat	the (x0) cat (n1) sit (v1) on (x0) mat(n1)
2	Gods are willing to help	god (n1) be (v1) willing (a2) to (x0)help (v1)
3	Meat is eaten by piranhas	meat (n1) is (v1) eat (v1) by (x0) piranha (n1)
4	Cars are faster than motor bikes	car (n1) are (v1) faster (r1) than (x0) motorbike (n1)

Table 1 Given Criteria

Document Statement	Raw statement	Processed Statement
α	The feline rested on the covering	The(x0) feline(n1) rest(v1) on(x0) the (x0) covering (n2)
β	Devils are not wanting to aid	devil(n2) are (v1) not (r1) want (v1) to (x0) aid (v1)
χ	Fish consume pork	Fish (n1) consume (v1) pork (n1)
δ	Cheetahs can out pace a motor vehicle	Cheetahs (n1) can (v1) out (x0)pace(v2) a(x0) motor(n1) vehicle(n1)

Table 2 Document Sentences

Document Statements → / Criteria↓	α	β	χ	δ
1	**450**	**0**	**0**	**0**
2	**100**	**225**	**0**	**0**
3	**100**	**100**	**216**	**0**
4	**0**	**0**	**0**	**83**

Table 3 Classification results

XLibris Document Appliance

Gene Golovchinsky, Bill N. Schilit, Morgan N. Price

Abstract

The XLibris document appliance is a device designed to support reading. It is a combination of software and hardware (a pen- based computer) that implements the "paper document" metaphor: a tangible device, page-at-a-time display, and the ability to annotate documents with free-form digital ink. The paper document metaphor draws on peoples' familiarity with reading on paper, while the computer augments that activity in several ways. Users can annotate documents in XLibris by making free-form digital ink marks; two colors of ink and three highlighters are available. To help find annotated passages some time later, the "Reader's Notebook" view in XLibris displays just the annotations along with corresponding document passages. The list may be sorted by type of ink and by time. XLibris also uses the presence of annotations to identify users' interests, derives full-text queries from annotated passages, and constructs links to the best-matching documents. Thus XLibris supports serendipitous discovery of information while reading. Similarly, a combination of all annotations on a document can be used to find additional documents related to aspects of interest to the reader. Finally, XLibris implements a skim-reading mode that highlights terms are characteristic of the document being read.

The papers by Schilit et al (1998a, 1998b) and by Price et al (1998) describe this work in more detail.

References:

Price, M.N., Golovchinsky, G., and Schilit, B.N. (1998) Linking By Inking: Trailblazing in a Paper-like Hypertext. In Proceedings of Hypertext '98 (Pittsburgh, PA, June 20-23), ACM Press, 1998, pp. 30-39.

Schilit, B.N., Golovchinsky, G., and Price, M.N. (1998a) Beyond Paper: Supporting Active Reading with Free Form Digital Ink Annotations. In Proceedings of CHI 98 (Los Angeles, CA, April 19-26), ACM Press, 1998, pp. 249-256.

Schilit, B.N., Price, M.N., Golovchinsky, G. (1998b) Digital Library Information Appliances. In Proceedings of Digital Libraries 98 (Pittsburgh, PA, June 23-26), ACM Press, 1998. pp. 217-226.

JavaMiner: Non-linear Visual Browsing of Huge Java Documents for Program Understanding and Software Mining (System demo).

Mao Lin Huang, Peter Eades and Vladimir Estivill-Castro

Department of Computer Science and Software Engineering
The University of Newcastle
NSW 2308, Australia

{*mhuang,eades,vlad*}*@cs.newcastle.edu.au*

Abstract

We address the problem of visualizing and browsing structure in textual data. Our methods are suitable for very large amounts of data. We have instantiated our ideas in a system `JavaMiner` *for visualizing and browsing Java source libraries. The* `JavaMiner` *system can visualize a variety of relationships between terms in Java code; it can be used for "software mining".*

We visualize terms in the text as graphical nodes, and the relationships between them as links between the nodes. We use a new visualization technique called `Online Animated Visualization` *to deal with this huge graph. It can dynamically display the graph which is unable to be fit in the screen at a time.*

Our technique builds the visualization incrementally, as the user is exploring the huge graph. It allows the user to browse the huge graph without requiring the whole graph to be known.

The view of the user at any point in time consists of a picture of a small subgraph P of the huge graph G. This subgraph is defined by a set F of "focus nodes". These are the nodes that currently hold the user's attention (see Figure 1). The subgraph P includes the focus nodes, all neighbors of the focus nodes, and all edges between these nodes. The way in which the graph P changes is defined by an updating policy.

The layout of P is computed using a force-directed graph drawing algorithm, and we use multiple animations to guide the user smoothly between pictures. Our system also allows backtracking through the "history" of the browsing by displaying previously visited focus nodes (see Figure 1).

Browsing text using our system is "non-linear" in the sense that we can move smoothly and directly from one part to another, as opposed to scrolling up and down (in a "linear" way) through text.

Proceedings of the Third Australian Document Computing Symposium, Sydney, Australia, August 21, 1998.

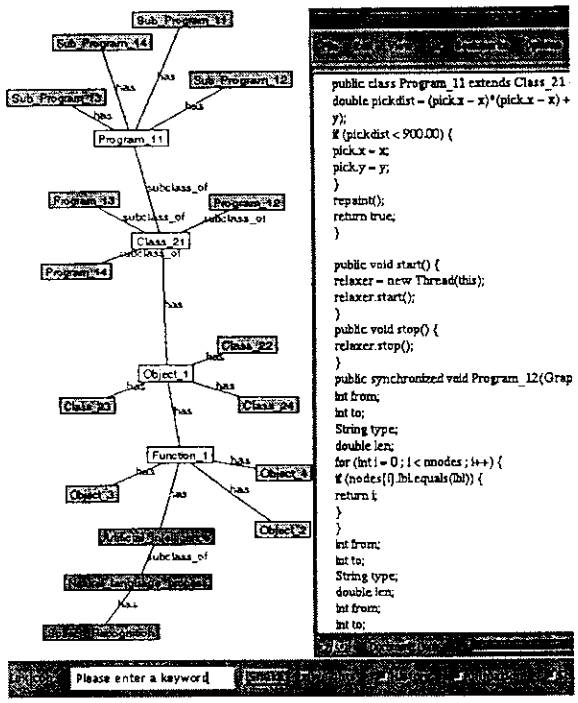


Figure 1: This subgraph P_7 has a set, F_7 = {`Function_1`, `Object_1`, `Class_21`, `Program_11`} Notice the history tail of the browsing consisting of nodes, `Natural_language_process`, `Artificial_intelligence` and `Speech_Recognition`.

The challenges in creating pictures of these graphs include:

- *Ensuring that the resulting diagram is readable, and*
- *Smooth transitions from one picture to another.*

This demo presents a framework for visualizing relationships between terms in Java documents. In particular, we answer the two challenges above.

Horses for Courses.
Fusing Dynamic and Static Web Sites

James Uther

DEDE
University of Sydney
Sydney, NSW, Australia

Hemul@gmp.usyd.edu.au

Vicki Taylor

DEDE
University of Sydney
Sydney, NSW, Australia

Vickit@gmp.usyd.edu.au

Abstract

The Graduate Medical Program (GMP) operates a large ($>10^4$ document) web site to support teaching and organisation. The management of such a large site has proved challenging. DBMS site management tools exist but their limitations have caused us to avoid them. We explore these limitations and some planned solutions.

Keywords Document Databases, Document Management, WWW, Internet, Multimedia.

1 The GMP Web Site

The Graduate Medical Program operates a web site to support it's teaching and organisation needs. As a Problem Based Learning (PBL) course, the GMP requires the broad distribution of resources such as x-rays, patient interview videos, pathology reports, ultrasound videos etc. These must be distributed not only to workstations on the main campus, but also to machines in clinical schools across the state connected via a wide range of bandwidths, and even to students in rural placements communicating across modem links. Consequently, the resources are static, enabling efficient caching at the destination. Additionally the site supports a number of applications including an online assessment system for students to gauge their progress against a bank of questions supplied by the faculty, and an involved feedback system that captures and routes comments on the site and course. These involve heavy use of databases and dynamic web pages.

In addition to our central web site and database servers, we use six caching servers at major clinical schools to ensure reasonable quality of service even over the often slow and clogged hospital Internet connections.

2 Current Art

The most common method of web site development is to author web pages in some form of editor, and place the pages under the document root of a web server, which then reads the pages from the filesystem. By not parsing pages, and by fully supporting the caching capabilities of HTTP/1.1[1], such sites operate efficiently. They are also suited to ad-hoc web sites, or sites with little structural similarity between pages.

An alternative to static web sites is to place the web site in a document database, and use explicit structure and relationship information to solve many of the management issues. This solution has been popularised by many (often DBMS) vendors. A prime example is Lotus Domino [2], which exports a notes database to the web, translating documents to HTML on demand. In these systems a database stores the raw document content, and generates HTML pages from this content on request. Templates are used to define page types, and so to author a page simply requires choosing a template, and then filling in the fields required by that page type. Flat identification schemes or queries name the pages, rather than a filename, which avoids any renaming issues. Similarly, the use of referential integrity within the database can prevent the removal of a page that would cause a broken link. These systems excel in installations where the content of the pages already exists in something like a product database. Linking the database directly into the web site ensures the site is always current requiring no page maintenance.

In our experience with and evaluation of both methods of web site management and creation, we have observed problems and limitations with both.

3 Limitations of Current Methods

3.1 Static Sites

Allowing the system to disregard page structure and relationship information for the sake of efficiency can lead to difficult site management issues. For instance, changing the name of a page on the web site also involves changing all pages with links to the changed page. Similarly, deleting a page can produce a broken reference within the site. Although site management tools that attempt to mitigate these problems exist, such as SiteMill [3] and FrontPage [4], we have found that they have trouble with large sites. Additionally, changing the style of the site involves changing all documents. New standards such as Cascading Style

Proceedings of the 3rd Australian Document Computing Symposium,
Sydney, Australia,
August 21 1998.

Sheets (CSS) [5] address this problem, but still lack support in browsers. Proprietary solutions exist, but often fall into the problems inherent in dynamic sites.

3.2 Dynamic Sites

These systems require pages to follow an explicit structure (page type). Multiple page types may be defined, however in diverse and developing web sites such as ours the number of page types can start to become unmanageable in itself. Web sites undergoing a rapid revision cycle or with a large set of page types are difficult to model in these systems.

The web is slow. High speed Internet service providers such as @HOME and the Big Pond cable modem service rely on large caches of web content to ensure the download speeds their customers are paying for. We use the same system in our remote sites. In essence, as our users retrieve documents, a copy is stored on a server on the local high-speed local network. When the next user requests the document, our cache server sends an 'If-Modified-Since' header to the main web server to see if the document has changed. If not, the local copy is sent at LAN speed. Only if the document has changed does the cache server request the new version from the main web server. If that document is a five-megabyte video, the cache server allows a download time of thirty seconds instead of five minutes.

In many cases, systems that offer dynamic sites use weak web servers that break the caching functions of HTTP/1.1, or are not programmed to reply sensibly to an 'If-Modified-Since' request. We require good quality of service at the end of slow links, and achieve this with cache servers. Any web server we use must reply sensibly to an 'If-Modified-Since' request.

We also make copies of the web site on CD-ROM and require the site to look like a filesystem, and many of these systems use URLs that are illegal in some filesystems. Others use URLs in a flat naming scheme, which in cases such as ours with many documents could expose limitations in the filesystem's handling of large directories.

Finally, such sites only accept documents that adhere to the set page structures. This usually requires a proprietary authoring tool, and these tend to be restrictive.

4 Solutions

The solutions are largely a matter of choosing the right tool for the job. This section will detail some of the parts of our web site, and how we have overcome the problems described above, or how we plan to overcome them in the future.

Our online assessment system is a classic example of a dynamic web site. It's just a database application. We have made no effort to enable our cache servers to store the basic pages, as they are by definition different for each request. On the other hand, we have made sure that the dynamic data is small (a few lines of text), and that any large bodies of data, like images for navigation buttons, are taken from a static part of the web site and may be cached. In this way, we have ensured that the application is useable even over slow links.

Our test result pages present many possible resources, and each page has been hand authored to best display the content. For instance, although a pathology slide and an x-ray are both images, clinicians require different displays of each. X-rays are particularly difficult to display well, requiring many edit-review cycles with radiologists before the staff are satisfied with the presentation. These pages are also large, and so very cache-dependent. These pages exist on a static web site and we have no immediate plans to move them.

We have a middle ground as well. The course consists of 540 'Learning Topics'[6], each of which consists of a title, author, keywords, 1-2 pages of topic summary, and references. The format is standard. Although these pages do not change often, we can see the advantages in storing the information in a database and producing the pages when required. The problem has been to ensure these documents are copied on our caching servers, and that the generated URLs look like they come from a reasonable filesystem. Our solution (currently in design) will use message digests and PathInfo information.

PathInfo is information passed to a web executable (servlet, CGI, etc) that describes path information in a URL beyond the name of that executable. For instance, if we have a servlet called
http://webserver.au/servlets/LearningTopics
but use a URL
http://webserver.au/servlets/LearningTopics/more/path
then the servlet LearningTopic.class is executed and given /more/path as it's PathInfo parameter. This simple mechanism allows us to use an executable to generate pages, but still access the site through a meaningful and useful URL namespace.

A more difficult problem is ensuring the servlet passes back a reasonable answer to an 'If-Modified-Since' request header. In this case, the client request headers contain an If-Modified-Since line and a date. It is requesting the document only if the document is newer than the date specified. The servlet must decide whether this is the case. In many simple cases where the page data is contained in a single row in a database, we obtain this information from the timestamp of that row. In complex cases involving multiple queries, this becomes prohibitively difficult and so we intend to implement a solution using message digests.

Message digest algorithms such as SHA-1 [7] or MD5 [8] process a body of data and produce a short string of bits that is probably unique to that data. For

example, the SHA-1 algorithm takes a file smaller than 2^{64} bits, and produces a 160-bit signature string from that file. To a very high probability, any changes to the file will change the signature. Similarly, there is a very low probability of ever producing a file with the same signature.

By storing a digest of generated web resources and the date of digest generation we can tell the date last changed of a resource in all cases. For instance, when resource X is first requested, we generate the page and send it, but also generate a SAH-1 digest of the page and store that along with a timestamp. When a request for resource X arrives with an 'If-Modified-Since' header, we regenerate the resource and generate a digest. If the digest and the previous digest don't match we update the digest and timestamp and send the page. If the digests match, we can then compare the digest timestamp and If-Modified header date and only send the page if the page is newer than the header. One further optimisation would be only to do the calculation every given timeout period, and cache the page within the generating servlet in the interim. This would be useful for high load pages with a low change rate.

5 Conclusion

Many solutions exist to the many problems of building and maintaining a large web site. Each solves a problem, but not all problems. The GMP web site thus uses a combination of solutions, each solving the most pressing problem of that area of the site, but with modifications to ensure that the solution to one problem does not cause more. We initially found that our needs for flexibility, development and caching precluded the use of database-based site management tools for the bulk of the site. As our site structure and requirements have firmed we have developed methods to move some of these pages to a database without compromising our performance.

Ackowledgements

The whole of DEDE, and in particular Stewart Barnet for providing requirements to keep us honest. Judy Kay for encouragement and bottom kicking.

6 References

[1] R.Fieding et. Al. Hypertext Transfer Protocol -HTTP/1.1
http://www.w3c.org/Protocols/rfc2068/rfc2068.

[2] Lotus Domino.
http://www.lotus.com/products/domino.nsf

[3] Adobe Pagemill.
http://www.adobe.com/prodindex/pagemill/pagemill20/siteben.html

[4] Microsoft Frontpage
http://www.microsoft.com/frontpage/

[5] Cascading Style Sheets.
http://www.w3.org/Style/

[6] The GMP Visitors Site.
http://www.gmp.usyd.edu.au/visitors/

[7] The SHA-1 Algorithm.
http://www.itl.nist.gov/div897/pubs/fip180-1.htm

[8] The MD5 Message Digest Algorithm.
http://theory.lcs.mit.edu/~rivest/Rivest-MD5.txt

Electronic Archiving – a 100 Year Experiment

Rhys Francis[1], Ross Gibbs[2], Leon Harari[3], Justine Heazlewood[2], Brendan Hills[1], Nick Leask[3], Ainslie Sefton[2], Andrew Waugh[1], Ross Wilkinson[1,4]

1 Division of Mathematical and Information Science, CSIRO Australia
2 Public Record Office Victoria
3 Ernst and Young
4 To whom correspondence should be addressed:
Ross.Wilkinson@cmis.csiro.au

Abstract

In this paper we describe the characteristics of archiving electronic records, and consider how to preserve these records forever. This is a pressing issue as enormous numbers of records are being created electronically but as yet there are only interim solutions being adopted, and no consensus has emerged on how to tackle this problem. A particular difficulty associated with this very practical problem is demonstrating a solution takes 100 years or more! How do we convincingly demonstrate that records have been preserved as they were created, and how can we find them? In this paper we consider a number of technical alternatives, and describe a prototype solution currently being created.

***Keywords**: electronic records, archiving, long term storage formats*

1 Introduction

In most document computing environments, the time between a document being created, and finally discarded, is sufficiently short that even if a new system is introduced, the nature of both the old and new is well understood. In this paper we describe the characteristics of archiving electronic records, and consider how to preserve these records forever. This is a pressing issue as enormous numbers of records are being created electronically but as yet there are only interim archiving solutions being adopted, and no consensus has emerged on how to tackle this problem. A particular difficulty associated with this very practical problem is demonstrating a solution takes 100 years or more! How do we convincingly demonstrate that records have been preserved as they were created, and how can we find them?

Record keeping has a long history. We know quite a lot about Egyptian life from its government records. Unfortunately much of government business is now transacted by electronic means and there is no simple method of keeping track of this business on stone! Records management systems were introduced to keep track of the paper records of government, and have gradually introduced means of keeping track of electronic records. However, the solutions that have been adopted generally are based on the data residing in a given system, with a specified underlying database. This is fine for the life of the record management system, but to archive records we have to store them beyond the life of several record management systems. It is for this reason, that we consider, how do we make a record available and readable in 100 years?

In this paper we will first introduce electronic records and consider their principle characteristics, and then briefly introduce the discipline of archiving. Since we are concerned in this paper only with longevity of electronic records we will consider only some of the activities of archiving. We then go on to describe the methods that might be pursued in keeping electronic records forever, and describe the advantages and disadvantages of these approaches. We describe the particular approach that is being prototyped at the Public Record Office Victoria. Finally we analyse the various approaches that can be taken and provide recommendations on which approaches seem to

Proceedings of the Third Document Computing Symposium, Sydney, Australia, August 21, 1998

be appropriate to achieve long term electronic records archiving.

2 Electronic Records

Records are a means of recording information about some event. In government, records are often evidence of some business transaction or decision. They will consist of one or more documents, and the associated contextural information, kept in a form that allows the information to be used as evidence that will stand up in a court of law. There have been several very careful examinations of the nature of records, and in particular the nature of electronic records. There are two notable, and competing, views. Simplified, one view is that of Duranti[4], who says that a record is an object that has been put aside because of its importance. Another view is that of Bearman[1], who says that a record is evidence of a business transaction. A useful comparison between approaches can be found in Marsden[4]. While the views do compete, particularly with regard to what objects should be preserved as records, there is considerable agreement over the metadata that is stored as part of a record.

Independent of this debate, we can view a record as one or more documents, each with their associated metadata, plus the metadata of the record, plus means of establishing the integrity of a record. We could thus illustrate a record as in Figure 1.

3 Archiving

The principal role of archiving is to store records, potentially indefinitely, and make them available to those entitled to access them. There are other important functions necessary to support this role: defining disposal schedules, advising on record keeping, sentencing records, destruction of records, etc. however we shall not consider the impact of these activities in this paper, other than to note that the information needed to make these decisions should be kept with the records. To understand the broader role of archiving, one might consult Pederson[6]. When records are stored in an archive, there are three key challenges:

- to ensure that the record is understandable and readable indefinitely, even though the system that created the document that forms part of the record has long since disappeared,
- to provide means of finding records pertinent to those who wish to find them, and
- to ensure the record maintains its evidentiary status.

To address the first issue, we must ensure that the document description language that is used in the document is not tied to a particular product or system. Thus saving a document in a word processing format is likely to be less helpful than storing in a structured language like XML, or a presentation format like Postscript. However, simply being able to read the document is only part of the story. In order to understand the document, it is equally important to capture the context of the document, generally as metadata. This capture needs to occur at the time that the document is produced, because often context is lost or expensive to recapture at a later time.

To find records, local context alone may not help. If somebody wishes to explore very large collections of information stored long ago, it is often helpful to have other access paths developed. In the archival world these are called finding aids, and help users navigate down through hierarchies of records, that might be organised into several levels above individual documents. Interestingly, there has been very little investigation of content based search. This

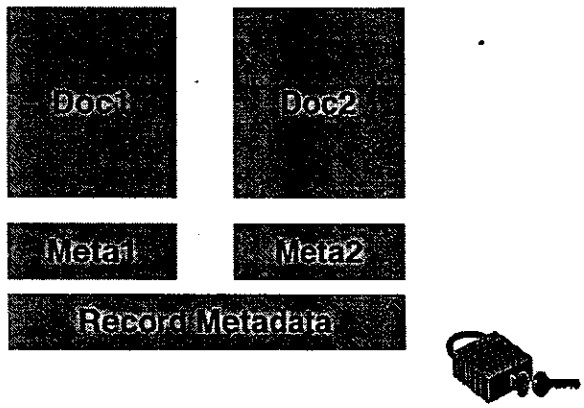


Figure 1

has not been possible for archives to date, as they have principally been paper based, and it is unclear what form of content search will be desirable for electronic records.

To address the third issue, there must be mechanisms of ensuring that a record is what it purports to be. To ensure that an electronic document and its metadata are unchanged from the time of creation, it is possible to lock the record with an electronic signature. Note that due to the very long lasting nature of the records, quite strong locking is required in order to deal with attack. For example it may be that the signature is never accessible – only a message that reports on integrity is provided based on the signature check.

It is of course possible to use these methods whether or not a document is removed from a record management system to a separate archival system, but it is essential if they are removed.

4 Methods of Archiving Electronic Records

It would be convenient if all records were already in a format that supported the three properties mentioned above. However, the world is not like that - information is likely to be available as scanned images, word-processed files, emails, reports from databases, and many more complicated formats. What then are we to do with this information? We have several options available. In any of the solutions the integrity of the record is fundamental. The proposed options are methodologies to manage electronic records. Let us examine each of them, looking at the technical means of pursuing these options, their advantages and disadvantages.

4.1 Print

If we are dealing with viewable objects, we can simply print them. We then have the task of associating metadata to the paper representations, and filing them as one would have done to date. The advantage of this approach is that it fits in well to existing approaches to record keeping and archiving. However there are several disadvantages. The first is that the paper representation of an electronic object is different, with different properties, and has been rejected by U.S. District Judge Paul L. Friedman who said:

"Electronic communications are rarely identical to their paper counterparts.... Records stored electronically have search, manipulation and indexing capabilities not common to paper records"

If the record is different, what is its evidentiary status? The next difficulty is capturing the relevant metadata, but the most compelling difficulty is that the paper must then be stored. The cost of storing paper vastly outweighs storing electronic objects, and is thus increasingly beyond archival budgets.

4.2 Preserve Systems

The next option is to keep the records available in the software and hardware system in which they were created or used. This is ideal from the evidentiary point of view, as a later user can examine information in exactly the same way as the original users. The main problem with this solution is technical – computing systems are just not designed to last forever, or anywhere like 100 years. Thus there is an increasing cost in maintaining and replicating systems for each system that is in use in an organisation over a long period of time. This solution is quite practical for some classes of records – those records whose life does not extend much beyond the life of the system. However, it appears to be completely infeasible if an organisation expects to be creating long life electronic records.

4.3 Migrate Data

Another possible solution involves keeping all long term records in the organisation's current record keeping system. This involves migrating records each time the record keeping system is changed. This is an achievable solution as all records are migrated at the time of update. Over 100 years, this is likely to mean many changes indeed. There is thus considerable concern over mechanisms to ensure the integrity of records. How does one ensure the integrity of these records? First it is not possible to enforce standard integrity checks, such as checksums, and signatures, as the point of migrating is to change the records. The desire is to ensure their visual integrity, the continued accuracy of the metadata, and possibly their evidential integrity. The first can perhaps only done by people. The last can be done by very careful audit trails that explain exactly what the transformations that took place at each migration. In some ways, the preservation of the metadata may be hardest, as the metadata may be transformed and stored in databases where the semantics of the metadata may be implied by the structure of the database. In this case it can be particularly difficult to ensure the integrity of many migrations.

4.4 Preserve Data and Migrate Viewers

Most systems in which electronic records are created are quite complex, and migrating the systems to new environments is likely to be very complex. On the other hand, the software needed to view a Microsoft Word document, is considerably simpler to develop than that required to process a Microsoft Word document. In this vein, it is possible to imagine developing viewers for each of the data types that are used in an organisation's records, and as systems change and new data types are introduced, to maintain a current set of viewers for all data types in the archive. This solution has the advantage that there is no transformation of the data, and the corresponding software effort is minimized.

4.5 Long Term Data Format

A final solution is to transform the data into a long term format at the time of record capture. This solution has the disadvantage that there is a transformation but it occurs at the time of creation so that it can be validated at that time as well. This solution relies on there being a viewer of the long term format, but this can be assumed as long as widely accepted formats are adopted. Note that the solution does not require that the choice of long term format never changes. What it does require is that for each long term format chosen, there will continue to be a viewer for that format. This strategy is achievable, as we will describe in the next section, but is exposed to the risk that if many organisations adopt different long term formats, there is likely to be a proliferation of viewers required.

5 The VERS Prototype

A prototype is being developed by a team from CSIRO and Ernst and Young as part of the Victorian Electronic Record Strategy (VERS) being developed by the Public Record Office Victoria. The VERS prototype system has three goals:

1. To demonstrate capture of electronic records as close to the time of creation as possible by careful understanding of the organisations creating the records, and careful integration of workflow systems that support automatic metadata capture.
2. To demonstrate that it is possible to archive common record types found in Victorian and Australian government agencies in a cost effective manner. The goal is to 'pick the low hanging fruit', that is, it is not intended to demonstrate archiving of all possible types of records. Instead, the project will focus on records that can be viewed. Fortunately this includes all of the common records, which is not surprising as records to date have been paper objects. The set includes paper documents, word-processed documents, email, and a representative database. We are not attempting to archive more complex record types (e.g. CAD files). These will be left for future work. We believe that it is far better to start archiving electronic records than to wait until a perfect system has been developed.
3. To produce a specification of an electronic archival system. Agencies will be able to use this specification as part of the process of obtaining future records management systems and electronic document management systems. Vendors should be able to use both the specifications and the prototype to determine how they can satisfy the agency needs by providing such functionality within their products.

The VERS design is primarily based on the existing policies of the Public Record Office Victoria. The current process, for example, is based on series (of related files). Management of electronic records, therefore is primarily based on the series (and secondarily) on the files, and not on individual records.

The approach selected is to archive the records immediately the record is produced. The process of archiving involves a migration from the original document format (e.g. WORD) into a permanent archival format. The archival format has to be oriented to capturing the way the document looks. We have selected PDF as the permanent archival format as it is currently the best standard for describing pages. Although PDF is a proprietary standard, the standard has been published and it would be feasible to generate independent PDF viewers.

Migration always implies loss of functionality. Immediately migrating the record into the permanent archival format means that the organisation loses the ability to manipulate the record (e.g. use the record as a template for a new document). However records should not change. Documents, which form part of a record, but do need to be modified, must be kept in their original format in a separate document repository. Having two separate storage systems will increase the cost to the agency.

We feel that the benefits of immediately archiving records outweigh this cost. The digital signatures associated with an electronic record can only verify when the record was entered into the archive. They cannot protect a record against modification or forgery before the record has been archived. By archiving the record immediately it is created the window for modification or forgery is almost completely closed. Importantly, it is not necessary to depend on a second system, a document management system, say, for integrity.

The PDF representation of the document captures the content and structure of the record. The context of the record is captured as metadata. The metadata is stored with the document because storing the metadata separately leaves open the possibility of losing the metadata.

We have chosen the Resource Description Framework (RDF)[7] to represent the metadata. RDF defines a metadata model and syntax; it does not define the metadata semantics. The metadata to be captured with each record is based on the Pittsburgh model[1].

RDF uses XML (eXtensible Markup Language)[2] to represent the metadata. XML is a textual based markup system. It will consequently be possible to view the contents of a record using a normal text editor and directly read the metadata. XML tags are words (e.g. '<Title>') and so have inherent semantic meaning. This ability to directly examine the contents of records is an important component of the prototype solution. It makes it far easier to reconstruct the information in the record if the prototype system documentation is lost.

The VERS project is not taking a position on who will have custody of electronic records. The proposed solution will support keeping the records permanently in the organisation that created them, or the transfer of custody of records to an archival organisation, such as the Public record Office Victoria. This project does take a position on the technical aspects of how electronic records are represented and stored.

6 Analysis

In order to determine the appropriateness of an electronic archiving solution, we must consider a number of factors, which we draw from the earlier discussion. It is important to note that the same factor may have quite different characteristics in the short and long term. These factors are:

1. Ease of implementation in short term
2. Ease of implementation in long term
3. Cost in short term
4. Cost in long time
5. Evidential status in short term
6. Evidential status in long term
7. Support for access to records
8. Need for agreement

We now provide an initial analysis of each of the solutions that we have discussed in the light of these factors. As shown in Figure 2.

Based on this analysis, it seems clear that the most appropriate solutions are the data viewer solution and the long term data format solution. The choice of one or the other of these solutions depends on the level of agreement that is achievable in achieving agreement on long term data format. If this format is possible, then it appears the desirable solution. If this is not the case then developing a suit of data viewers appropriate to the range of data media seems appropriate.

FACTORS	Print	Preserve	Migrate	Viewers	LT Data
Implementation-S	Medium	Easy	Easy	Easy	Easy
Implementation-L	Medium	V. Hard	Hard	Moderate	Easy
Cost-S	High	Low	Low	Low	Low
Cost-L	V. High	V. High	Medium	Medium	Low
Evidence-S	Poor	Good	Good	Good	Good
Evidence-L	Poor	Good	Poor	Good	Good
Access	Poor	Poor	Good	Moderate	Good
Agreement	Low	Irrelevant	Irrelevant	Irrelevant	High

Figure 2

7 Conclusions

In this paper, we have described the problem of archiving electronic records. We have described a set of technical solutions that enable archiving, and analyzed their strengths and weaknesses. By looking at the requirements of long term electronic archiving, we are able to determine two solutions that appear to have clear advantages over others. We have described a project to develop a prototype that demonstrated electronic record capture and archive, and shown how this prototype implements one of the solutions.

References

[1] D. Bearman, J. Trant *Electronic Records Research Working Meeting: A Report from the Archives Community*, D-Lib Magazine, July/August, 1997 <http://www.dlib.org/dlib/july97/07bearman.html >

[2] D. Connolly, Editor, *XML: Principles, Tools, and Techniques*, World Wide Web Journal: Volume 2, Issue 4, 1997

[3] J. E. Dryden, *Archival description of electronic records: an examination of current practices*, Archivaria (40), pp.99-108, Fall 95.

[4] L. Duranti, *The Archival Bond*, Archives and Museum Informatics, vol.11, nos. 3-4 , pp.213-218, 1997

[5] P. Marsden, *When is the Future? Comparative Notes on the Electronic Record-Keeping Projects of the University of Pittsburgh and the University of British Columbia*, Archivaria, Vol. 43, pp. 158-173, 1997

[6] A. Pederson, *Keeping Archives*, Australian Society of Archivists Inc., 1987

[7] W3C, O. Lassila, R. Swick (eds.), *Resource Description Framework (RDF) Model and Syntax* <http://www.w3.org/TR/1998/WD-rdf-syntax-19980216>, 1998

Searching the World Wide Web Made Easy? The Cognitive Load Imposed by Query Refinement Mechanisms

Simon Dennis
School of Psychology; University of Queensland
Brisbane, 4072, Australia
s.dennis@psy.uq.edu.au

Robert McArthur
Distributed Systems Technology Centre, University of Queensland
Gehrmann Laboratories, Brisbane, 4072, Australia
mcarthur@dstc.edu.au

Peter Bruza
School of Information Systems, Queensland University of Technology
GPO Box 2434, Brisbane, 4001, Australia
bruza@icis.qut.edu.au

Abstract

This article addresses the effectiveness of search reformulation using query refinement mechanisms on the Internet. Cognitive load was measured using a secondary digit-monitoring task. The load was found to be lower when using the refinement mechanisms than when perusing document summaries - suggesting that the development of refinement mechanisms can make Internet searching easier. Two refinement mechanisms, one based on statistical term co-occurrence and the other on a shallow natural language parsing technique were tested. No difference in load was found, possibly because of the limited time that subjects spent in the refinement process.

Keywords: information retrieval evaluation, internet searching.

Introduction

Short queries on the WWW result in large, imprecise result sets. If longer queries could be elicited from the user, then the precision in the retrieval could be improved. For this reason a number of query formulation aids have appeared in conjunction with web-based search engines. For example, the AltaVista and Excite search engines produce lists of keywords that the user can use to make the initial query more specific. Similarly, the Hyper Index Browser[1] (HIB) provides the user with a list of phrases that include the initial query terms and that can be used in subsequent calls to the retrieval engine.

The challenge for query refinement mechanisms is to provide an interface to the search space that makes searching easy and productive for the user through the careful selection of information to present to the user. In this paper, we intend to address two key issues.

Firstly, what are the merits of remaining in the refinement processes rather than perusing document summaries? The commitment to query refinement mechanisms hinges on the assumption that it is easier and more productive for users to process the refinements than to peruse a list of document summaries. Perhaps, however, there is no substantive difference between refinement and summary processing and the query refinement mechanisms are unlikely to lead to more effective search.

Secondly, how should a refinement mechanism choose candidate refinements for the users perusal? AltaVista and Excite produce candidate keywords for query refinement using term co-occurrence statistics. The refinements selected are words that tend to occur with the target terms. By contrast, the Hyper Index Browser (HIB) selects phrases that contain the initial query. For example, if the query is "Internet", refinements like "Internet security", or "guide for Internet" are presented. The second question we will address is whether the statistical or linguistic approach to refinement generation leads to more effective search.

In the discussion above, we have mentioned terms like "easy", "productive" and "effective" in conjunction with search. There are a number of dependent variables we might use as measures of these factors. We could use the traditional

[1] http://www.dstc.edu.au/cgi-bin/RDU/hib/hib

Proceedings of the 3rd Australian Document Computing Symposium,
Sydney, Australia,
August 21 1998.

information retrieval measures such as recall and precision. However, these measures have a couple of important limitations that should be noted. Firstly, our domain of interest is the World Wide Web where recall is difficult to measure accurately because it relies on identifying all of the relevant documents in a collection. Secondly, and perhaps more importantly, recall and precision tell us little about the demands placed upon the user while refining his or her query. We would prefer a measure that captures more closely the users search experience.

One possibility is to consider the amount of time that a user spends on a search task. Long searches might be considered less effective. In addition, however, we can look to the human factors literature for methods that more directly reflect the amount of cognitive load experienced by the user.

Cognitive Load and Dual Task Methodology

There are a number of reasons why the direct measurement of search time may not accurately reflect the amount of effort employed by the user. Firstly, users are capable of employing greater cognitive resources in response to task demands (Norman & Bobrow [9], Eysenck & Keane [6], Wickens [10]). Provided the user can command ample resources to complete the search task, an inferior engine might require the user to apply effort which is not reflected in the amount of time they spend on the task. Secondly, two search mechanisms might provide different amounts of data to the user (Wickens [10]). The engines we will consider in this article (the HIB and Excite) typically generate different numbers and types of refinements. Extra time might be spent with the HIB because it generates a larger number of refinements, but each of those refinements may be easier to process requiring less effort from the user. Again, total time may not be sensitive to such differences.

For these reasons, we chose to employ the secondary task, or **dual task** technique. The dual task technique has a long history in the human factors literature (see Wickens [10] for a summary). The idea is to have subjects do two tasks simultaneously, to use any excess resources that might be available. The amount of effort they employ on the primary task (i.e. the Internet search) is inversely proportional to their performance on the secondary task.

A large number of secondary tasks have been employed, from finger tapping (Michon [8]), to random number generation (Baddeley [1]). After piloting, we chose a digit-monitoring task in which subjects listened to a stream of digits (from one to five) and responded when a digit was repeated. This task seemed to impose a considerable but not overwhelming load on the subjects and provided two simple measures of secondary task performance: reaction time and the miss rate.

When employing dual task methodology, it is important to keep in mind its limitations. Firstly, current theories of human resources contend that there are multiple pools and that tasks that draw on different pools will not necessarily impact upon each other. In this case, the secondary task performance may not be sensitive to differences in primary task demands. Secondly, the secondary task should not precipitate a change in the strategy subjects use on the primary task. For instance, if it were the case that the digit-monitoring task caused subjects to process only a couple of refinements, where they would have processed many more had they not been required to perform the secondary task, then we are no longer measuring the task of interest. One of the primary purposes of this research is to determine the extent to which the dual task methodology and this particular secondary task are suitable for measuring search engine performance.

For the purposes of this study we assume that *there is a relationship* between cognitive load and retrieval effectiveness. Consider when a user has a high cognitive load, then it is likely that (s)he may miss potentially relevant information, thus affecting recall. The detailed nature of the relationship between cognitive load and retrieval effectiveness is a large research question. The study in this paper makes a small step in fleshing out this area.

Refinement versus Summary Load

Our first main hypothesis is that cognitive load is lower when the user is viewing candidate refinements (whether statistical or linguistic) for a query than when they are viewing document summaries. Document summaries typically include a list of candidate document titles with a small automatically generated precise of each page. For most queries, this list is long, often in excess of 1000 documents, and searching the summaries can be tedious. Furthermore, evaluating each of the summaries is difficult because the process of precis extraction is imperfect and is not customized to the user's information need. In contrast, the refinement mechanisms provide a relatively small number of candidate queries, and these candidate queries are typically only a few terms in length making them potentially easier to process.

Statistical versus Linguistic Refinements

The second of our main hypotheses is that linguistic refinements will decrease cognitive load and improve effectiveness of search as compared to statistically generated refinements. In this section, we will outline

how both statistical and linguistic refinements are generated.

Generating Statistical Refinements

If terms co-occur frequently, then there must be a relationship between them. This hypothesis is the basis of automatic thesaurus construction, but can also be used to produce candidate terms for query refinement. An often-employed technique is based on the vector space model. Given a query vector, term vectors are ranked according to similarity with the query vector. Those term vectors above a given threshold are deemed sufficiently similar to the query vector to warrant displaying as candidate terms for query refinement.

The Excite search engine uses the vector space model (Cutting [5]). For this reason it is reasonable to assume that Excite employs the above strategy (or variant thereof) to produce candidate terms for query refinement and we used Excite to generate the statistical refinements to be used in the experiment.

Generating Linguistic Refinements

The Hyperindex Browser (HIB) is a query formulation tool which attempts to produce reformulations of a query as linguistically well formed expressions (Bruza [2]; Bruza & van der Weide [4]; Bruza & Dennis [3]). The user enters an initial query, which is passed onto a retrieval mechanism, and a summary set is generated. The documents in the result are not presented directly to the user but are analyzed using a shallow natural language parsing technique (see also Greffenstette [7] for a related technique). Phrases, called index expressions, are derived from the documents in the result set and displayed to the user as candidate refinement possibilities for the initial query. For example, a query "surfing" would produce refinements like "surfing in australia", "internet surfing", "wave ski surfing". The user can then select any of these refinements to become the new query.

The Research Browser

The Interface

For the purposes of measuring cognitive load and timing users when refining queries or perusing documents or document summaries, an experimental web browser interface was constructed.

There are six buttons prominent above the HTML window - Back, HIB, Excite, Bookmark, Start/Finish and Stop (see Figure 1). The Back button returns the subject to the previous Web page. The HIB button takes subjects directly to the HIB starting page (see Figure 1), while the Excite button takes subjects to the Excite starting page. Each of these pages allows the subject to enter a set of search terms.

Clicking Search presents a page of refinements (see Figure 2). Subjects can choose to click on the focus (the terms that got them to the current page), click on a refinement, click on an enlargement, or type in new search terms.

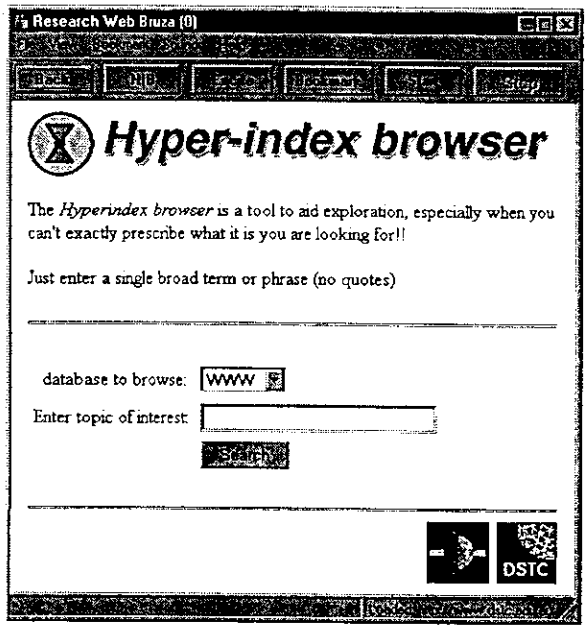


Figure 1: HIB Start Page

Clicking on the focus link moves the user into the document summaries - the focus is used as a query and the resultant document summaries are presented to the user. In this experiment, the Excite search engine was employed to retrieve document summaries. Activating a refinement or enlargement link modifies or replaces the current query under focus with the refinement or enlargement that was selected. Refinements are modifications of the query which make it more specific. Enlargements broaden the query.

The Bookmark button saves the URL of the current page. The Start/Finish button starts and stops the timing of a query session.

Audio files are played to the subject at random intervals (3.5 seconds ± 1.5 seconds). The subject hears a voice saying a random number between one and five inclusive. The system forces a repeat to be played - that is, one digit played that is the same as the immediately preceding digit - every five digits played. Consequently, there was a maximum of 17.5 seconds between the playing of repeated digits.

When a subject clicks on any of the first three buttons, or on a link within the HTML page, all buttons except the Stop button are disabled until the entire page is loaded from the network, parsed, and is

displayed in the HTML window. Consequently, subjects cannot begin processing the page until it has been completely loaded, allowing us to distinguish when users are searching and when they are waiting for network transfers.

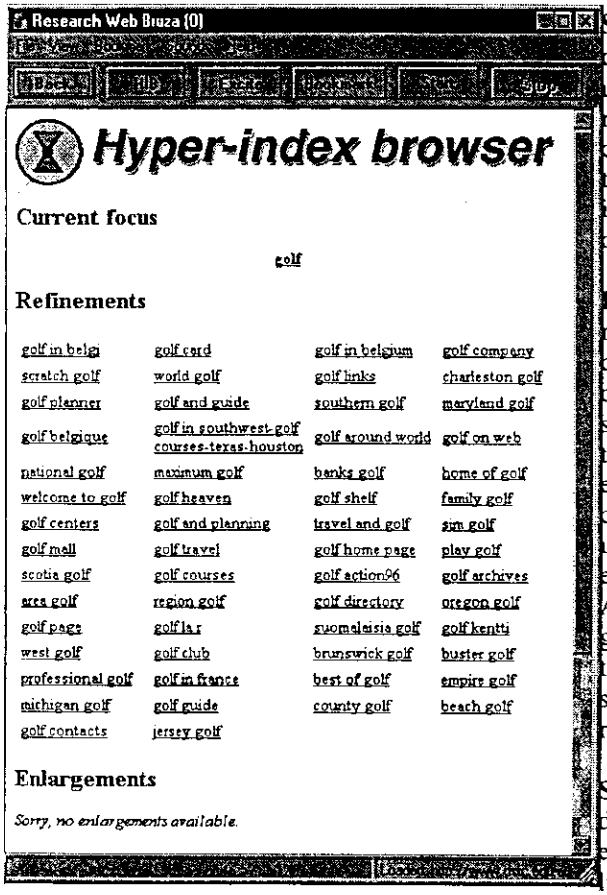


Figure 2: HIB Refinement Page

The Search Engine State Diagram

When using browser interface, the user is residing in one of four possible states, or is in transition between these states (see figure 3).

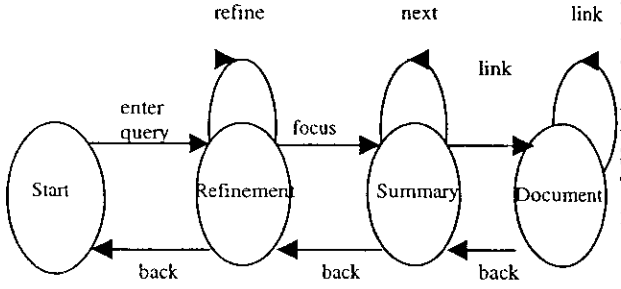


Figure 3: The search engine state diagram.

The possible states and their transitions are:

Start State In this state, the user enters an initial query. This always leads to a refinement state when using the Excite browser, but can lead to both a refinement (see Figure 2), or summary state in the case of the HIB browser. The HIB automatically transports the users to the document summaries when it cannot generate any refinements of the query description under focus.

Refinement State In this state the user sees a focus representing the current query description and candidate refinements of this focus, each of which can be selected to make the current focus more specific. The HIB also has enlargements, which allow the focus to be broadened. (This facility was not employed during the experiments). The current focus can also be selected. This results in the focus being used as a query to return document summaries. This event moves the user to the summary state. Alternatively, if the user clicks on a refinement generated by the HIB that cannot be further refined the user is transported automatically to the document summaries retrieved by using the activated refinement as query.

Summary State In this state, the user peruses document summaries retrieved from the Excite search engine, which consist of a title, short abstract, and links to the actual documents. The link can be activated to transfer the user to the document state.

Document State The state in which the user is perusing a web document. If found relevant it can be bookmarked. Alternatively, the user can follow a link to another document.

In all states, the back button can be activated to transfer the user to the previous state. In addition, the start state is accessible from any state if the user wishes to start afresh. The document state is a termination state when the user has bookmarked a document that satisfies the information need.

Any state can be a termination state due to a time out. Figure 3 depicts the search engine state diagram. It provides the conceptual framework for interpreting and discussing the experimental results.

Experiment

Subjects and Design

Ten subjects, two female and eight male, participated in the experiment for payment. In general, the

subjects had substantial computing experience. Seven use a computer daily and all had been using computers for more than a year. Similarly, most of the subjects had substantial experience on the Internet. Six used the Internet daily and only two indicated that they did not use the Internet on a regular basis. None of the subjects had used the HIB previously, but five had used the Excite search engine. When asked about their experience with the engines after the experiment, four preferred the Excite engine, three the HIB and three were uncertain. It should be noted, however, that both engines used the Excite summary mechanism and some subjects commented that had found it difficult to recall which engine they were using for some questions.

A 2x2 factorial design was employed. The factors were the refinement strategy used (statistical/Excite or linguistic/HIB) and the search state (either refinement or summary). Both factors were within subjects.

Procedure and Materials

Each subject began with a tutorial on the use of the research Web browser. This tutorial explained each of the buttons on the interface (see the description above) and provided the subjects with practice at performing the dual tasks.

Then each subject attempted to answer eight questions (see appendix A). Half of the subjects used the HIB for the even numbered questions and Excite for the odd numbered questions, while the other half used the HIB for the odd numbered question and Excite for the even numbered ones. The questions were arranged in this way to control for possible order effects. To answer a question the subject bookmarked a page that they felt contained the relevant information.

While using the engine to answer a question, the subjects were required to monitor a series of digits, which they heard through headphones. The digits ranged from one to five and the subjects were instructed to click the right mouse button when they heard a digit repeated. Reaction times were measured from the start of the playing of a repeated digit to the right button click. Subjects were informed that the digit-monitoring task was important and they should endeavor to click the right mouse button as quickly and accurately as possible.

Finally, the subjects were asked to fill in the subject information questionnaire.

Results

Figure 4 shows the time subjects spent in the refinement and summary states as a function of the refinement strategy. Subjects took longer in the summary states than they did in the refinement states, $F(1,8)=22.84$, $p = 0.001$, but there was no difference between the time as a function of search engine $F(1,8)=0.727$, $p = 0.419$, and no interaction $F(1,8)=0.105$, $p = 0.754$.

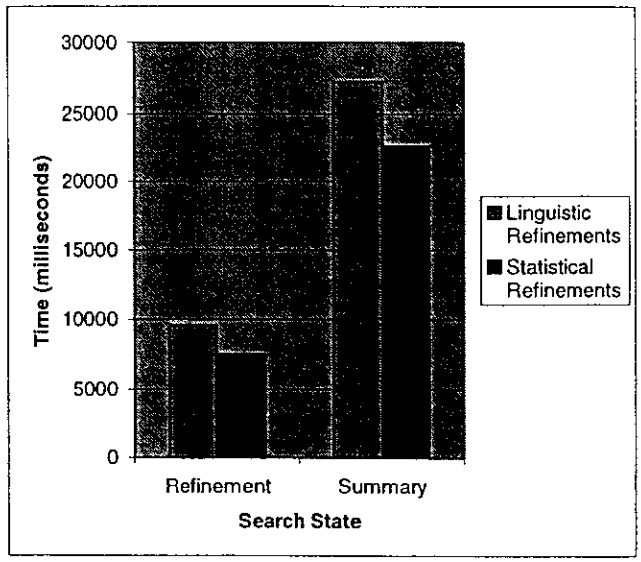


Figure 4: Reaction Time as a Function of Search State and Refinement Strategy.

Sampling of the subjects reaction times to the repeated digits was done randomly and consequently some cells were missing in the analysis. For this reason, it was decided to analyze each reaction time as a separate event (rather than doing it on a per subject basis) and to make comparisons between the HIB and Excite refinement reaction times and the refinement and summary state reaction times.

No difference was found between the HIB (M=1867 milliseconds) and Excite (M=1800 milliseconds) refinement reaction times $F(1, 39)= 0.029$, $p=0.865$. Nor was there a difference between the reaction times as a function of refinement (M=1823 milliseconds) versus summary state (M=1885 milliseconds) $F(1,186)= 0.059$, $p=0.808$. There was a significant difference between the transition reaction times and the states reaction times (Transition M= 1283, State M= 1847, $F(1, 9)=21.986$, $p =0.001$), indicating that reaction time was faster when processing requirements were low. Log reaction times were also examined because reaction time distributions tend to show substantial skew, which can undermine the assumption of the analysis of variance technique. However, the log reaction time analyses required no change to the interpretation, so the raw scores are reported.

In addition to reaction time, one can examine the number of times subjects fail to respond to a double

digit as an indication of cognitive load. The miss rate was calculated as the number of misses divided by the total number of double digits played (the number of hits plus the number of misses). There were few misses in the refinement state, so it was not possible to compare the linguistic and statistic refinement mechanisms. It was, however, possible to look at the miss rate as a function of state (see figure 5).

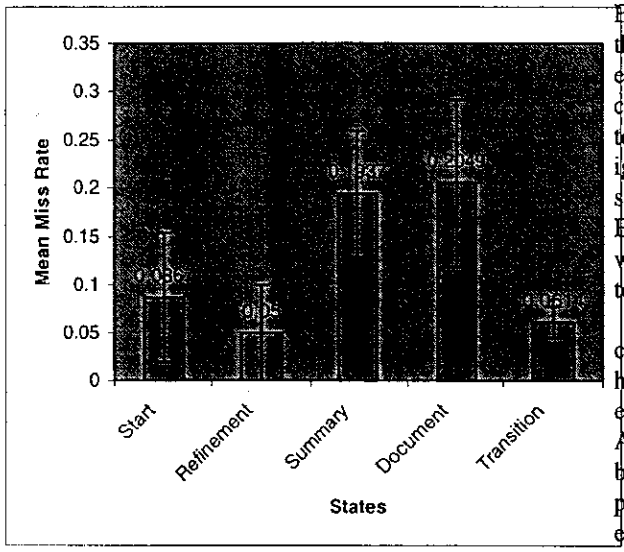


Figure 5: Mean Miss Rate as a function of State. The bars indicate the standard error.

Using the Wilcoxon matched-pairs signed ranks test there was a significant difference between the miss rate during the refinement state and the summary state ($T=3$, $p < 0.05$).

We were also interested in whether there would be a difference in search outcomes as a function of the refinement strategy. The mean number of bookmarked items was 0.92 for Excite and 1.12 for the HIB. This difference was not significant, however, $F(1, 69)=0.784$, $p=0.379$. We also looked at the total time spent on queries as a function of refinement strategy. Again there was no difference $F(1,69)=0.330$, $p=0.567$.

In an effort to determine to what extent subjects were using the refinement engines we examined the breakdown of the chains as a function of chain length. On the majority of occasions (62%) subjects did not refine their queries.

Discussion and Conclusion

In the introduction, we outlined two main hypotheses. The first of these, that refinement mechanisms reduce cognitive load, was supported. This result provides a sound basis for further work in the area. Refinement mechanisms can make searching the Internet easier.

However, we were unable to find support for the second hypothesis, that linguistic refinements induce less load than statistical refinements. Our ability to discern the load associated with statistical and linguistic refinements was hampered by subjects reluctance to use the refinement mechanisms. Regardless of whether they were using the HIB or the Excite refinements subjects spent on average less than ten seconds considering refinements. After entering an initial query, which produced a page of candidate refinements, they often proceeded directly to the document summaries. The users could not ignore the refinements as these were displayed on a separate page to the summaries, and in the case of Excite (and on most occasions for the HIB) users were required to view the refinements before moving to the summaries.

One possible explanation is that the users were not convinced about the benefits of refining. They may have been unaware of the connection between a more expressive query and precision in the retrieval result. Also, it may be that the subjects have been influenced by previous exposure to search engines that only provide summaries. An interesting avenue for further exploration is to test whether novice users or users with additional training employ the refinement mechanism.

This paper makes two main contributions. Firstly, the current results suggest refinement mechanisms do reduce the cognitive load experienced by the user.

Secondly, we have demonstrated the use of dual task methodology in the assessment of refinement mechanisms. The digit-monitoring task has been shown to be appropriate as a secondary task and both reaction time and miss rate are sensitive to load variations while conducting internet search tasks.

Acknowledgements

The work reported in this paper has been funded in part by the Cooperative Research Centers Program through the Department of the Prime Minister and Cabinet of Australia.

References

[1] Baddeley, A. (1966). The capacity for generating information by randomization. Quarterly Journal of Psychology, 18, 119-130.

[2] Bruza, P.D. (1990). Hyperindices: A Novel Aid for Searching in Hypermedia. In A. Rizk and N. Streitz and J. Andre, editors, Proceedings of the European Conference on Hypertext - ECHT 90, pages 102-122, Cambridge University Press.

[3] Bruza, P.D. and Dennis, S. (1997). Query reformulation on the Internet: Empirical Data and the Hyperindex Search Engine. In Proceedings of the

RIAO97 Conference - Computer-Assisted Information Searching on Internet, 488-499, Centre de Hautes Etudes Internationales d'Informatique Documentaires.
[4] Bruza, P.D. and Weide, Th.P. van der (1992). Stratified Hypermedia Structures for Information Disclosure. The Computer Journal, 35(3):208-220, 1992.
[5] Cutting, D.A. (1997). Space Optimizations for Total Ranking. In Proceedings of the RIAO97 Conference - Computer-Assisted Information Searching on Internet, pages 401-412, Centre de Hautes Etudes Internationales d'Informatique Documentaires.
[6] Eysenck, M. W., & Keane, M. T., (1995). Cognitive Psychology. Lawrence Erlbaum, Hove, UK.
[7] Greffenstette, G. (1997). SQLET: Short Query Linguistic Expansion Techniques, Palliating One-Word Queries by Providing Intermediate Structure to Text. In Proceedings of the RIAO97 Conference - Computer-Assisted Information Searching on Internet, 500-509, Centre de Hautes Etudes Internationales d'Informatique Documentaires.
[8] Michon, J. A. (1966). Tapping regularity as a measure of perceptual load. Ergonomics, 9, 401-412.
[9] Norman, D. A. & Bobrow, D. G. (1975). On data-limited and resource limited processes. Cognitive Psychology, 11, 44-64.
[10] Wickens, C. D. (1992). Engineering Psychology and Human Performance, Harper Collins: NY.

Appendix A: Questions

1) You are planning to move to Florida. Use the engine to find two pages listing jobs in the Florida area.
2) You need to replace a piece of pipe below the kitchen sink. Find two companies that could supply what you need. Bookmark their homepages.
3) Find three pages providing recipes for different varieties of carrot cake.
4) Find two pages listing indicators that a person has dyslexia?
5) Find a page that describes what happened to the singer Melanie who was popular in the 1960s.
6) Fine one page describing each of the two major streams of Zen Buddhism in Japan (one page for each stream is sufficient).
7) Find a page that clarifies the term "wiking" in the context of world war two Germany.
8) Find a page listing the current rankings of male tennis players on the ATP tour.

Meeting Log Analysis and Synchronous, Dynamic Document Derivation in Computer-supported Meetings

Gitesh K. Raikundalia

School of Multimedia and Information Technology
Southern Cross University
Coffs Harbour, New South Wales, 2457.

graikund@scu.edu.au

Abstract

Transcripts, or logs, of discussion carried out in a computer-supported meeting are provided by many text-based meeting discussion tools. These verbatim logs are a valuable record of discussion and associated meeting characteristics, such as timestamps for remarks. Logs provide a useful reference for reviewing discussion in other meetings. Unfortunately, log files can be very detailed, and their structure is not the most readable and usable by humans.

Derivatives *are meeting documents derived directly from the original logs. A derivative's content is a product of an analysis of the log. The analysis is a "massaging" of original log data into a far more applicable form for use during a meeting process. Derivatives also provide features such as presentation of discussion according to corresponding agenda items and facilitating hypertextual navigation of the derivative.*

This paper presents a Web Electronic Meeting Document Manager *(WEMDM),* Logan. *A WEMDM is a World Wide Web tool performing document management functions such as agenda creation. Logan contains a document generation engine that dynamically and synchronously generates derivatives for use in a computer-supported meeting. Issues involved in derivative generation are covered, with* verbatim minutes *and* participant remarks *derivatives provided as examples. Results from experiments conducted with derivatives are also reported.*

Keywords Computer-supported meetings, document generation, document management, World Wide Web.

1 Introduction

Documents have always been a most important aspect of traditional, face-to-face meetings. Documents play essential roles such as providing relevant information for discussion or driving the meeting as in the case of the agenda. The structure and presentation of documents affects their usability in conduct of a meeting which may then influence the output of a group. The importance of documents remains true in the case of computer-supported meetings and so these documents issues are still of importance for study.

Although work in Electronic Meeting Systems and Group Decision Support Systems has been plentiful in recent years and resulted in plentiful software, specialised document support for meetings has not grown as significantly. Of the software for document support that has been produced, some software has dealt with issues associated with the formal documents of agendas and minutes. Other tools facilitate authoring of various specific documents, for instance, software specifications (Kaiya and Saeki [1]) or proposals and joint papers (Maly *et al* [3]). Additionally, general documents are authored usually with document conferencing/collaboration tools (e.g., MultiMETH by Lubich and Plattner [2]) where document content differs according to the meeting type and situation at hand.

However, the discussion taking place amongst participants during a meeting also contains useful and relevant content. Such discussion is the basis for generating decisions that will affect the group and possibly the organisation(s) in which the participants work. Discussion therefore leads to actions carried out by participants after the meeting. Many important details, even those appearing to be minor, are exchanged by participants and are pertinent for use outside of the meeting, especially in subsequent meetings. For instance, contact details (such as an email address) of an authority discussed during the meeting who a participant needs to contact for purposes relevant to the group.

Unfortunately, research into analysis of verbatim meeting discussion from which documents, containing exact results from rearrangement or filtration of discussion, are generated synchronously for use in meeting processes, has been lacking. The research as presented in this paper, however, puts forth a tool performing such log analysis and document generation along with results from experiments conducted with the tool. These documents can be used during discussion to assist it by providing relevant details found in previous discussion. For instance, a document containing rearrangement of and effective access to a participant's remarks made until some point in a meeting will assist in understanding the participant's line of argument. Alternatively, a document may be applied to other meeting processes such as minutes creation (where minutes pro-

Proceedings of the Third Australian Document Computing Symposium, Sydney, Australia, August 21, 1998.

vide outcomes and decisions of discussion). A document providing convenient access to different parts of discussion, which are readable and organised according to agenda items, assists in determining minute content.

This research has developed a *Web Electronic Meeting Document Manager* (WEMDM) called *Logan*. Logan provides a range of meeting document functionality such as agenda and minutes creation and usage (discussed in Raikundalia and Rees [6]). Logan is also a *derivative generator* that analyses logs to produce a range of derived documents (*derivatives*). Log files such as those as used in this research are detailed but are a high-value record of discussion and other meeting characteristics. Further processing of log files to generate derivatives provides documents with greater usability during meeting processes.

Derivatives contain rearranged or filtered ("massaged") log content. Derivatives are also marked up to contain appropriate hyperlinks supporting navigation within the derivative and externally to another derivative. Different parts of a derivative are accessed easily via these links. Logan is a Web tool and therefore uses Web-based linking, user interface elements and mechanisms such as popup menus and text fields. Logan is implemented using Common Gateway Interface (CGI) programming. Each derivative is generated by a corresponding CGI script.

Meetings supported by Logan are of a *five phase structure*. This structure is an extension of the well-known three phase structure containing pre-meeting, in-meeting and post-meeting phases. The five phase model divides the in-meeting phase into *startup*, *discussion* and *windup* phases. Therefore, Logan both analyses logs and lays out derivative content according to these five phases—functionality that is not supported in any other system.

2 Related Work

The great majority of the work in analysis of transcripts or logs is conducted manually, rather than for automating generation of an immediate and more useful form of information for a group. In many cases, there is interest in an investigation using transcripts. From the investigation, the researchers gain understanding about a problem and apply what has been learnt.

An example of investigation is in conferencing (Saunders and Heyl [9]). The authors present the application of conferencing to education, between a teacher and students. The study determined the appropriateness of computer conferencing in conveying complicated subject matter.

Major student concerns with computer conferencing found from transcripts concerned aspects like:

- difficulty in following course discussions
- segmentation of the program into two-week blocks
- anxiety in using the computer

As a result of manual analysis, recommendations were made for future conferencing activity; for instance, the need to solicit background information from students concerning prior computer experience.

Work has been done also in automatic generation of documents, such as hypertext documents. Masiero *et al* [4] present an instance of such work—a method for automatic creation and update of hypertext databases. The method determines the structural components of an office document and the processes for creating the components. The authors describe an implementation of the method for creation, storage and retrieval of documents in formal, face-to-face meetings. However, no analyses of meeting transcripts of the type carried out by Logan are prescribed or tested, and such support would require an extension of the implementation. Other related work in automatic generation of documents suffer similarly.

Much work has been done in forms of memory used by a group or within an organisation—*meeting*, *team* and *organisational memory* (Morrison [5] and Sandoe *et al* [8]). These forms of memory can provide a useful and well-organised store of information for a group or organisation. However, such work stops short of analysing meeting discussion, if contained by these forms of memory, to synchronously produce newer forms of information specifically for enhancing discussion. No such generation of documents as produced by Logan (shown later) is reported by research in memory.

3 Yarn Electronic Meeting Logs

The discussion logs used in the research are *Yarn* telemeeting logs. The Yarn telemeeting system (Rees *et al* [7]) is conversationally-oriented where remarks typed by a participant are displayed line-by-line. The ordering of characters when typed is retained when the characters are shown in the tool. This means that discussion is recorded word-for-word. Remarks are viewed by all participants in a WYSIWIS manner, that is, participants see exactly the same view of remarks.

A portion of a Yarn log is displayed at the left-hand side of Figure 1. The log represents discussion in a given *channel*. Timestamps in hour and minute format are displayed before the beginning of remarks. Remarks are contained within a pair of dashed lines with the participant's name inside the first of the pair. This dashed line also contains a remark number, preceded by a hash ("#"), so that a continuous count of remarks made during the meeting is available. For purposes of space and readability, other characteristics of the log are unable to be shown. These include timestamps indicating times of various events beginning each line of the log in hour, minute and second format, and lines containing "LOG:" stamps reflecting events unrelated to remarks, such as the entry or exit of participants or voting details.

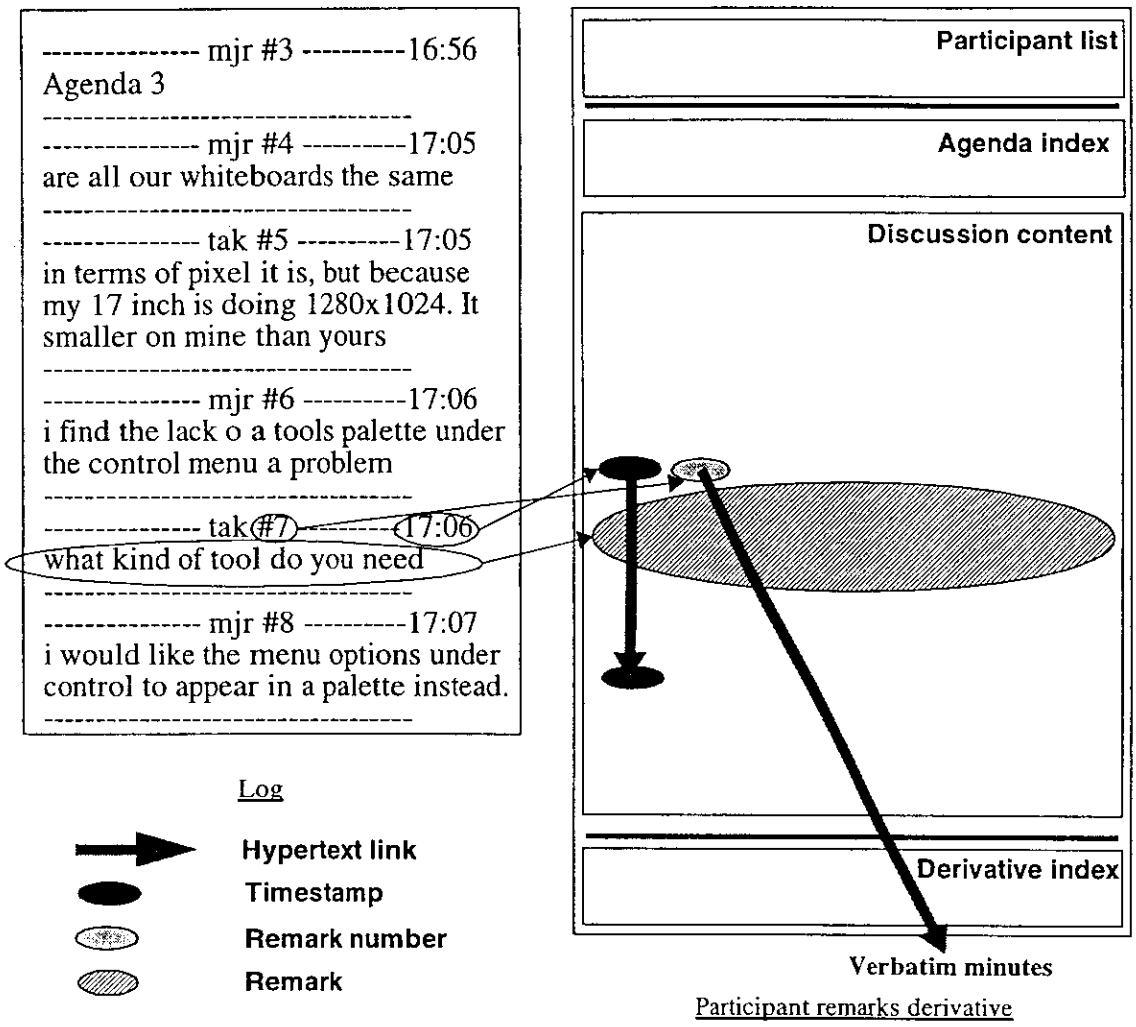


Figure 1: Example log content analysis for derivative

4 Derivative Generation

Derived document generation is shown in Figure 2. The generator consists of a set of CGI scripts. A particular script is dedicated to producing one of the many derivatives shown on the right-hand side of Figure 2. Verbatim minutes are a tidied-up version of the log retaining all discussion (and therefore, the entire context of discussion) that occurred during the meeting. This derivative functions as the "home page" for all other derivatives. That is, it is always possible to return to the original discussion (in the verbatim minutes) from a link in the derivative. This path to the original discussion is achieved through a variety of links from other derivatives back to the verbatim minutes.

The relevant script processes the file and creates the output dynamically ("on-the-fly"). That is, the derivative content is newly-created at the point of generation. Usually the case is that new content is added to the content of the last version of the derivative generated. At any point when a derivative is generated, the derivative content is dependent upon log content at that point. The output generated is directed to and displayed in the Web browser.

This Web CGI mechanism eliminates the need for creation and persistent storage of files containing derivatives. Thus, *lazy generation* of derivatives is employed. This mechanism is akin to a demand-driven approach to provision of an artifact. That is, when an artifact needs to be used, the operation to create the artifact is executed at the time of need. The script must be re-executed every time the derivative is to be displayed, except if the navigation buttons and menus on the browser, such as the "Back" and "Forward" buttons, are used when the derivative has been cached.

Returning to Figure 1, elements of the log which are useful for analysis are shown. The Figure shows analysis based on the elements of timestamps, remark

number and remarks. The diagram gives a general idea about analysis and is not a complete coverage of specific, minute examples of element use. The important aspects shown are the use of elements and the types of linking in a derivative. This particular analysis is of participant remarks made chronologically in a meeting, that interweave naturally in the case in normal conversation and in the log. Yet this *Participant remarks* derivative rearranges this information according to the participants who made each remark.

On the left side of the Figure is an example log and on the right side is the created derivative. The derivative is segmented into four components. For details in the *Agenda item index*, access to the "Agenda" identifier and item number (here, "Agenda 3") and number is required. The discussion of the meeting, found in the *Discussion content* component, will be composed largely of usernames, timestamps and remarks. Finally, there is an index for access to other derivatives of the same meeting, the *Derivative index*, but this index requires nothing from the log to be generated.

5 Derivative Types

The types of derivatives generated by Logan are:

1. verbatim minutes—tidied and marked up verbatim discussion
2. participant remarks—presentation of remarks by participants
3. time range—presentation of remarks according to a time frame specified by the user
4. remark range—presentation of remarks according to a range of remarks specified by the user
5. keyword—occurrences of a keyword used in discussion
6. addresses—email addresses and URLs used in discussion marked up to assist in authoring email or navigating the Web

Some derivatives require parameters to determine their content. An example of a parameter is a keyword required for a keyword derivative, such as "project". The essential content of the derivative will be all remarks containing "project". For reasons of space, two types of derivatives—verbatim minutes and participant remarks—will only be covered here.

5.1 Verbatim minutes

Verbatim minutes preserve remarks and mark up meeting discussion with:

1. an Agenda index composed of agenda item names and numbers, and startup and windup phase entries,
2. headings for agenda items and startup and windup phases (in the Discussion content),
3. linking from the Agenda index to the headings in 2
4. re-organisation and linking of remark information.

An agenda item consists of both an item number and an item name, e.g. "7 Other Business". Because the verbatim minutes are a home page for all derivatives, agenda item, startup and windup phase headings are not linked whereas they are linked in other derivatives back to the verbatim minutes.

Figure 3 displays the very top portion of an example verbatim minutes derivative. The top section of the derivative consists of the Agenda index containing entries for startup ("Startup") and windup ("Windup") phases and agenda items. An item name links to the corresponding discussion within the derivative, for example, the "Startup" link takes the participant to remarks made during the startup phase (as shown at the bottom of the Figure). If there has been no discussion of an item, then

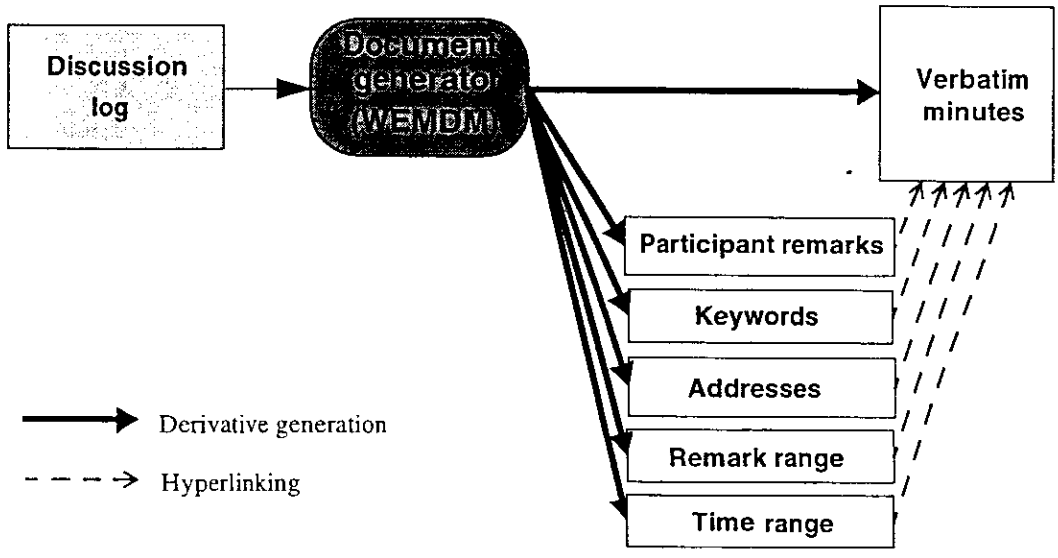


Figure 2: Derivative generation process

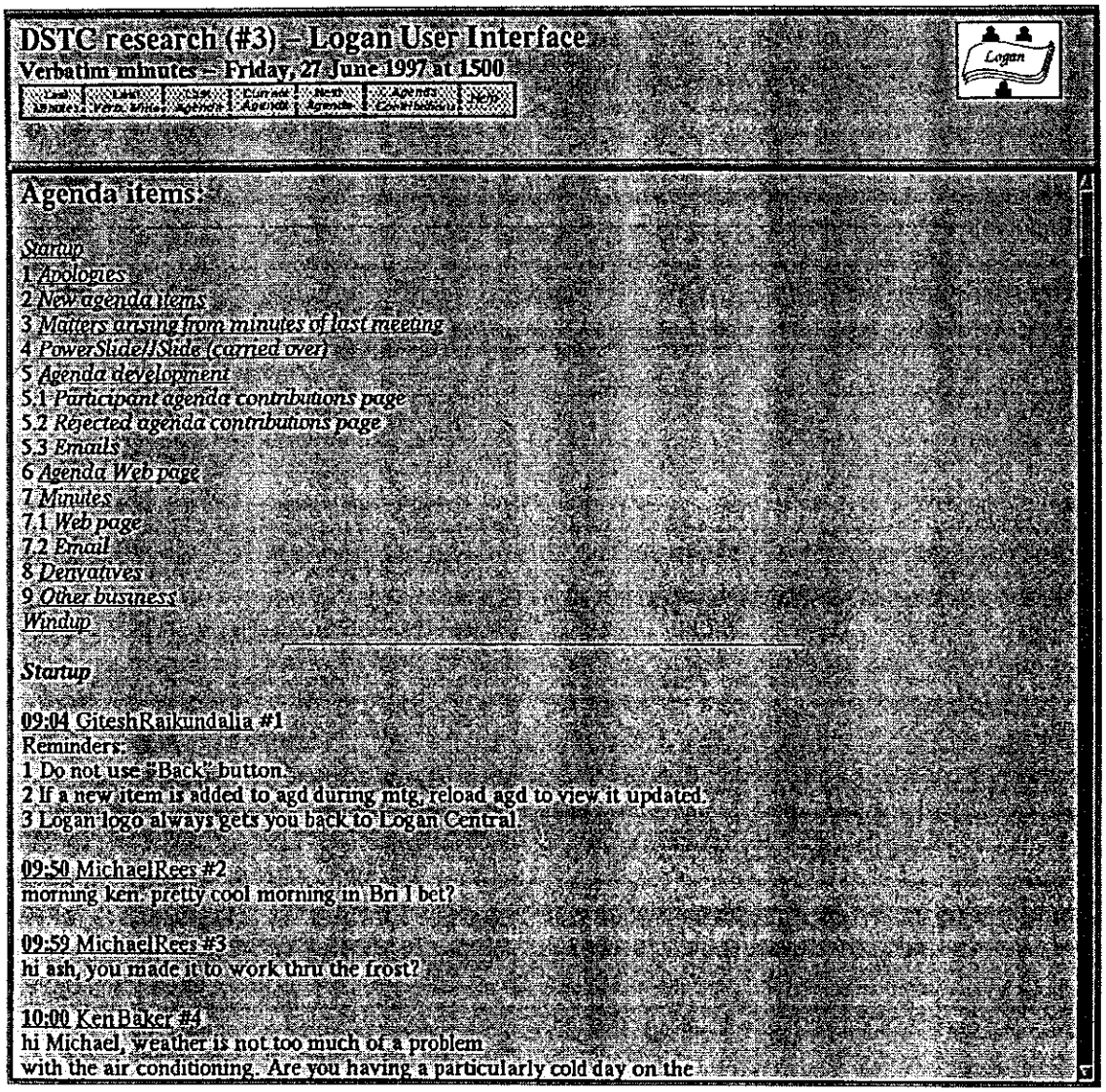


Figure 3: Verbatim minutes page—top section

the name is printed as normal text, that is, the name is not a link. An example of an item not discussed is item 5.1, "Participant agenda contributions page".

The Discussion content component is composed of remarks made by participants during discussion. A remark is presented with the following attributes on the same line preceding the remark: a timestamp (optional), username of the participant and remark number.

Each timestamp begins a new set of remarks that occur at the time represented by the timestamp. Therefore, all remarks occurring at 9:04 will be collected under the timestamp "09:04". The timestamp is a link to the next timestamp in the derivative. Thus, in Figure 3, the timestamp "09:04" will be linked to "09:50", "09:50" will be linked to "09:59", and so forth. Clicking on the last timestamp in the derivative will take the user back to the remark of the first timestamp in the derivative.

The username is a link to the next remark by the same participant, regardless of the time the remark was made. In the example, clicking on "MichaelRees" at remark #2 would take the user to remark #3. Like the case of timestamps, the last remark made by a participant will take the user back to the first remark in the derivative made by the participant.

Finally, the bottom of the derivative (not shown) consists of a form allowing the user to select any other type of derivative derivable from the log of the same meeting. The form presents a popup menu for selecting the type of derivative and a text field for supplying a parameter (for specifying a parameter-based derivative).

The verbatim minutes provide a view of discussion "as-is", that is, word-for-word. Ordering of remarks is retained exactly as found in discussion. If access to the meeting in its entirety is needed, then this derivative is appropriate. For this reason, verbatim minutes have been exploited in the *minutes creation* tool of Logan. An agenda always has a link to the verbatim minutes of the last meeting. Thus, the verbatim minutes are available from the agenda to assist with review of the last meeting. The minutes are the primary form of meeting

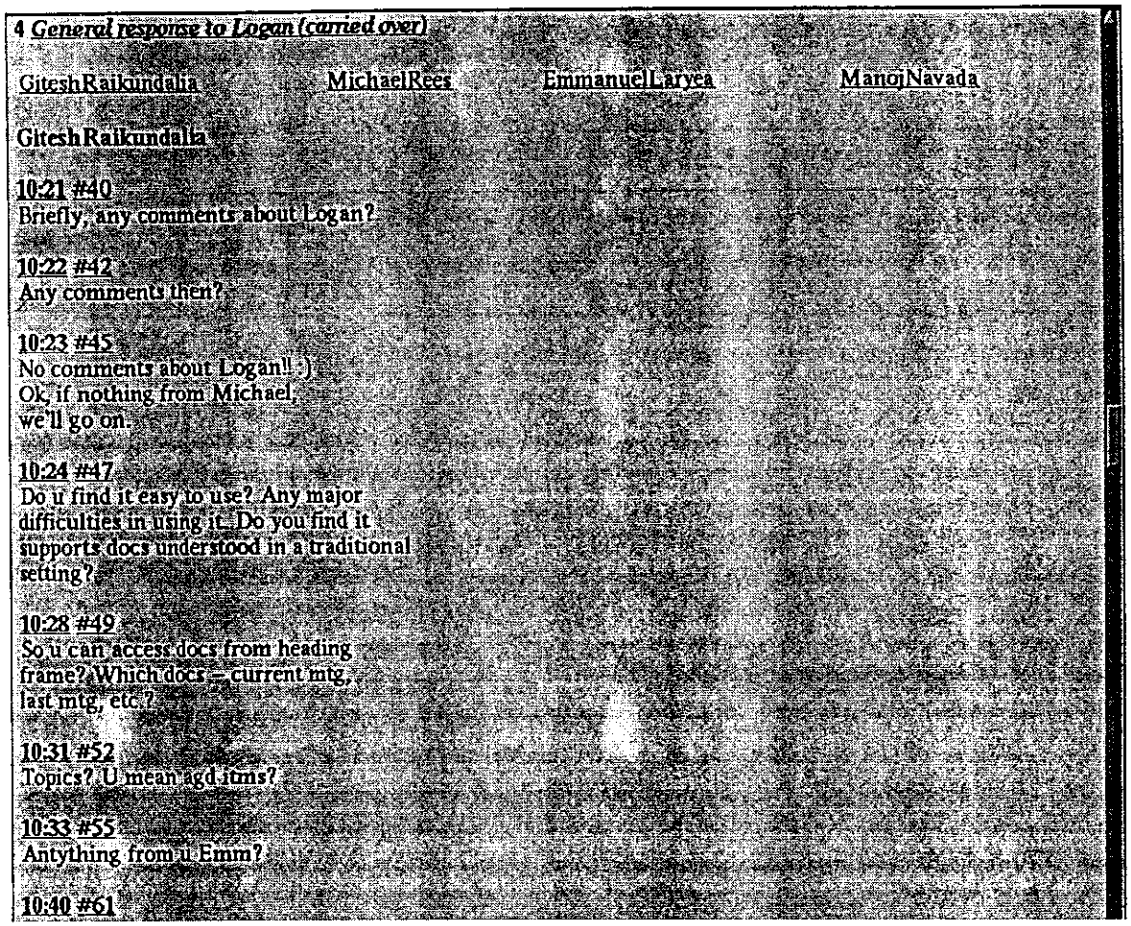


Figure 4: Participant remarks derivative—inner cross-section

revision in summary, but verbatim minutes allow full viewing of related discussion. Immediate navigability to agenda item discussion is provided.

5.2 Participant remarks

The purpose of this derivative is to group and present all remarks according to the participants who made them. The document content is effectively a grouping of all remarks *by participant*. This grouping is analogous to the database concept of information queried from a database being grouped according to an attribute, such as by person. Like other derivatives, all information is grouped by agenda item in chronological order. Within each grouping by participant, again remarks are presented in the order they were made.

There are situations for which the derivative is especially beneficial. An example is where a participant is providing instructions to one or more other participants to which these other(s) respond. Obviously it is more appropriate to provide the capability of focusing only on the instructions being given. Other work such as that of Saunders and Heyl [9] found that a benefit of computer conferencing for teacher-student interaction was that conferencing was "most appropriate for giving and receiving directions" (p 36). Formal meetings may contain similar instructional discussion, as supported by the experience with Logan.

Figure 4 provides a snapshot of an inner section of the Logan participant remarks derivative. A horizontal hypertext index of usernames that link to the set of remarks made by participants during item discussion is shown. For instance, clicking on "EmmanuelLaryea" will take the user immediately to Emmanuel Laryea's remarks below in the derivative (not shown). The remark number is a link to the same remark in the verbatim minutes. If the participant wishes to view the same remark in its original context, this remark number link allows access to the original context in one step. No participant name linking exists, as in the case of verbatim minutes, as all remarks by a participant are collected under the one heading.

6 Experimental Results

Two series of experimental meetings were carried out with Logan with participants. The first series was *exp1*, consisting of five meetings of four participants, and the second was *exp2* consisting of four meetings with six participants. Participants were distributed and took part

from Unix, PC and Macintosh platforms using popular Web browsers. One purpose of meetings was to carry out iterative design of documents. The following are major results gained from participant feedback to questionnaires, where interviews using the questionnaires were carried out for exp2.

In exp1, the linking in the derivatives was found by one participant as "very good". Only one participant stated that the linking on timestamps was found to be a "little bit confusing, even though it's very useful". No further improvement to this linking was discovered in retaining this facility. This participant found derivatives like verbatim minutes "cluttered". A suggestion was given in a later meeting to use horizontal space better in the derivatives by joining every two consecutive lines. This reduced the size of a derivative by half and improved readability. Another response was that the derivatives were a "good facility" that gave "ease of access [to discussion information]".

Feedback from exp2 participants revealed literally no complaints about the derivatives. This was exhibited in no responses to questions about derivatives or responses like "fine". One response was very positive to certain features of the derivatives. The response indicated that the agenda index was appropriate for accessing discussion on items. It was indicated that the format of the derivative—agenda index, participants' discussion and the index to derivatives—was natural. The feature where an agenda index was left as plain text where no corresponding discussion had occurred was also found useful. The method of linking for timestamps and usernames was approved. It was suggested that the reverse linking of the timestamps, that is, a timestamp navigating the user to the remarks associated with a previous timestamp, could also be helpful as an extension to the tool.

7 Conclusion

Derivatives have been shown to be a useful and improved presentation of log content. Log content can be presented in other more useful ways or filtered in relevant ways. Lazy generation of derivatives means that no storage space is required for derivatives, which is particularly helpful in long meetings involving a large quantity of remarks.

Iterative design and experimentation with end users in meetings lead to gradual improved design of derivatives. A very favourable outcome of experimentation was that by the end of the second experiment participants had no major complaints with derivative content or usage, nor were able to suggest further types of derivatives of use in meetings.

8 Acknowledgements

The work reported in this paper has been funded in part by the Cooperative Research Centre Program through the Department of Industry, Science and Tourism, Australia.

References

[1] H. Kaiya and M. Saeki. A groupware for face-to-face meetings to develop software specifications. In *InfoScience '93*, Korea Information Science Society, pages 691-698, October 1993.

[2] H. P. Lubich and B. Plattner. The MultiMETH conferencing and joint editing system. *Collaborative Computing*, Volume 1, Issue 2, pages 147-162, 1995.

[3] K. J. Maly, H. Abdel-Wahab, R. Mukkamala, A. Gupta, A. Prabhu, H. Syed and C. S. Vemuru. Mosaic + XTV = CoReview. In *Third International World Wide Web Conference*, Darmstadt, Germany, April 1995.

[4] P. Masiero, C. de Oliveira, F. Germano and G. Pierri. Authoring and searching in dynamically growing hypertext databases, *Hypermedia*, Volume 6, Number 2, pages 124-148, 1994.

[5] J. Morrison. Team memory: information management for business teams. In *Twenty-Sixth Annual Hawaii International Conference on System Sciences*, pages 122-131, Hawaii, January 1993.

[6] G. K. Raikundalia and M. J. Rees. Enhancing collaboration in formal, synchronous electronic meetings with LoganWeb. In *1996 Australian National Symposium on Computer-Supported Cooperative Work*, pages 6-13, University of Queensland, August 1996.

[7] M. J. Rees, G. P. Smith, R. Iannella, A. Lee and T. K. Woo. Yarn: text-based electronic meeting tools in a distributed environment, In *Computational Support for Distributed Collaborative Design*, pages 3-20, University of Sydney, September 1993.

[8] K. Sandoe, L. Olfman and M. Mandviwalla. Meeting in time: recording the workgroup conversation. In *Twelfth International Conference on Information Systems*, pages 261-271, New York, December 1991.

[9] C. S. Saunders and J. E. Heyl. Evaluating educational computer conferencing. *Journal of Systems Management*, Volume 39, Number 4, pages 33-37, April 1988.

ISBN 1 86487 001 X