Searching For Meaning With The Help Of A PADRE

David Hawking and Paul Thistlewaite
Co-operative Research Centre For Advanced Computational Systems
Department Of Computer Science
Australian National University
{dave,pbt}@cs.anu.edu.au

December 1, 1994

Abstract

Full-text scanning offers significant advantages over other methods of document retrieval but is normally too slow for use on large collections. The Fujitsu AP1000 parallel distributed-memory machine has been used to reduce the time penalty for full-text scanning to acceptable interactive levels. The query language for the retrieval software (called PADRE) is described herein and differences between PADRE and traditional systems are highlighted. The advantages of the full-text scanning in broader retrieval contexts are outlined. TREC precision-recall results are discussed and timings are reported.

1 Introduction

The computational cost of full-text scanning as a method of document retrieval is usually regarded as prohibitive for large document collections. The availability of a large-memory parallel machine challenges this wisdom by permitting a full scan of a TIPSTER-sized collection to be carried out in of the order of one second.

Full-text scanning offers the following advantages over other methods:

- It preserves all the information in the document collection.
- It allows queries to be constructed from completely general operations over the text.
- It permits rapid response to changes in the document collection, such as the addition or deletion of documents.

PADRE is a document retrieval system designed and implemented by the author for the Fujitsu AP1000 parallel distributed memory supercomputer. The architecture of this machine is described in [1, 2, 3]. Fuller details of the design and implementation of PADRE are to be found in [9]. PADRE's capabilities are constructed on top of a small set of basic pattern matching operations.

2 Capabilities Of The PADRE Query Language

The PADRE query language is more fully described in the User Manual [7]. Below is a brief summary of its capabilities. Note that the query language is not very user friendly. It is intended that PADRE queries will be generated by a more friendly, perhaps graphical interface. Indeed a graphical front-end for PADRE does exist (see Hawking and Bailey [6]) but it is only capable of accessing a subset of PADRE functionality.

2.1 Primary Terms

The basic pattern matching operations implemented by PADRE are as follows:

- Literal strings (using Boyer-Moore-Gosper (BMG) string matching);
- Numeric ranges (matches strings of digits whose numeric value lies in a nominated range, eg. 1776..1859); and
- Regular expressions (using GNU regular expression code)

2.1.1 Word-Start Mode

By default, matches for literal strings are constrained to start at the beginning of a word. For example, searching for `"ice"` will find only words beginning with that string. This behaviour can be changed by changing the value of the `wsmode` variable to `any` in which case matching is independent of word starts. In this case, `ice` would match `nice` and `spices` as well as `iced`.

2.1.2 Case Sensitivity

The `casesensitive` variable allows the searcher to specify whether searches for literal strings are to be considered case sensitive or otherwise.

2.1.3 Spaces In Patterns

A space in a PADRE literal term matches any single non-alphameric character.

This allows the searcher to locate phrases, such as `"wool exports"`, and word suffixes, such as `"ize "` which with `wsmode=any` will match any word ending in `"ize"` even if the character following is a punctuation mark.

Introducing a meta-character to match an arbitrary sequence of non-alphamerics would be a significant improvement but this has not yet been implemented in the BMG context. A regular expression can achieve this end but the computational cost is much greater. Alternatively, the searcher can make multiple BMG searches for the pattern with one space, two spaces, ..., etc. and then combine the results.

2.2 Regular Expressions

PADRE's `regexp` command allows the searcher to locate all occurrences of strings matching a regular expression using the Unix *egrep* syntax. PADRE incorporates the Free Software Foundation GNU regular expression code almost unmodified. The following example matches the most common pattern of vehicle registration (license) plate from the Australian Capital Territory:

```
R1: regexp "\<Y[A-Z][A-Z][0-9][0-9][0-9]\>"
```

2.3 How PADRE Represents The Results Of A Search

The result of a PADRE search takes the form of an ordered array of pointers to the first character of each match in the text collection. Such an array is called a *match set*. Match sets are used directly by the commands which display lexicographic context and are also used to calculate a component of the cumulative relevance measure for a document. Note that the match set is distributed across all processors in the parallel machine.

2.4 Compound Terms

Compound terms modify the behaviour of primary terms or combine the match sets resulting from searching for other terms, either primary or compound. There are three types of compound term:

- Component search (eg. *pattern* **within component** *component name*);
- Proximity search (eg. *pattern1* **near** *pattern2*); and
- Set operations, which allow pairs of match sets to be combined using difference, union and intersection operations. Set operators allow named as well as immediate operands.

2.4.1 Component Searches

PADRE permits the searcher to define components such as title, abstract, definition, headword etc., by specifying start and end markers for each component. For example the title of a document may be enclosed by `<title>` and `</title>` markers.

Subsequent searches for primary terms may be constrained to lie within particular components. Currently, only literals may be used in component searches but there will be no particular difficulty in generalising this.

2.4.2 Proximity Searches

Proximity is currently expressed as a fixed number of characters; it would be useful, and may also be feasible, to express it in terms of paragraphs, sentences or words.

The proximity operators `near` n and `fby` n operate on the n most recently computed match sets and produce a result set consisting of all the members of the first match set which satisfy the appropriate *near* (or *followed by*) relationship with members of the other sets. The following group of commands creates a result match set indicating all the occurrences of "Clinton" which are followed within 200 characters by one or more occurrences of "Yeltsin".

```
>> {proximity 200}
>> "Clinton "
>> "Yeltsin "
>> near 2
```

Note that proximity operators can be applied regardless of how the match set operands were computed.

2.4.3 Set Operations

Union, intersection and difference operators may be applied to named or immediate operands. They produce a result match set by applying the appropriate operator to two match sets. The union operator is useful for matching a series of alternatives. The following command makes a set of pointers to all occurences of each of the strings.

```
>> "car " + "vehicle " + "sedan "
```

Difference can be used to exclude specific instances of a general pattern. For example, the following command finds all words starting with `comput` except those starting with `computer`.

```
>> "comput" - "computer"
```

2.5 Measuring Relevance

A document's estimated relevance is computed according to the following formula:

$$R_d = 10000 \times \sum_{t=1}^{k} (I_t \times r_{(t,d)}) \tag{1}$$

where:

R_d is the estimated relevance of document d,

$r_{(t,d)}$ is the relevance of a document d due to term t,

I_t is the manually assigned importance of term t, and

k is the number of terms.

$$r_{(t,d)} = \frac{f_{(t,d)}}{\sqrt{F_t \times l_d}} \tag{2}$$

where:

$f_{(t,d)}$ is the frequency of term t in document d,

F_t is the frequency of term t in the entire collection, and

l_d is the length of document d

The factor of 10,000 is introduced merely to scale the relevance measures into a more convenient range. It may be that a logarithmic function rather than a square root would be even more effective at preventing the bias against long documents and toward infrequent terms from being applied too severely.

2.6 Identifying and Displaying Relevant Documents

The commands `top` n and `retrieve` n respectively identify and retrieve the n documents with the highest accumulated relevance scores.

2.7 Displaying Lexicographic Context

The searcher may request the display of each match in the current match set with specified number of characters of pre and post context. The `pr` command displays all matches; the `sample` n command displays at most n of them.

2.8 Key Differences Between PADRE And Conventional Systems

1. Use of pattern matching as the basic operation.

2. Availability of manually assigned term importance weighting.

3. Ability to use negative or zero importance weights.

4. Different meaning of set operations.

5. Support of regular expressions.

3 TREC-3 Participation

PADRE was entered only in the Ad Hoc section of the TREC competition. Unfortunately, the generation of both Manual and Automatic queries was subject to significant constraints on human resources and time. A TREC-ready form of the PADRE query language was first implemented only a few weeks before the deadline for entries and insufficient time was available to benefit from the training topics.

3.1 Generation Of Manual Queries

Manual queries were generated by an unskilled researcher (the principal author). Proximity operators and manually assigned term importance weightings were used extensively to try to improve precision and word prefixes were used to match multiple variants of the same word. For example:

```
{weight 0}
"smoking" + "smokers" + "tobacco "
"ban " + "prohibit" + "disallow" + "forbid" + "outlaw" + "bann"
{weight 1000}
near 2
```

Negative term importance weightings were used to implement exclusions specified in the topics.

```
{weight 1}
"murder" + "homicide" + "kill" + "assassinate" + "strangle" + "manslaughter"
"motive" + "reason " + "because "
{weight 1000}
near 2
{weight 0}
"popular" + "author " + "fiction" + "reviewer" + "book " + "whodunnit"
{weight -900}
near 2
```

Manual thesaurus expansion was used to improve recall. Synonyms and specific instances of general terms were generated using general knowledge, dictionaries, thesauri, subject-specific material and a friend who knew the names of American restaurant chains. Terms were combined using the PADRE union (+) operator.

3.2 Generation Of Automatic Queries

The core of the automatic query generation software is an algorithm for generating an ordered list of key words and phrases from a document. The topic specification provided by TREC is processed by this algorithm, and the output is formatted to comply with the PADRE query language.

The algorithm is based on familiar statistical techniques using word and phrase frequency information, word stemming, and phrase subsumption. Currently, no thesaurus or semantic-net information is used, and the algorithm could clearly be strengthened by its incorporation.

The relative weights for the PADRE query are calculated by normalising the rank values generated by the keywording software. Equivalent words and phrases (e.g., "Australian live sheep shipments", and "shipping live sheep from Australia") are detected by the algorithm, and the PADRE "+" operator is used to represent their synonymy. Phrases are not represented as strictly ordered sequences of words; rather the PADRE proximity and near operators are used to permit greater flexibility in word separation and order.

4 PADRE Performance

This year's results are relatively poor, though having achieved best results on three of the topic-specific measures is quite encouraging. We believe that the major area of potential improvement for next time lies in the process of query generation rather than in the basic design of PADRE.

The lack of a user-friendly query interface, coupled with lack of sleep on the part of the person generating the manual queries, caused a significant reduction in performance in the Manual category. No fewer than fourteen of the 50 Manual queries were later discovered to include obvious errors. Errors included neglecting to reset weight to zero before terms of no significance by themselves (12 times), leaving out a proximity operator (three times), and mistyping "santion". Correcting these errors (before looking at the documents retrieved) caused dramatic performance improvements on three topics, changing performance levels from worst or near worst to above median.

More generally, a lack of experience with the document collection and with generation of retrieval queries, shows up as rather poorly formulated queries.

4.1 Case Study - Topic 155, Manual

Right Wing Christian Fundamentalism - 42 relevant documents.

This was originally submitted with a missing "weight 0" meaning that any document mentioning any of `"constitution"`, `"Constitution"`, `"civil liberties"`, `"Civil Liberties"`, `"US"`, `"U.S."`, `"United States"` or `"USA"` was regarded as highly relevant. Consequently, 1000 documents were retrieved of which none were relevant.[1] The topic as originally submitted appears below.

```
topic 155
{weight 0}
{proximity 100}
"right wing " + "fundamentalis"
"christian"
"grass roots " + "politic" + "religious agenda " + "elector"
{weight 500}
near 3
{casesensitive 1}
"constitution" + "Constitution" + "civil liberties" + "Civil Liberties"
+ "US" + "U.S." + "United States" +"USA"
{casesensitive 0}
{weight 1000}
near 2
top 1000
```

[1] Sincere apologies to the person who read the 200 documents erroneously retrieved due to this error!

After correction of the error, 27 documents were retrieved of which still only 5 were relevant. To investigate why, all documents officially judged relevant and the documents retrieved by the modified PADRE query were extracted and examined. It soon became clear that the query was defective in several ways:

- A number of words and phrases connoting the religious dimension were missing from the query partly due to the query author's lack of subject knowledge. Examples include `"Moral Majority"`, `"evangeli"`, `"christian right"`, `"prayer"`, `"preacher"`, `"ministry"`, `"Pat Robertson"`, `"Jerry Falwell"`.

- A number of words and phrases connoting the US political dimension were missing from the query partly due to the query author's lack of subject knowledge. Examples include `"GOP"`, `"Republican"`, `"Democrat"`, `"primaries"`, `"Super Tuesday"`, `"Campaign"`, `"presidential"`, `"nomination"`, `"party"`.

- During initial query design, the author could not see how to enforce the requirement that the documents must relate to the U.S. It was hoped that by assigning greater weight to documents whose references to religion and politics occurred also near references to the United States, the constitution or the second amendment, this would cause actually relevant documents to achieve the highest rank, while allowing documents rendered irrelevant because of nationality to cause false hits lower down the list. This strategy did not work, largely because documents written in the United States about the U.S. political process do not generally name the U.S. A better strategy would attempt to include documents which mentioned names of U.S. states, U.S. cities, U.S. polititicians, U.S. religious figures and U.S. political parties and to exclude documents which frequently referred to foreign places and personalities. In fact, most of the erroneously retrieved documents could be excluded or down-weighted by assigning negative weights to places and people associated with Lebanon.

- The proximity of 100 was too tight when looking for associations between terms connoting the religious dimension and terms indicating the U.S. political dimension.

We are optimistic that a refined query addressing these defects would achieve dramatically better performance.

4.2 Speed

PADRE runs for the TREC competition were run on a 512-processor configuration of the Fujitsu AP1000, a machine with 8 gigabytes of RAM. Competition runs used full-text scanning exclusively. Table 1 shows the average times taken to process TREC-3 topics in this way.

Though it was not used in the TREC competition, PADRE does have the ability to build a distributed inverted file index. Table 2 shows the relative speeds of the three alternative PADRE methods for locating literal strings. Table 3 shows the time taken to build an inverted file index over the TREC data.

A time of 40 seconds to process a complex TREC query does not seem excessive. However, considerable improvement is possible. Dramatic improvements could be made by building an inverted file and using it to search for literal terms. The applicability of the inverted file could readily be extended.

Even without using an inverted file, improvements are likely to be possible in the following areas.

Query Type	No. terms per topic	No. set/proximity operations per topic	Mean time (sec.) per topic
Manual	11–110 (mean=27.4)	9–105 (mean=24.7)	38.1
Automatic	5–57(mean=18.7)	1–29 (mean=9.4)	42.3

Table 1: Elapsed time required to process TREC-3 topics on a 512-processor Fujitsu AP1000 using full-text scanning. Times are averaged over all 50 topics. They do not include time taken to load the data into memory across the network.

Search method	Eapsed time (sec.)
Boyer-Moore-Gosper	1.12
GNU Regular Expression	6.60
Indexed	0.02

Table 2: Elapsed time required to find all occurrences of a literal term using three different methods. Each time is the average of the times taken to locate the six strings: "ape", "fish", "cannibal", "Australia", "reject", and "unconditional". The data set comprised all three TREC CD-ROMs after purging manual indexing terms and the machine was a 512-processor Fujitsu AP1000.

Data set	Elapsed time (sec.)
CD1 and CD2 combined	92.18
CD1, CD2 and CD3 combined	128.41

Table 3: Elapsed time taken to build an inverted file index over the TREC-3 data, using a 512-processor Fujitsu AP1000. Note that the inverted file actually consists of 512 separate partial indexes which do not need to be merged.

- Replacing the multiple literal scans and multiple union operations used to find alternates with a single-pass finite state automaton method.
- Replacing the very inefficient algorithm for returning documents in order of decreasing rank.
- Introducing a more effective automatic method of balancing load across cells.

PADRE's ability to handle dynamic document collections has been reported elsewhere [9]. Retrieval downtimes caused by additions to or removals from the document collection did not exceed 20 seconds for changes of 10 megabytes or more.

5 Desirable PADRE Extensions For Future TRECs

1. Tuning up the software which generates automatic queries and adding thesaurus and semantic net capabilities are likely to produce the greatest benefit.
2. For manual query generation, a well-designed user interface is sorely needed.
3. It would be very useful if PADRE incorporated pre-defined *class terms* which could be efficiently located in the text. For example, the class term `US-states` might match any of Alaska, Arizona, etc.

6 Beyond TREC-3

The architecture of PADRE is capable of supporting retrieval based on any function computable over text. In its present form PADRE includes capabilities which were not exercised in the TREC-3 competition.

Numeric ranges are useful in conjunction with proximity operators. They would also be useful in component searches. It is planned to remove the current implementation restriction which prevents the latter.

```
>> "Nixon "
>> 1968..1972
>> near 2
..
>> 1968..1972 within component publication_date
```

Thought will also be given to extending PADRE to handle more general date comparison functions. For example:

```
>> published before June 1985
```

Below are some illustrations of PADRE's regular expression capabilities. R2 matches numbers from 1900 to 1999. R3 could be used to find documents containing Gaithersburg area telephone numbers. It matches several different forms, while R4 matches NIST email addresses.

R5 and R6 illustrate the use of regular expressions to match alternative spellings or variations on the same word. R6–R8 could be used to find documents of particular types, based on content. R6 matches a common form of LaTeX command, R7 matches the familiar SGML pattern `<blah> .. </blah>` and R8 matches C language comments.

```
R2: regexp "\<19[0-9][0-9]\>"
R3: regexp "\<\+?1?\-?\(?301\)?[- ][0-9][0-9][0-9][- ]?[0-9][0-9][0-9][0-9]\>"
R4: regexp "\<[a-zA-Z_\-]+@(ncsl.nist.gov)|(NCSL.NIST.GOV)\>
R5: regexp "\<judge?ment\>"
R6: regexp "\<ha(th)|d|s|(ve)|ving\>"
R6: regexp "\\begin{.*}"
R7: regexp "<([a-zA-Z]+)>.*</\1>"
R8: regexp "/\*.*\*/"
```

PADRE regular expressions can also be used to find pallindromic words of up to 19 characters in length, however this requirement arises rather infrequently in practical document retrieval applications.

The PADRE architecture could feasibly be extended to support retrieval based on stylistic properties. For example, retrieving documents in order of decreasing stylistic similarity to the Gettysburg address. A future PADRE might also be able to be used to retrieve documents which quote or paraphrase texts of interest.

Acknowledgements

Fujitsu Laboratories provided access to AP1000 machines in the Fujitsu Parallel Computing Research Facility and assisted in various ways. Robin Stanton gave support and advice. Peter Bailey and Michael Hiron worked on elements of PADRE code. GNU regular expression code from the Free Software Foundation is incorporated in PADRE. I extend my sincere thanks to all these people and organisations.

Recent work on PADRE has been supported by the Co-operative Research Centre for Advanced Computational Systems (ACSys).

References

[1] Horie, T., Ishihata, H., Shimizu, T. and Ikesaka, M. 'AP1000 Architecture And Performance Of LU Decomposition,' in *Proc. 1991 Int'l Conf. On Parallel Processing*, pp. 634-635, August 1991.

[2] Ishihata, H., Horie, T., Inano, S., Shimizu, T. and Kato, S. 'CAP-II Architecture,' in *Proceedings of the First Fujitsu-ANU CAP Workshop*, paper 1, Kawasaki, Japan, (Nov 1990).

[3] Horie, T. Ikesaka, M. and Ishihata, H. 'Interconnection Network For Multiprocessors', in *Proceedings of the First Fujitsu-ANU CAP Workshop*, paper 2, Kawasaki, Japan, (Nov 1990).

[4] Hawking, D.A. "High Speed Search of Large Text Bases On the Fujitsu Cellular Array Processor," *Proceedings of the Fourth Australian Supercomputing Conference* pp. 83-90. Gold Coast, Australia, (Dec 1991).

[5] Hawking, D.A. "PADDY's Progress (Further Experiments in Free-Text Retrieval on the AP1000)," in *Proceedings of the First Annual Users' Meeting of Fujitsu Parallel Computing Research Facilities* paper ANU-8, Kawasaki, Japan, (Nov 1992).

[6] Hawking, D.A. and Bailey, P. "Towards a Practical Information Retrieval System For The Fujitsu AP1000)," in *Proceedings of the Second Fujitsu Parallel Computing Workshop* paper P1-S, Kawasaki, Japan, (Nov 1993).

[7] Hawking, D.A. *PADRE User Manual* Department of Computer Science, Australian National University, Canberra, Australia, Oct 1994.

[8] Hawking, D.A. "PADRE — A Parallel Document Retrieval Engine," in *Proceedings of the Third Fujitsu Parallel Computing Workshop* paper P2-C, Kawasaki, Japan, (Nov 1994).

[9] Hawking, D.A. "The Design And Implementation Of A Parallel Document Retrieval Engine," *In Preparation*