View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by AIS Electronic Library (AISeL)

Association for Information Systems
AIS Electronic Library (AISeL)

ICIS 1988 Proceedings | International Conference on Information Systems (ICIS)

1988

NATURAL LANGUAGE DOCUMENTS: INDEXING AND RETRIEVAL IN AN INFORMATION SYSTEM

Bernd Teufel
Institut fur Informatik Eidgendssische Technische Hochschule

Follow this and additional works at: http://aisel.aisnet.org/icis1988

Recommended Citation
Teufel, Bernd, "NATURAL LANGUAGE DOCUMENTS: INDEXING AND RETRIEVAL IN AN INFORMATION SYSTEM" (1988). *ICIS 1988 Proceedings*. 24.
http://aisel.aisnet.org/icis1988/24

This material is brought to you by the International Conference on Information Systems (ICIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in ICIS 1988 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

NATURAL LANGUAGE DOCUMENTS: INDEXING AND RETRIEVAL IN AN INFORMATION SYSTEM

Bernd Teufel
Institut für Informatik
Eidgenössische Technische Hochschule

ABSTRACT

A steadily increasing number of natural language (NL) documents are handled in information systems. Most of these documents typically contain some formatted data, which we call strong database data, and additionally some unformatted data, i.e., free text. The task of a modern information system is to characterize such unformatted (text) data automatically and, in doing so, to support the user in storing and retrieving natural language documents. The retrieval of natural language documents is a fuzzy process because the user will formulate fuzzy queries unless he uses some strong search keys. Retrieval of natural language documents can be facilitated with natural language queries; that is, with searches based on natural language text comparisons.

1. INTRODUCTION

One of the problems of the eighties is the information explosion. We are especially aware of this problem in an office environment. In such an environment, most information is available in the form of natural language documents. The task of a modern information system is, first, to describe such natural language texts automatically and, second, to find some of these texts satisfying a specific user query. For the user it is impossible to remember all the stored documents. Therefore, an information system has to support the user in finding the documents he needs.

In our opinion, searching for natural language documents is easier with natural language queries. We all know the classical systems which allow the query formulation using terms from a controlled vocabulary (e.g., a thesaurus). In most of these systems, the user has the capability of qualifying the terms through logical combinations or term weighting. Because formulating complex logical expressions is not a trivial task for an untrained user, this extremely formal way of formulating questions is very inconvenient. We propose that natural language queries can be one solution to this problem. We base this claim on the assumption, that if somebody is searching for natural language documents on a computer, he already stores in this computer's memory some natural language documents describing some of the needed information. Therefore, users want to use one (or a combination) of these documents as a query. For example, a lawyer has a database of natural language documents describing court cases. When this lawyer protocols a new case in the form of a natural language document, he can search for similar cases in his database. In doing so, he uses the new natural language document as a query. Thus, a query may no longer consist of terms and logical operators but of text written in natural language. Consequently, natural language query processing is a must for modern information systems.

Finding natural language documents as an answer to fuzzy natural language queries is a process that begins with the indexing of the documents. One cannot separate the indexing process from the retrieval process. Indexing natural language documents means describing the contents of the documents, i.e., creating an abstraction that can be evaluated with the retrieval process. Here we have to distinguish between an exact description of the content, i.e., the extraction of the concepts in the document, and a description based on statistical analysis. The exact description is not done very well automatically -- it is still a domain of intellectual indexing. Several systems perform automatic indexing based on well known statistical and information theoretical methods (Jucquois-Delpierre 1987; Salton and McGill 1983). Salton states that the full scope of language understanding may not be needed in information retrieval (Salton and McGill 1983, p. 257), suggesting the concept of *homeosemy* as defined by Karlgren (1977). But most of these systems do not allow natural language queries; for example, queries which are documents themselves. A new indexing method, allowing natural language queries, has been found based on word fragments or n-grams.

Several tests have shown that n-gram indexing is a useful discrimination method for the retrieval of text documents (de Heer 1982; Mah and D'Amore 1983; Teufel and Schmidt 1988). The advantage of using n-grams of fixed length n is that the maximum number of possible n-grams is a priori known. This fixed indexing set for any natural language document is a major advantage over keywords. For example, if we choose $n = 3$ and an alphabet with 26 characters, we will have an indexing set whose size cannot exceed $26^3 = 17{,}576$ 3-grams (*trigrams*). Because n-grams are widely used to detect and correct spelling mistakes (see for example, Zamora, Pollock and Zamora 1981; Angell, Freund and Willett 1983; Mundt 1987), n-gram indexing provides an additional side-effect: writing mistakes or synonymous spellings (e.g., *color* and *colour*) will be corrected automatically.

For every natural language document a so-called information trace or syntactic trace can be generated (de Heer 1974). The information trace of a document is comparable to the trace somebody leaves when walking in the sand -- or, in other words, the information trace is the footprint of a natural language document. De Heer defines the information trace of a text t (π(t)) as the set of all overlapping trigrams. For example:

π(MISSISSIPPI) = {IPP, ISS, MIS, PPI, SIP, SIS, SSI}

This definition of an information trace, together with statistical information about the elements of the trace, is the basis for the n-gram indexing and retrieval method described in this paper. Additionally, we can generate a kind of document spectrum if we take the frequency of each trigram into consideration, as shown in Figures 1 and 2. The frequency distribution of the trigram set of a single document is shown in Figure 1, while Figure 2 shows the distribution of 4570 different trigrams generated by our test collection consisting of 2472 INSPEC documents (title, abstract, author, etc.). The figures must be interpreted in the following way: For each possible trigram (a_1, a_2, a_3), a_1 is plotted along the y-axis and a_2 along the x-axis. Each (x, y) coordinate intersection contains on the x-axis the 26 points corresponding to a_3 and on the y-axis the relative frequency of the trigram.

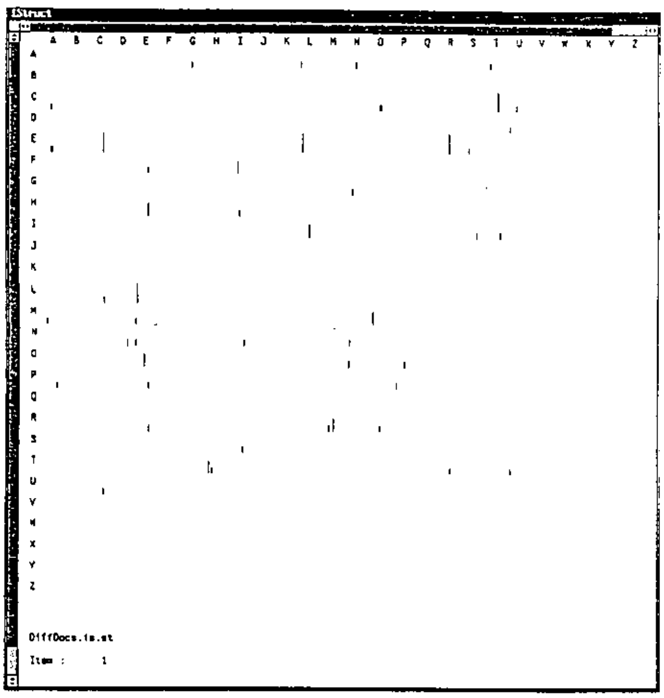

Figure 1. Trigram Distribution of a Document

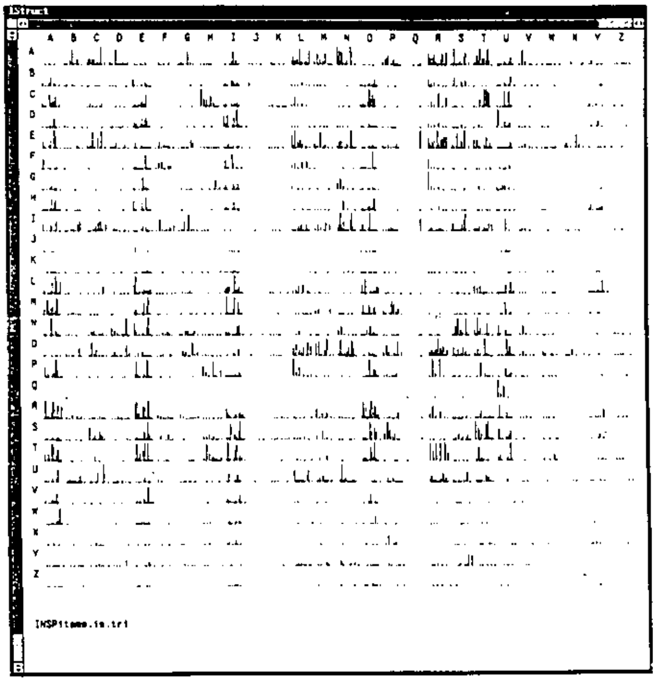

Figure 2. Trigram Distribution of the INSPEC Test Collection

As a side note, we observe that the question for the reconstruction of a text out of an information trace is very interesting. Considering a single word and its trace, there exist a one-to-one correspondence between the word and its trace, if the trace can be mapped onto a cycles-free graph. Further investigations of this field can be found in (Chudacek and Benshop 1981).

2. DOCUMENTS

As mentioned above, we consider an office as the environment where documents occur. There exist several proposals to distinguish and to categorize documents and information items in an office environment or within an office information system (Krönert 1985; Rabitti 1985; Schmidt and Teufel 1987). By such a classification in categories, we have to keep the retrieval function in mind. We do not need complex and expensive search methods if one knows exactly what one is looking for, because in this case one can specify the document in question with exact database keys and the retrieval is done using ordinary database algorithms. However, if one does not know exactly what one is looking for, one needs a sophisticated retrieval algorithm based on the properties of the unformatted document data. For this reason we define a document as shown in Figure 3.

A document has two main parts: formatted data and unformatted data. We call the formatted data *strong database data* and the unformatted data *free text*. Because there exist many solutions to handle documents through search keys, we will only consider how to treat free text documents. Thus, we only ask how to process natural language documents in an information system. We give *no* answer for the processing of multimedia documents.

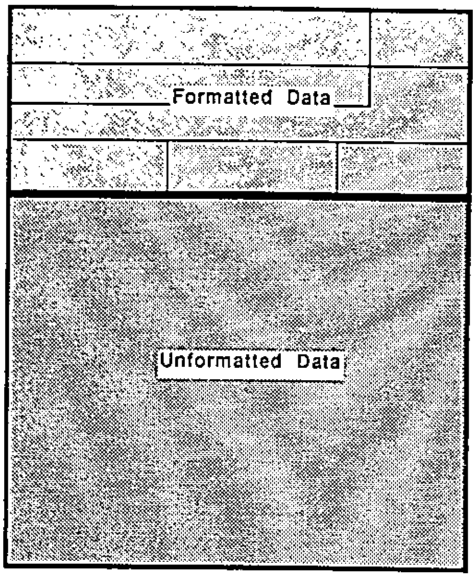

Figure 3. Document Consisting of Formatted and Unformatted Data

3. PREPARING NL DOCUMENTS

Full-text systems generally use two steps to prepare text (Schwarz 1982). First, one has to distinguish between information-holding and non-information-holding words, usually through application of a stoplist containing all function words and other words which hold no information (e.g., A, ABOUT, ABOVE, ACROSS). Second, all different morphological forms of a word must be reduced to one unique common form. This is done by the application of a stemming algorithm.

Depending on a stemming algorithm's result, one can subdivide stemming methods into linguistically correct and linguistically incorrect methods as shown in Figure 4.

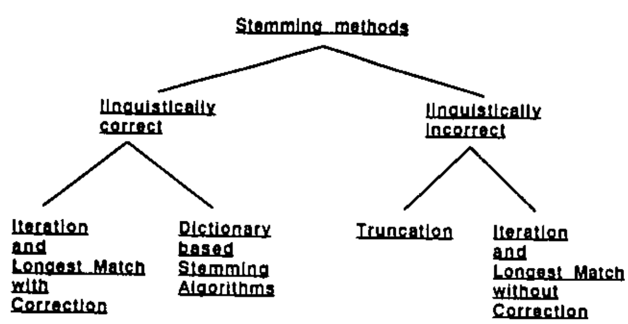

Figure 4. Classification of Stemming Algorithms

Keeping in mind that, in an information system, natural language texts are processed for reasons of indexing and retrieval, it is not necessary that the reduction of words to a common form be done in a linguistically correct way. The representation of a reduced word must only be interpreted by the computer and not by a qualified speaker of the language (Kuhlen 1974). Therefore, linguistic correctness is not a basic requirement for a stemming algorithm used in an information system.

Truncation is the worst algorithm with respect to linguistic correctness, because words are truncated after the first *k* characters. Porter's (1980) algorithm is an example for the iteration method, while Lovins' (1968) algorithm is based on the longest match method. Harman (1987) gave a comparative analysis of these algorithms.

The main reason for reducing the different morphological forms of a word to one common form is that different morphological forms imply *term dependencies* which, in an information system, must be determined and eliminated. In the context of an information structure there exists a BT-relation (BT = broader term) between the morphological forms and the common form of a word. In this sense the common form is a *formal broader term* of the morphological forms. Consider the words COMPUTE, COMPUTES, COMPUTED, COMPUTING, which all describe the same concept. We can choose COMPUT as the formal broader term of these different morphological forms. Additionally, we can see that COMPUT is a common form, but it is not a linguistically correct word stem.

Figure 5 presents the effect of the application of text reducing mechanisms (stoplist and an improved form of the Porter algorithm) to the intersection of the information traces of two documents. In the upper part of Figure 5 we see the intersection of the information traces of two similar documents, on the left the non-reduced texts and on the right the reduced texts. In the lower part we see the same for two totally different documents. We observe, that after using a stoplist and a stemming algorithm, similar documents appear similar, but dissimilar documents will be better discriminated.

In our system we have implemented an improved version of the Porter (1980) stemming algorithm. The main expansions are:

i) There are some additional rules in step 4 (e.g., $(m > 1)$ IAL $\rightarrow \epsilon$, $(m > 1)$ UAL $\rightarrow \epsilon$, $(m > 1)$ IATE $\rightarrow \epsilon$, . . .).

ii) We have a new step 5c, where I - if the last character of a word - is changed to Y, if there was any other rule applied before.

iii) Steps 2 through 5c are applied twice.

We can show, that we get better results with this improved algorithm for a *n*-gram based system, because more noisy word endings are eliminated.

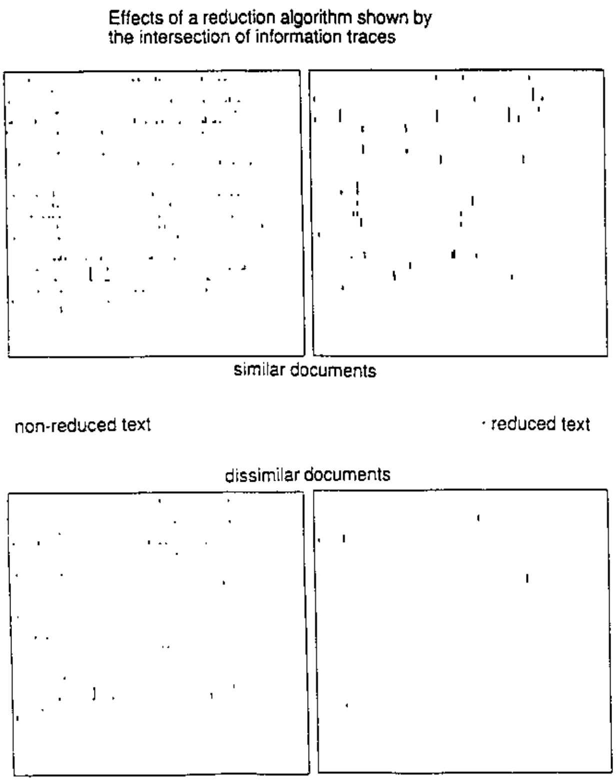

Figure 5. The Effect of Stoplist and Stemming Algorithm

4. IMPROVED n-GRAM INDEXING

In section 1, we showed that trigrams provide a fixed indexing set of 17,576 element. Different investigations have shown that only about 25 percent of all possible trigrams occur in natural language texts (Stolley 1978; Suen 1979; Teufel and Schmidt 1988). Therefore, the usage of trigrams becomes practical, although a theoretical motivation for their utilization has been rejected by Chudacek (1984).

A motivation for the usage of trigrams can be given through experimental investigations. Obviously, monograms are too small. Heuristically we can say, that digrams do not characterize any text well enough. Stolley (1978) states that the selectivity of digrams is too small, while tetragrams are hard to manage because of their great number. D'Amore and Mah (1985) have shown that digrams will be suitable only for small text collections. Experiments with the Cranfield test collection showed that trigram encoding of words performs noticeably better than the use of digrams (Willett 1979). In extensive tests we have shown that trigrams are the smallest units which are representative for natural language texts (Teufel and Schmidt 1988). The investigations have shown that each text produces in its information trace a highly characteristic subset of trigrams, and that tetragrams result in similar recall and precision values as trigrams. We tried also to give a motivation for the usage of trigrams in terms of entropy and computational costs (Teufel and Schmidt 1988).

Trigrams therefore seem to be the most reasonable choice. However, further investigations have shown that even with a good stoplist and stemming algorithm there remain in the document base about 6 percent (i.e., nearly 300) of the trigrams with great noise, as defined in an information theoretic sense, i.e., in analogy to Shannon's entropy definition. Thus, the noise is a measure of a trigram's concentration in the document collection (Salton and McGill 1983). This means that trigrams with high noise occur in most test documents. Table 1 shows the noise measured for some arbitrarily chosen trigrams.

Table 1. Trigram Statistics

Trigram	No. of Docs.	Tot. Freq.	Noise
ACT	714	1788	9.1411
BGR	11	25	3.3231
BIL	124	271	6.7059
COE	103	228	6.4671
CON	1200	3345	9.8598
DIV	51	91	5.2928
DIX	2	5	0.9710
DIZ	1	1	0.0
DMO	3	4	1.5000

To better discriminate among documents, trigrams with high noise must be eliminated. Therefore, we improve indexing by using the corresponding tetragram whenever a high noise trigram occurs. In the same way, we use a pentagram if the chosen tetragram also has high noise. This can result in a mixed indexing set for the documents (information trace), consisting mainly of trigrams, a few tetragrams and some pentagrams. Our test results show this improved n-gram indexing to be better than pure trigram indexing.

5. SEARCHING WITH NL TEXT COMPARISONS

The information trace of a document is the footprint of that document. That is, each natural language document maps some of its characteristics to the information trace. But note that the information trace is based on syntactic occurrences. Thus, the retrieval can only be done on a syntactical level.

What does it mean to search for natural language documents based on n-gram indexing? We search for natural language documents by determinating the similarity between information traces. Comparing information traces means retrieving documents by comparing the statistic-syntactical properties of the documents. With this method we cannot search using single term queries. Instead, we need a (unformatted) text describing the information need (e.g., a "somehow gotten" document itself).

As a similarity measure, we use the compound similarity function as defined by de Heer (1982). His calculation needs a threshold to select the "non-highly-frequent" trigrams. Because of the application of a stoplist and a stemming algorithm, we do not need such a threshold, which can only be determined experimentally. One advantage that results is that trigrams such as *THE*, eliminated in all contexts when we use a threshold, are maintained when necessary; for example, the *THE* of the word *ATHEISM*. Thus, using a stoplist and a stemming algorithm we eliminate such trigrams only where they have to be eliminated and we leave them where they contribute information. Through the application of such text reducing methods we no longer work only, on a syntactical basis, but also have a semantical base.

De Heer's similarity measure consists of two factors: a direct (Y_D) and an indirect (X_D) similarity factor. Each factor complies with the properties of a membership function and therefore defines a fuzzy subset on the document base *D*. The measure is defined as the algebraic sum of the two factors:

$$H_D = X_D + Y_D - X_D * Y_D$$

The direct factor can, for example, be the Dice measure, while the indirect factor is mainly based on the number of *n*-grams a document has in common with other documents. The task of the indirect factor is to add the similarity between two texts with regard to their similarity with other texts. Consider the following three "texts": t_1, t_2, and t_3.

t_1 = 'QUEUEING THEORY'
t_2 = 'QUEUEING THEORY, SERVER SYSTEMS'
t_3 = 'SERVER SYSTEMS'

Then,

$\pi(t_1)$ = {QUE, UEU, EUE, UEI, EIN, ING, THE, HEO, EOR, ORY}

$\pi(t_2)$ = {QUE, UEU, EUE, UEI, EIN, ING, THE, HEO, EOR, ORY, SER, ERV, RVE, VER, SYS, YST, STE, TEM, EMS}

$\pi(t_3)$ = {SER, ERV, RVE, VER, SYS, YST, STE, TEM, EMS}

Obviously, $\pi(t_1) \cap \pi(t_3) = \emptyset$. Therefore, the Dice similarity $S(t_1, t_3) = 0$. Thus, using only the direct similarity function, no correspondence between the "texts" t_1 and t_3 will be found, even though the terms used are somewhat synonymous. Nevertheless, both values $S(t_1, t_2) = 0{,}69$ and $S(t_2, t_3) = 0{,}64$ imply that there must exist a similarity between t_1 and t_3. Consequently, the direct similarity measure, S, represents a similarity which is worse than the similarity actually existing between these two "texts."

This problem occurs because the Dice similarity measure is not transitive. Thus, the idea behind the indirect similarity factor is transitivity; i.e., if two arbitrary documents are similar to a third document, then these two arbitrary documents cannot be totally different. Figure 6 shows the advantage of the compound similarity function over the direct similarity function with respect to the recall of 26 arbitrarily chosen queries. Figure 7 shows the same but with respect to the precision.

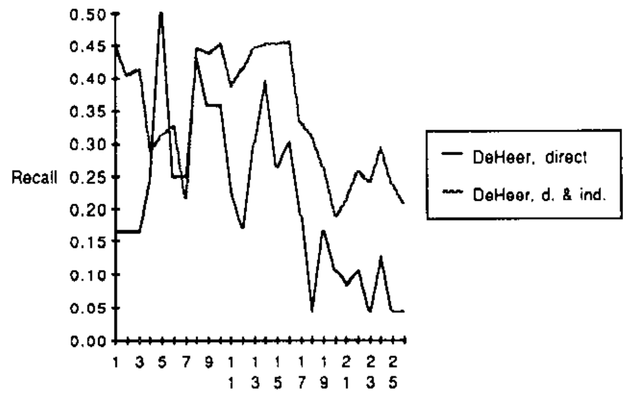

Figure 6. Recall of Direct and Compound Similarity Function

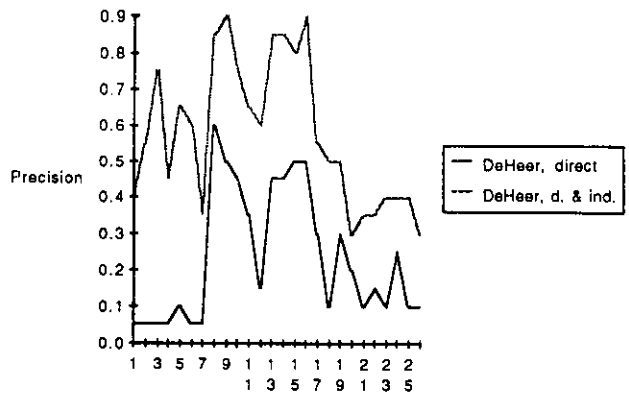

Figure 7. Precision of Direct and Compound Similarity Function

Both recall and precision are better if we use de Heer's compound similarity function. Considering all 26 queries, the recall is more than 10 percent and the precision is more than 25 percent better on average when we use the compound similarity function.

6. EXPERIMENTAL RESULTS

We developed a system based on the theory dicussed above. It is neither integrated in a broad information system, nor is it tuned to have minimal response times, because our primary aim was to test the theory. The implementation is done in MODULA-2 (about 8000 lines of source code, running on a SUN-workstation under UNIX) with a clearly defined module structure as shown in Figure 8. Because all important functions are exported, it is possible to use these modules in a larger system.

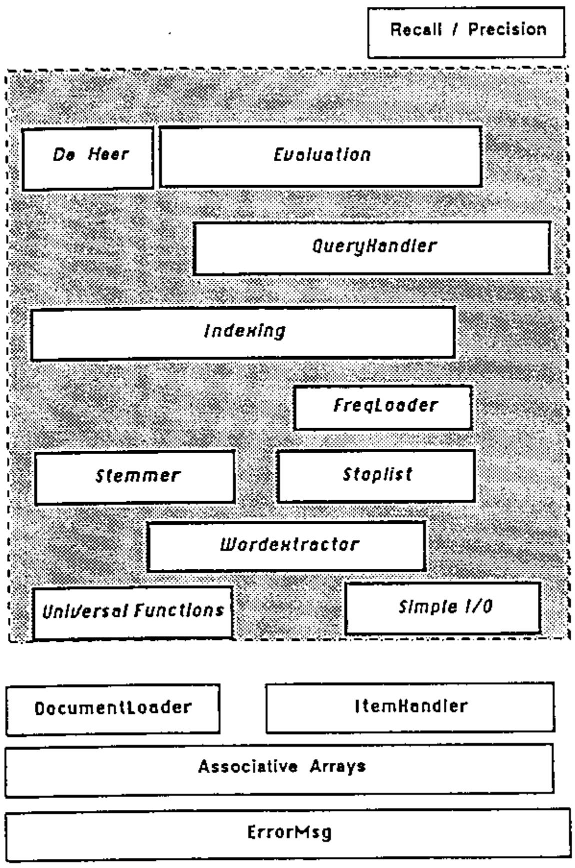

Figure 8. Module Structure

Two thousand four hundred seventy-two documents were used to test our system. These documents are elements of an INSPEC data collection of bibliographic data on Computer and Control and not only include author, title, keywords and abstract but also information on the place and year of publication, language, etc. Because this aforesaid strong database information is of no consequence to our tests, the documents are processed as follows:

- title
- keyword
- abstract of the document

For this test collection we generated a set of queries and their corresponding answer sets. Unfortunately, most of these queries are too short in text because the system does not work with single term or other short queries. The statistical properties of the system call for a minimum length of the query, e.g., the average length of the texts in the document base. Because the query length should be related to the average length of the documents in the base, we chose four representative queries with a length similar to the average document length. We then took 25 documents out of the answer sets of these queries. Thus we used 29 texts as the query set for this paper.

In the graphic presented in Figure 9, we can see the recall for the 29 queries. Figure 10 shows the precision for the same queries. The values of the recall range from a maximum of about 51 percent to a minimum of about 23 percent. Thus, we have an average recall of 36 percent with a standard deviation of 9.3.

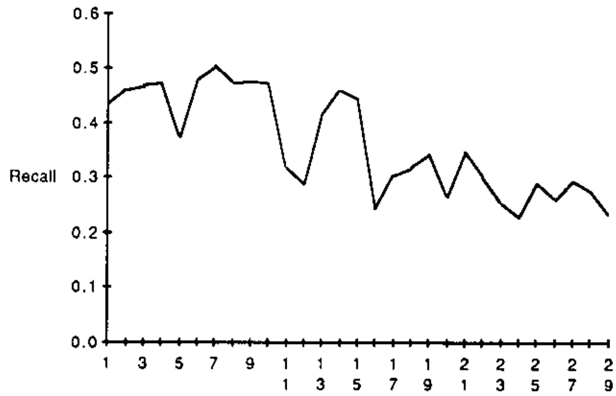

Figure 9. Recall of 29 Test Requests

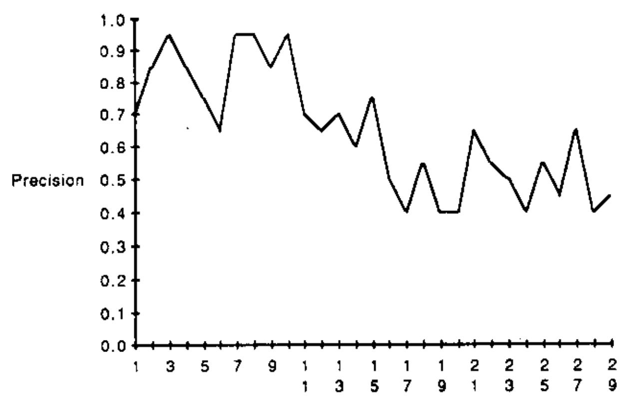

Figure 10. Precision of 29 Test Requests

To calculate the precision, we made a cut off after the twentieth document of our ranked answer list. The values of the precision range from a maximum of 95 percent to a minimum of 40 percent. The average value of the precision is 64.5 percent with a standard deviation of 18.7.

The scatter diagram of Figure 11 plots the relationship between precision and recall for the 29 queries. It shows in more than 50 percent of the cases values of precision greater than 60 percent with a recall above 30 percent. This scatter plot shows also that high values of precision are not necessarily accompanied by low values for recall.

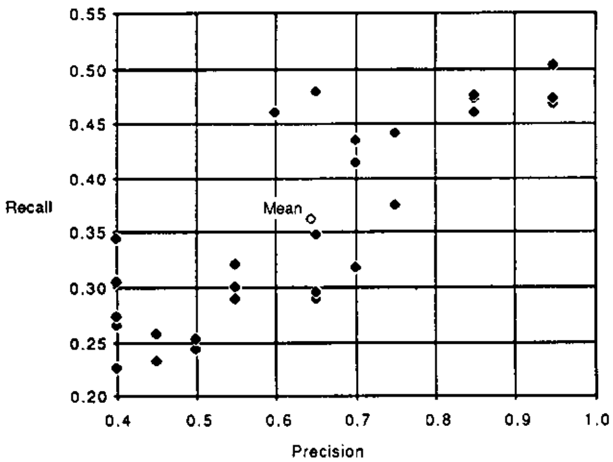

Figure 11. Precision Versus Recall

7. CONCLUSION: NATURAL LANGUAGE IN INFORMATION SYSTEMS

According to Faloutsos (1985), in an information system, three classes of access methods for natural language documents exist: full text scanning, inversion, and multi-attribute retrieval methods. While full text scanning is not satisfying, inverted keyword lists are widely used in information retrieval because they are easy to implement, yield useful results, and generate fast responses. Other approaches on handling natural language documents are based on so-called superimposed coding to create text signatures as developed by Tsichritzis and Christodoulakis (1983). Several systems are implemented using text signatures (Gebhardt 1987; Bertino, et al. 1986). They all use trigrams and/or tetragrams. Unfortunately, this method does not consider the frequency information contained in a text.

Concept-based systems are used more and more in information retrieval, but even concept-based methods present some problems

1) they work well -- but only in a restricted domain of discourse;

2) they need a consistent information structure which, in our opinion, should be known by the author of a document (or a query), because with the terms one uses, one must imply the same concepts as can be derived from the information structure;

3) text understanding cannot be done only on a semantic level, but on the syntactic, semantic, and pragmatic level; i.e., we also have to consider the context to extract the concepts of a text;

4) the operational basic units are words with all the problems they bear, such as different spellings, misspellings or the different morphological forms. In contrast to systems extracting the concept of a document using keyword lists, we can assume that these concepts are described in various (and extended) ways with the full text of the document. Thus, generating an information trace of this text is generating an information trace of the concepts described within the text.

There are also various systems providing a natural language user interface to a bibliographic or text database. For example, the IR-NLI from the University of Udine, Italy, (Brajnik, Guida, and Tasso 1987) allows natural language queries. But we think that humans behave according to the Principle of Least Effort (Zipf 1949), so we do not believe that the user wants to type in questions like "I would like to have references about . . . I am also interested in . . . and I like German or English documents." The user wants only to fill out the points above--or, as a query he wants to submit an already available document describing his information need. The latter is a normal case in an electronic office and the point where we started.

Our test results show that *n*-gram indexing can be considered to be a good alternative to term indexing. Furthermore, it provides several advantages over term indexing. First, this method allows natural language requests and is highly tolerant against spelling errors and the like. That is, *n*-gram indexing and retrieval is very robust and yields good results in noisy environments where texts are generated with spelling errors, incorrect morphological forms, etc., or where texts are received (from a network) through noisy channels. Second, it works with a fixed indexing set (vocabulary), which does not change as language expands: Newly defined terms or words can be processed in the same way as terms already known (this is also valid for licence numbers, trademarks, or chemical formulas and protein structures consisting of characters other than letters or meaningful words). Third, no information structure dependent on a special language is needed. Thus, the same system can be used without any changes or additions for document bases in different languages. Fourth, once the documents are indexed, the problem of re-indexing never occurs, even when new documents are added to the document base.

There are also some disadvantages on *n*-gram indexing and retrieval. The method is very processor-intensive, because comparing information traces and calculating the similarity values are time-intensive processes (many floating-point operations). Special purpose hardware will help here. Another drawback of the system, as implemented, is that, in order to generate a response to a request, all documents have to be taken into consideration. We can overcome this handicap by defining clusters on the document base (of course, to this we can use our system, too) and determine a centroid for each cluster.

In conclusion, we can say that the method introduced is simple and easy to understand, it is robust and it works.

Typical environments where it can be used are law offices, patent offices, or in general, environments where texts of different subjects occur (perhaps in a noisy manner).

8. REFERENCES

Angell, R. C.; Freund, G. E.; and Willett, P. "Automatic Spelling Correction Using a Trigram Similarity Measure." *Information Processing Management*, Vol. 19, No. 4, 1983, pp. 255-261.

Bertino, E.; Converti, A.; Giuntini, A.; Luxardo, G.; Rabitti, F.; and Savino, P. *Text Retrieval Techniques for Office Documents*. Pisa: Olivetti Research Report, 1986.

Brajnik, G.; Guida, G.; and Tasso, C. "Design and Experimentation of IR-NLI: An Intelligent User Interface to Bibliographic Databases." In L. Kershberg (ed.), *Expert Database Systems*, Menlo Park, CA: The Benjamin/ Cummings Publishing Company, 1987.

Chudacek, J., and Benshop, C. A. *Reconstructie van Tekstdelen uit hun Syntactische Sporen*. IWIS-TNO Report, A 81 IN 74 10, The Hague, 1981.

Chudacek, J. *Non-grammatical Language Processing.* Preprint, Institute TNO for Mathematics, Information Processing and Statistics, The Hague, 1984.

D'Amore, R. J., and Mah, C. P. "One-time Complete Indexing of Text: Theory and Practice." *Proceedings of the Eighth Annual International ACMSIGIR Conference*, Montreal, 1985.

de Heer, T. "Experiments with Syntactic Traces in Information Retrieval." *Information Storage and Retrieval*, Vol. 10, 1974, pp. 133-144.

Faloutsos, C. "Access Methods for Text." *Computing Surveys*, Vol. 17, No. 1, 1985, pp. 49-74.

Gebhardt, G. "Test Signatures by Superimposed Coding of Letter Triplets and Quadruplets." *Information Systems*, Vol. 12, No. 2, 1987, pp. 151-156.

Harman, D. "A Failure Analysis on the Limitations of Suffixing in an On-line Environment." *Proceedings of the Tenth Annual International ACMSIGIR Conference on R & D in Information Retrieval*, New Orleans, 1987.

Jucquois-Delpierre, M. "Information Retrieval in the Office Environment." *International Forum on Information and Documentation*, Vol. 12, No. 3, 1987, pp. 15-18.

Karlgren, H. "Homeosemy: On the Linguistics of Information Retrieval." In D. E. Walker, H. Karlgren, and M. Kay (eds.), *Natural Language in Information Science*, Stockholm: Skriptor, 1977.

Krönert, G. "International Standard for an Office Document Architecture Model." *Journal of Information Sciences*, Vol. 10, 1985, pp. 69-78.

Kuhlen, R. "Morphologische Relationen durch Reduktionsalgorithmen." *Nachrichten für Dokumentation*, Vol. 25, No. 4, 1974.

Lovins, B. J. "Development of a Stemming Algorithm." *Mech. Transactions of Computing Linguistics*, Vol. 11, 1968, pp. 11-31.

Mah, C. P., and D'Amore, R. J. "Complete Statistical Indexing of Text by Overlapping Word Fragments." *SIGIR-Forum*, Vol. 17, No. 3, 1983, pp. 6-16.

Mundt, B. Korrektur von Wörtern mit einem Elementarfehler unter Verwendung von Trigrammen. Unpublished PhD Thesis, Technische Universität Berlin, Berlin, 1987.

Porter, M. F. "An Algorithm for Suffix Stripping." *Program*, Vol. 14, No. 3, 1980, pp. 130-137.

Rabitti, F. "A Model for Multimedia Documents." In D. Tsichritzis (ed.), *Office Automation*, Berlin: Springer-Verlag, 1985, pp. 229-250.

Salton, G., and McGill, M. J. *Introduction to Modern Information Retrieval*. London: McGraw-Hill, 1983.

Schmidt, S., and Teufel, B. "CIX: A Tool for the Controlled Exchange of Information in a Software Development Environment." *Proceedings of the IEEE Week Montech '87*, Compint '87, Montreal, November 9-11, 1987.

Schwarz, C. "Freitextrecherche - Grenzen und Möglichkeiten, Anmerkungen aus der Sicht der Informationslinguistik." *Nachrichten für Dokumentation*, Vol. 33, No. 6, 1982.

Stolley, J. O. *String Retrieval in German Texts by Means of Trigrams*. Philips Research Report, Pub. No. IDR-R-ST/7809/2039, Eindhoven, 1978.

Suen, C. Y. "N-Gram Statistics for Natural Language Understanding and Text Processing." *IEEE Transactions Pattern Analysis Mach. Intell.*, Vol. PAMI-1, No. 2, 1979, pp. 164-172.

Teufel, B., and Schmidt, S. "Full Text Retrieval Based on Syntactic Similarities." *Information Systems*, Vol. 13, No. 1, 1988, pp. 65-70.

Tsichritzis, D., and Christodoulakis, S. "Message Files." *ACM Transactions of Office Information Systems*, Vol. 1, No. 1, 1983, pp. 88-98.

Willett, P. "Document Retrieval Experiments using Indexing Vocabularies of Varying Size. II. Hashing, Truncation, Digram and Trigram Encoding of Index Terms." *Journal of Documentation*, Vol. 35, No. 4, 1979, pp. 296-305.

Zamora, E. M.; Pollock, J. J.; and Zamora, A. "The Use of Trigram Analysis for Spelling Error Detection." Information Processing Management, Vol. 17, No. 6, 1981, pp. 305-316.

Zipf, G. K. *Human Behaviour and the Principle of Least Effort*. Reading, MA: Addison-Wesley, 1949.