THE NEAREST NEIGHBOUR PROBLEM IN INFORMATION RETRIEVAL

An Algorithm Using Upperbounds

A.F. Smeaton and C.J. van Rijsbergen

Department of Computer Science,
University College Dublin, Belfield, Dublin 4.

INTRODUCTION

The nearest neighbour problem can be expressed as follows: Given a set of N points in n-space and a distinguished point, q, find the m (m < N) points 'closest' to q, closeness being measured by some distance measure. In information retrieval we can apply the model to the situation where N is the number of documents, each document corresponds to a point in n-space, q corresponds to a query and m corresponds to a user specification of how many retrieved documents or nearest neighbours are desired. 'Closeness' is measured by any of a number of similarity measures (see later) and n-space corresponds to n concepts or index terms. For simplicity we consider the case where the user requires only one, the nearest, neighbour to a query, but later on we can generalize to more nearest neighbours.

The standard way of doing a nearest neighbour search is a sequential search of the entire collection, calculating a similarity measure for each document and selecting the best. This requires O(N) calculations which is unreasonable for large collections. Bentley and Friedman <1> gave an algorithm of order log(N) but this was unusable in the information retrieval case because it has a multiplicative constant of 1.6^n, where n is the dimension of the space. I.R. systems have typically thousands of tens of thousands of dimensions or index terms so $1.6^n \log(N)$ is also unreasonable.

A deterministic algorithm of Eastman and Weiss <2,3> is based on a binary tree search using priorities which searches a portion of the collection and calculates an upperbound value of a similarity measure for the rest of the collection and the algorithm stops when the upperbound for the remainder is less than the largest value of similarity measure encountered so far. This algorithm works in $O(\log N)^K$, K being a collection dependent constant, about 4 for document collections tested so far. This algorithm produces full search results without a full search, but for small collections only saved around 5-10%.
Weiss <4> produced a probabilistic method based on the above, which is substantially faster than a full search while providing nearly, but not quite, the same level of performance as a full search. In this algorithm, the user specifies a limit to the error tolerance, but in many applications of the nearest neighbour problem in I.R. (Maximum or Minimum Spanning Tree generation) a cost/quality tradeoff is not desired.

If an inverted file of the document collection is available, then one can search the entries of the appropriate records of the inverted file for possible nearest neighbours. The case may arise when a document may be compared to a query more than once, if it is indexed by more than one query term and so occurs in more than one 'appropriate record' of the inverted file.

This can be overcome by maintaining a set, say S, containing all document numbers compared to the query in the search so far. Document numbers as entries in records of the inverted file are tested to see if they are in S, and if they are then that document has been encountered previously as a possible nearest neighbour, and it can be ignored. Maintaining this set S is an overhead, and the decision of whether to include it or not depends on the properties of the document collection. For our algorithm we shall include this set and test for duplicate entries.

THE ALGORITHM

The algorithm maintains two sets, R and S. R contains all those documents which at any given time are candidates for the final set of nearest neighbours. Let us assume for now that this set is of size one, i.e. we are seeking the nearest neighbour. S contains all documents examined so far regardless of whether they are in R or not. Initially both are empty. The algorithm takes as input a test query of unweighted search terms of the form

$$q = (t_1, t_2, \dots t_k).$$

We arrange the terms of q in order of their increasing frequencies of occurrence within the entire document collection, i.e. in order of their increasing numbers of entries in the appropriate records of the inverted file. We then examine each of these terms and for each we find all the documents in the collection which contain that term. The nearest neighbours to q will be found among these documents. We can do this if we assume that if the number of co-occurrences of index terms between a document and a query is zero, then the similarity is zero, so we can eliminate all those documents not indexed by at least one of the search terms from our nearest neighbour search. This principle is used for Maximum Minimum Spanning Tree generation by van Rijsbergen et al <5>.

Each term can be thought of as a list of documents

$$t_i = (d_{i1}, d_{i2}, \dots d_{im(i)})$$

where $m(i)$ is the number of documents containing term i. We inspect these documents in that order. For each document we do the following: Check if the document is in S. If it is then we can ignore it as it has already been encountered. If not then we add it to S, and we calculate a similarity value between this document and the query using one of the formulae (see later). If this value is greater than the best value encountered so far, then this document becomes the candidate for the nearest neighbour and is added to R. After looking at every document containing a given term we can calculate the maximum possible similarity value among those documents not yet encountered, the upperbound, and if this upperbound is less than the current best value, we can terminate the algorithm.

We can do this as follows (see Porter <6>). Assume we have searched the entries in the records of the inverted file, for the first i-1 terms of the query. If any un-inspected document d_{ij} is better than the best nearest neighbour found so far, it cannot contain any of the index terms $t_1, t_2, \dots t_{i-1}$, otherwise it would have been encountered earlier, therefore it hasn't any more than k-i+1 terms in common with q, k being the size of q. Let L_r be the length of the shortest term list of all documents indexed by t_r

$$L_r = \min_{j>=1} | d_{rj} |$$

and l_i to be the smallest of the L_r of terms in q, from term i onwards,

$$l_i = \min_{r>=i} L_r = \min_{r>=i} | \min_{j>=1} | d_{rj} | |$$

then we get the following upperbounds for these similarity measures between document and query:

MEASURE	FORMULA	UPPER BOUND
SIMPLE	c	(k-i+1)
IVIE	c/ak	$(k-i+1)/(k*l_i)$
DICE	2c/(a+k)	$2(k-i+1)/(l_i+k)$
COSINE	c**2/ak	$(k-i+1)**2/(k*l_i)$
JACCARD	c/(a+k-c)	$(k-i+1)/(l_i+i-1)$
OVERLAP	c/min(a,k)	$(k-i+1)/\min(k,l_i)$

where c equals the number of terms in common (co-occurrences) and a and k are the sizes of the term lists of the document and query respectfully.

If the maximum possible number of co-occurrences for all documents not yet encountered in the search is greater than the shortest term list of documents indexed by term i and onwards of the query ($k-i+1 > l_i$), then we reset the value of l_i to the value of the maximum possible number of co-occurrences in order to get the highest theoretically possible value for all, as yet uninspected documents, as nearest neighbours. This upperbound is applicable to any similarity measure which combines number of terms in common and term lengths of document and query.

TEST COLLECTIONS

The algorithm has been applied to two test collections, UKCIS2 and NPL. The United Kingdom Chemical Information Service (UKCIS) collection is a large test collection of about 27,000 documents about chemistry <7>, and roughly split into two halves (UKCIS1 and UKCIS2) dealing with seperate aspects of the subject. The queries and document titles are automatically indexed using a stemming algorithm. In our experiments we use the UKCIS2 collection.

The national Physical Laboratory (NPL) test collection is another large document collection of 11429 documents about atmospherics and computer science and was prepared by Vaswani and Cameron <8>. For this collection, the document title and abstract are also automatically indexed by stemming. The relevant statistics for the two collections are given below.

	UKCIS2	NPL
number of documents	15748	11429
number of terms	8882	7491
average terms/doc.	6.4	19.9
average docs./term	11.3	30.4
number of test queries	182	93
average terms/query	7.8	7.14

From these figures it can be seen that the size of the collections are of the same order with regard to the numbers of documents and of terms. The first important difference to note between the collections is the different levels of indexing exhaustivity. The UKCIS2 collection has an average of 6.4 terms per document, while for NPL this figure is nearly 20. This means that the NPL documents are described in more detail, i.e. by more index terms, than the UKCIS2 documents.

A direct consequence of this difference which will effect the results of our nearest neighbour algorithm is that more documents are indexed by a given term in NPL than in UKCIS2, the figures being 30.4 to 11.3. This means that in the NPL collection more comparisons of document to query are necessary to find a nearest neighbour for an average query, than in UKCIS2, if one is searching the appropriate records of the inverted files. We shall come back to this point later.

The algorithm was run for the 182 and 93 test queries of the UKCIS2 and NPL collections respectively. These queries consist of an average of 7 index terms each. Most queries are composed of both high and low frequency index terms, i.e. terms occurring both a lot and rarely, among all documents in the collections. We shall see later on how important this property of the test queries is.

EXPERIMENTAL RESULTS AND ANALYSIS

These are the results obtained for the 2 collections. The figures show the actual numbers of similarity value calculations needed to obtain the nearest neighbour, using the Cosine, Ivie and Dice measures. The figures in parentheses for the NPL collection show the numbers needed to calculate the best five nearest neighbours according to the given measures and I/F means the inverted file.

	UKCIS2	NPL
I/F Search (inc. dups.)	1117	4078
I/F Search (exc.dups.)	1040	3156
Cosine + u/bounds (exc.dups.)	663	1876 (2251)
Dice + u/bounds (exc.dups.)	585	1591 (1900)
Ivie + u/bounds (exc.dups.)	615	1755 (1989)

There are some important points to make about these results. Firstly, the striking difference in the number of comparisons needed in all cases between the two collections. This 1:3 ratio, right down the table, is a direct consequence of the differing levels of indexing exhaustivity between the collections, also about 1:3. An average of 11.3 documents indexed by a given term in UKCIS2 as opposed to 30.4 in NPL means that more comparisons are needed for the average query term in NPL than in UKCIS2. Because the average number of terms/query for the two collections are almost the same, this ratio also applies to the complete queries. An interesting discussion of the influence of indexing exhaustivity on calculating similarity functions efficiently can be found in <9>.

The second point to note about the results is the effect of including the set S to test for duplicate entries in the inverted file records. The difference is quite noticible, especially for NPL where the reduction is almost 25% (4078 comparisons reduced to 3156). The reason for the slightly less emphatic improvement in UKCIS2 (1117 reduced to 1043) is once again attributed to the differing levels of indexing exhaustivity of the collections. Because there are less documents to be compared to the query, there is a smaller chance of duplicate entries occurring. Willett <10> gives details of how duplicate checking may be implemented.

The last, and most important, point we wish to make about the results are about the reductions the upperbound method yields over an inverted file search. This is approximately 40% for all cases. The differing figures for the three similarity measures we have chosen is a property of the definition of those measures, and these differences seem to be consistent for both collections. We decided to calculate the five nearest neighbours using the given similarity measures, for the NPL collection. This increased the number of comparisons needed by 15% to 20%.

CONCLUSIONS

From the previous section it can be concluded that our algorithm yielded quite an improvement over an inverted file search, while still maintaining full search performance. There are, however, certain overheads with our algorithm. As well as a document file and an inverted document file, we also need a file containing the values of L_r for all index terms in the collection. These shortest document term lists of all documents indexed by given terms are needed to calculate the upperbounds accuratly.

We mentioned earlier the fact that we checked for duplicates in inverted file record entries and consequently our upperbound results would be less of an improvement if we omitted this. Whether to include this set S or not would be a property of both the document collection and how the algorithm was implemented.

Because query terms are sorted in order of their increasing frequencies of occurrence within the entire collection, query terms are processed in this order. This means that the inverted file records of the query terms with the highest document frequency are not searched at all if an upperbound is reached. An analysis of the queries shows they are composed of medium and high frequency terms with most queries reaching an upperbound with about one or two query terms yet to process. It is an important fact that the terms left are the highest frequency terms, and this explains why the results have been so good.

ACKNOWLEDGEMENT

We wish to thank Dr Porter and Dr Harper for their help in formulating some of the ideas in this paper

REFERENCES

1. Bentley, J.L., Friedman, J.H., and Finkel,R.A. "An algorithm for finding best matches in logarithmic time", Stanford University Computer Science Report, No.STAN-75-482, (1975).

2. Eastman, C.M., "A tree based algorithm for nearest neighbour searching in document retrieval systems", Ph.D. dissertation, The University of North Carolina at Chapel Hill, (1977).

3. Eastman, C.M. and Weiss, S.F., "A tree algorithm for nearest neighbour searching in document retrieval systems", Proc. International Conference on Information Storage and Retrieval, SIGIR, Rochester, NY, (1978).

4. Weiss, S.F., "A probabilistic algorithm for nearest neighbour searching", Proceedings ACM-BCS Symposium on Research and Development in IR, Cambridge, England, (1980).

5. van Rijsbergen, C.J., Harper, D.J. and Porter, M.F., "The selection of good search terms", Information Processing and Management, 17, pp.77-91, (1981).

6. Porter, M.F., "The nearest neighbour problem in document space - an algorithm", Private Communication, (1980).

7. Barker, F.H., Veal, D.C. and Wyatt, B.K., Retrieval Experiments Based on Chemical Abstract Condensates", Research Report No.2, UKCIS, Nottingham, (1974).

8. Vaswani, P.K.T. and Cameron, J.B. "The National Physical Laboratory experiments in statistical word associations and their use in document indexing and retrieval". National Physical Laboratory, Teddington. (1970).

9. Harding, A.F. and Willett, P. "Indexing exhaustivity and the computation of similarity matrices". Journal of the American Society for Information Science, 31, pp.298-301, (1980).

10. Willett, P. "A fast procedure for the calculation of similarity coefficients in automatic classification", Information Processing and Management (in press).