Snippet Search: a Single Phrase Approach to Text Access

Jan Pedersen and Doug Cutting (Xerox PARC)
John Tukey (Xerox PARC and Princeton University)

Information Access

Due to the ever increasing affordability and accessibility of very large, online, text collections, processing natural language texts for search and retrieval has recently been the focus of heightened attention, although researchers have been active in the field since the early sixties. Numerous approaches have been attempted, but they all suffer from the obvious difficulty that search and retrieval is quintessentially a cognitive task; the degree of automatic language understanding required for a completely automatic solution is clearly beyond the bounds of current technology. Instead, heuristic search techniques attempt to match an admittedly incomplete query description with an admittedly incomplete set of features extracted from the texts of interest. The challenge therefore lies in the development of procedures that more effectively bridge the gap between an individual's partially stated desires and a universe of text, which typically appears, computationally, as a sequence of uninterpreted words.

Many of these procedures are statistical in nature; they take advantage of repeated occurrences of the same word to infer relations between documents, and between queries and documents. [1] For example, similarity search induces a "relevance" ordering on the text collection by scoring each document with a normalized sum of importance weights assigned to each word in common between it and the query, where the importance weights depend upon document and collection, or corpus, frequencies [28]. A more formal approach scores documents with their estimated probability of relevance to the query by adopting a text model which assumes word occurrences are sequentially uncorrelated and training on a set of known relevant documents [3, 30]. In contrast, polysemy (one word having multiple senses) and word correlation is directly addressed by Latent Semantic Indexing, which attempts to exact characteristic linear combinations through a singular value decomposition of a word co-occurrence matrix [12]. The availability of interdocument similarity measures suggests clustering, which has been pursued both as an accelerator for conventional search and as a query broadening tool [31]. Finally, linear discriminant analysis has been deployed to

[1] Here "document" need not correspond to any particular organization. It might be a chapter within a book, a section within a chapter, or an individual paragraph. In the following we will assume that the set of documents forming a corpus is an exhaustive and disjoint partition of that corpus.

classify documents based on a training set which matches features, including word overlap and word positioning, with relevance to previous queries [13].

Another suite of techniques attempt to enrich the basic feature set by annotating words with their lexical and syntactic function. For example, fast lookup algorithms from computational linguistics reduce words to their stems [18]. Hidden Markov Modeling has been successfully employed to induce part-of-speech tags, given a lexicon, with greater than 95% accuracy [20]. An extension of this technique, known as the inside-outside algorithm, promises to provide a method for inducing a stochastic grammar given sufficient training text [1, 14]. Less ambitious procedures aim at robustly extracting noun phrases given a sequence of part-of-speech tags [5]. Word co-occurrence relations have been exploited to order alternatives in cases of lexical ambiguity [15, 11]. Non-parametric classification procedures have been used to detect sentence boundaries in the face of typographic ambiguity [23].

A typical scenario confronts a user with an information need, or interest, with a corpus of natural language text documents. The task is to satisfy that need, usually by delivering one or more relevant documents, or, alternatively, by indicating persuasively that no such documents exist in the corpus. This is accomplished by extracting from each document a feature set, and providing the user with a tool which allows search over these features in some prescribed fashion. For example, a standard boolean search technique assumes the feature set is one or more words extracted from the text of the document, and the query language is boolean expressions involving those words [17]. Since it is anticipated that the corpus may be very large, construction of a feature index by preprocessing each document is a standard search accelerator [26].

Conventional search techniques are cast within a framework that might be referred to as the library-automation paradigm. It is presumed that the cost for evaluating a query is sufficiently high that a single iteration must return as high quality, and as complete, a response as possible. This is in keeping with online systems that charge for connect time, and is reflected in evaluation criteria that discount the cost of query formulation and search resolution while measuring the precision and recall levels for the ranked set of documents which is implicitly presumed to be the search result [26]. Ironically, the best improvements to date, with respect to these criteria, come from a incremental query reformulation technique, known as relevance feedback [27].

We have argued that the availability of high-interaction user interfaces on modern workstations should modify this model, and have proposed an alternative paradigm, which we refer to as information theater [8]. In essence, the user is brought back into the loop by making interaction between the user and partial search results an explicit component of query resolution. The user is employed as an active filtering and query reformulation agent, which is only plausible if one provides rapid response to user intervention. We have developed a search method, Snippet Search, a form of guided boolean search with proximity, and an associated browsing tool, which exemplifies these principles.

Snippet Search's basic underlying assumption is that short queries, consisting of a few search terms, are by their very nature radically incomplete. Hence, query repair and elaboration through user interaction and iteration are essential to achieve an adequate combination of recall and precision. This calls for a high-interaction interface rapidly delivering results to the user in a way that can be quickly appreciated, and for a search method whose operation is intuitive and which offers information as to which next step is likely to be most effective in achieving a desired result.

Related Work

Boolean keyword search is a well-known search technique in information retrieval [26]. Essentially, a set of terms, typically individual words, or word stems, is extracted from the unrestricted text of each document in a corpus. Search then proceeds by forming, as a query, a boolean expression in terms of these keywords which is resolved by finding the set of documents that satisfy that expression. For example, a typical query might consist of the conjunction of two search terms. Documents that contain both terms in any order and any position would then be returned. Disjunction and, less frequently, negation are also likely to be supported.

Unconstrained boolean search represents a document as a set of keywords; sequence information is ignored. Proximity search paradigms modify this representation by placing non-boolean nearness constraints on otherwise standard boolean queries [26]. A proximity operator is introduced that demands that two given search terms occur within some given distance (expressed as a number of characters or a number of words) in order for the basic conjunction to be satisfied. For example, the a two-word query may be narrowed by requesting that the two search terms appear within one word of each other, either in any order or in the given order.

Proximity search enables the user to form phrase-like queries; that is, a local combination of terms is treated as a search unit. This assumes considerable importance when one recalls that a query is a representation of an "information need". Often the concepts inherent in this information need are not expressible as single words; instead phrases and even complete sentences must be employed to adequately specify the thought. Boolean queries allow for the partial expression of these sorts of combinations, but they also clearly allow for too many unwanted matches. Higher precision is achievable by making use of nearness constraints. However, complete specification of term order may be detrimental since it is a property of most natural languages (including English), that phrasal units may be rewritten in multiple ways without a change in meaning. For example "dog's ankle" and "ankle of a dog" express the same concept. Hence, the application of proximity constraints must be strong enough to filter out disconnected occurrences, yet flexible enough to account for trivial language variations.

Traditional applications of boolean or proximity search within the library-automation paradigm result in a candidate set of documents which satisfy the search criterion — the "hits". The user must then judge the effectiveness of the query by perusing these documents, a potentially time consuming operation, especially if document titles are insufficient to distinguish relevant from non-relevant hits. In fact, there is empirical evidence that boolean searches tend to fall into two classes, those whose result sets contain only a very few hits (narrow query), and those that result in a great many hits (broad query) [2, 22]. In the case of only a few hits, the user is left with the uncomfortable feeling that something may have been missed, which leads to a desire to broaden the existing query. However, if the query is over-broadened, the user is presented with far too many hits, and the task of separating out the relevant documents from the mass becomes daunting.

The problem with boolean search is that the user is provided with little or no assistance in query reformulation. A dictionary of available search terms can aid the search for alternative terminology, as can an online domain-specific thesaurus [16], but, often, the help of a highly trained intermediary, such as a research librarian, is required to reach a desirable reformulation.

One solution is to provide enough information about each hit that the user can rapidly determine the contextual usage, and hence arrive at a relevance judgement and, possibly, a reformulation, without necessarily scanning the entire text. Paper-based keyword-in-context indices (and other styles of permuted indices) offer a solution for the case of single term queries. The user enters the index with a single term, the "gutter word", and finds single lines of text for each occurrence, with the gutter word aligned in a column and lines ordered alphabetically by the text appearing after the gutter word, with wraparound [21]. The choice of alphabetic sort key for the lines of context can be less than optimal if one is searching for related phrases containing the search key, since the words determining the phrase will often not follow the key word directly. An alternative sort key, that captures much of this intuition, has been employed successfully in the generation of an index to titles in Statistics and Probability [24, 25]. Computerized versions of these sorts of indices exists in a variety of different forms [6, 32], yet few, if any, elaborate on the basic query and display strategy.

Snippet Search

Snippet Search addresses these issues by allowing the user to directly inspect the space of phrases generated by a set of terms of interest. The intention is to aid query reformulation by exposing the user to the range of variation present in the corpus. For example, a snippet search keyed by the single term "information" might display phrases such as "information storage and retrieval", "advances in information retrieval", "sensory information", and "genetic information" among others, each of which occurs in the corpus.

From the user's perspective, Snippet Search resembles a phrase search where the query specifies constituents and results are returned which contain these and new constituents, organized in a fashion that emphasizes the new rather than the old. Query formulation consists of specifying one or more "constituents" in a way that requires little or no query syntax. These constituents are then matched against the corpus using a heuristic which interprets them as a boolean conjunction with a proximity constraint. Then, instead of returning matching documents and treating the search as if it were complete, as would a standard boolean search, Snippet Search returns returns a short textual context surrounding the matches. These *snippets* are intended to contain sufficient context to distinguish usage, but not so much text as to distract the reader or clutter the display.

Figure 1: The Text Browser

Figure 2: Snippet Query Specification

The current heuristic returns the text surrounding the search terms plus one other "significant" word, where significance is operationally defined by *not* being on some prespecified list of non-topic-influencing words (a *stop list*). The neighboring non-stop word provides distinguishing context and is highlighted in the display to draw the user's attention to what is new, rather than what was input (the query terms). If this context is insufficient to distinguish usage, the user is encouraged to ask for more (a snippet operation called "extend"). If the context shows a word combination which is *a priori* uninteresting, all snippets with similar word structure can be deleted (an operation called "forget"). Note that the "forget" operation is, in effect, boolean negation by example.

Since the times required for each of these operations can be made small, the overall effect is to encourage incremental query reformulation based on occurrences as they appear in the corpus of interest. In the case where the short context is sufficient to indicate that the snippet is indeed relevant, the user may proceed directly to the corresponding document (an operation called "view").

An Example

We have implemented a version of Snippet Search which realizes the strategy just outlined. In particular, Snippet Search is one of the search modes supported by the Text Database architecture (TDB) [10], a software artifact implemented in Common Lisp [29] which is directed towards fast prototyping of retrieval systems. A user interface to TDB, known as the Text Browser, uses the Interlisp-D [7] window system to present a multi-paradigm text search and retrieval tool (see figure 1). Currently, two search modes are supported over the same corpus, Similarity Search and Snippet Search. The first two panels concern themselves with snippet query specification and the presentation of results. The third panel is for the scrollable display of documents. The last two panels are concerned with Similarity Search. The ordering is not particularly significant, although it is anticipated that Snippet Search will be most useful for fairly directed queries, the results of which can then seed a browsing method, such as Similarity Search.

Prior to search, the target corpus (in this example, Grolier's encyclopedia, 64 Megabytes of ASCII text) was processed by an indexing engine that extracted the content words

Figure 3: Results of “movie” Query

Figure 4: Query Reformulation

Figure 5: Results of “movie industry” Query

(ignoring words on a stop list) in each document (in this example, one of the twenty-seven thousand articles in the encyclopedia), normalized them through the removal of inflectional morphology, and recorded their sequential offsets in an inverted index [9].

Search then proceeds by specifying a set of words which will form the components of a phrase match criterion (see figure 2). In this example the user is interested in phrases that include the word “movie” (or its inflectional variations). Note that the interface reports the marginal frequency of the search term, and the number of hits currently found. The query is resolved by interpreting it as a boolean conjunction with a proximity constraint; a match occurs if the all query terms occur with no more than one content word gap between them. In the example, since there is only one query term, all instances of “movie” match.

The result of a query is a set of text snippets, each satisfying the phrase match criterion snippet (see figure 3). In the example each instance of “movie” generates up to two overlapping snippets (for a total of 263). These are presented in a stylized fashion to aid perusal by the user. The display heuristic presents the query terms plus one additional non-stop word and all the intervening (unindexed) text, containing spaces, punctuation and stop words. The additional non-stop word is intended to provide distinguishing context. The inclusion of the intervening unindexed text provides useful syntactic information, especially through function words.

To focus the user’s attention on new information, snippets are formatted so that the additional non-stop word, is placed adjacent to an easily recognizable location. This has the effect of columnating these contexts next to a vertical strip of white space, known as the “gutter”. The gutter word is highlighted with a bold font, and the query terms are distinguished, but not as heavily emphasized, with an italic font. The final display is reminiscent of a keyword-in-context index, with the crucial difference that each gutter word is new information (not just part of the match criterion), and, each line may be the result of a multi-term query.

As with boolean search, no particular ordering of snippets is implied by the query resolution mechanism. In practice, it is often convenient to organize snippets to correspond to a particular scan order through an inverted index, since partial results may then be returned before the completion of the entire query. This is especially useful for queries with a large number of hits, since the user may begin perusal of the partial results without waiting for search termination. Other presentation orderings may also be useful. In particular, snippets may also be sorted by the gutter word, or by schemes that extract a sort key from the sequence of content words. This could be accomplished either incrementally or after search termination.

In this example, the user can easily see by inspection that “movie” occurs in phrases such as “silent movie”, “movie theater”, “movie industry”, as well as many others. To view more snippets without scrolling, the user at this stage may choose to eliminate phrases similar (in the sense of having the same gutter word) to the one currently selected by buttoning “forget” in the query panel. Alternatively, the user may narrow the query by picking one of the completions for further study. If the user re-evaluates the query adding “industry” as an additional term (see figure 4), twelve hits are returned (see figure 5). Again, by inspection it

Figure 6: Extending a Snippet

is easy to see, for example, that the article titled "Rome" has a reference to the Italian movie industry. The snippet "*movie industry* **operated**" is not especially revealing; however, the user may button "extend" to enlarge the viewed context "*movie industry* operated under a self-imposed **code**" (see figure 6). Any one of the snippets may be selected, and the associated document viewed (with the snippet highlighted) by buttoning "view" in the query panel.

Algorithms

It is presumed that each document, d, in a larger corpus is a sequence of words,

$$d = \{w^d_1, w^d_2, \ldots, w^d_{n_d}\},$$

where n_d is the number of word instances (tokens) in document d. [2] It will be convenient in the following to consider each word occurrence as an word interval of length 1. That is, let

$$(d, s, e) = \{w^d_s, w^d_{s+1}, \ldots, w^d_e\},$$

then

$$d = \{(d, 1, 1), (d, 2, 2), \ldots, (d, n_d, n_d)\}.$$

In the case of intervals of length one, let $(d, s) = (d, s, s)$.

An inverted map can be produced by preprocessing each document. This map identifies each word (type) with the sequence of length one intervals that contain it as a token,

$$\begin{aligned} I(w) \quad = \quad & \{(d^w_1, s^w_{1,1}), (d^w_1, s^w_{1,2}), \ldots, (d^w_1, s^w_{1,n^w_1}), \\ & (d^w_2, s^w_{2,1}), (d^w_2, s^w_{2,2}), \ldots, (d^w_2, s^w_{2,n^w_2}), \ldots, \\ & (d^w_{n_w}, s^w_{n_w,1}), (d^w_{n_w}, s^w_{n_w,2}), \ldots, (d^w_{n_w}, s^w_{n_w,n^w_{d_{nw}}})\}, \end{aligned}$$

where d^w_i is the i^{th} document containing an instance of w, $s^w_{i,j}$ is the word offset of the j^{th} instance of w in d^w_i, n^w_d counts the number of instance of w in d, and n_w is the number of

[2] Here, and in the following, word instances are non-stop words.

documents in which w occurs. If there exists an ordering on documents, $\prec$, (we can always construct such an ordering), then we will require that $I(w)$ is ordered as follows:

$$d_i^w \prec d_j^w \text{ if } i < j$$

and

$$s_{i,j}^w < s_{i,k}^w \text{ iff } j < k.$$

In this setting, it is natural to define disjunction as a merge operation on sequences of word intervals. That is, the result of a disjunctive query $q = \{w_1^q, w_2^q, \ldots, w_{n_q}^q\}$ is defined to be:

$$\bigsqcup_{i=1}^{q} I(w_i^q),$$

where $\bigsqcup$ denotes an n-ary merge operation on ordered sequences, as can be implemented by a priority queue in time proportional to $\log n_q \sum_{i=1}^{q} |I(w_i^q)|$, where $n_q = |q|$ [19].

Similarly, conjunction with proximity can be seen as a specialized merge operation. Suppose q is satisfied by a sequence of words if every word w_i^q occurs at least once in the sequence, and the total length of the sequence is no more than $|q| + p$, where $p \geq 0$ is the proximity parameter. Let $I_i = I(w_i^q)$, and define f_j^i to be the j^{th} interval in I_i. Set $c_i = 1$ for all i, and let $f_i = f_{c_i}^i$, initially the first interval in I_i. Let the I_i's be ordered by considering the f_i's;

$$I_i \prec I_j \text{ iff } f_i \prec f_j,$$

Let (d_i, s_i) refer to f_i. Consider the following algorithm:

```
0 Result = $\emptyset$

1 Sort the $I_i$'s

2 if $d_1 = d_{n_q}$ and

      $\sum_{i=1}^{n_q-1} s_{i+1} - s_i = s_{n_q} - s_1 \leq p + n_q - 1$

  then append $(d_1, s_1, s_{n_q})$ to Result

3 set $c_1 = c_1 + 1$

4 if $c_1 > |I(w_1^q)|$ return Result else goto 1
```

As defined here, not every interval that satisfies the query condition is necessarily returned; in cases where two candidate intervals share left edges, only the shorter will be selected. For example, suppose the query pattern is "xy" and $p = 1$, then the sequence "xyy" will generate only one result interval, although two could be found. It is possible, with the addition of backtracking, to modify this algorithm to be fully correct, but for the purposes of clarity, we will not pursue the issue further in this paper.

In the worst case, the inner loop of this algorithm is executed $\sum_{i=1}^{n_q} |I(w_i^q)|$ times, while the cost of step [1] is proportional to $n_q \log n_q$, hence the overall time complexity of this algorithm is proportional to $n_q \log n_q \sum_{i=1}^{n_q} |I(w_i^q)|$.

Summary and Further Steps

Information access to natural language text needs the active participation of a user, both because fully automated techniques would require deeper understanding of natural language text than can be automated today, and because users often need to sharpen their ideas as they proceed. While access based on words typically requires multiple words to be sufficiently expressive, boolean conjunction with a built-in proximity constraint can provide effective access via a single phrase (not a single word!) while keeping the query language so simple as to be nearly unnoticeable.

Presentation of short text stretches, appropriately organized, rather than paragraphs (or even whole documents), makes it much easier for the user to comb through the returns of the present query, either as a source of documents to be viewed in detail, or as a source of suggestions as to how an improved query might be formulated. Repeated cycles of scanning and query adjustment can be rapid and effective.

Snippet Search could be extended in a variety of ways. First, the current heuristic for choosing the nearby content word could probably be improved by statistically evaluating the likely topic-determining value of a list of nearby candidate words. This could be accomplished either by considering importance weights (as defined by similarity search), or by computing a dispersion measure based on a clustering of the corpus [4].

If a stochastic part-of-speech tagger were available it could be employed in at least two ways. Since part-of-speech tagging can be sense distinguishing (for example, "package" as a noun has quite a difference sense than "package" as a verb), the strategy would be to segregate (or sort) snippets based on the inferred part of speech of the query terms. Another use would feed an tagged-extended context to a noun-phrase recognizer in order to select a syntactically coherent subset for display purposes.

Snippet Search is most useful by generating candidate phrases given a single term query. In this case, it may not be necessary to generate an exhaustive listing. Instead, similar phrases could be represented as a single paradigm. This reduction to equivalence classes would expose the variation persent in the corpus more readily than the listing of repeated instance of the same (or similar) phrases.

Multi-term queries can be over-constraining; some form of automatic broadening may be appropriate if only a few hits are found. This could be accomplished by selectively weakening the match criterion until, at the extreme, it becomes a disjunction, rather than a conjunction. Such a strategy would differentially weight snippets based on the degree of match — and sort them accordingly.

The validity of any one of these extensions should be based on some evaluation criterion. However, evaluation procedures developed for the library-automation paradigm are not directly applicable to Snippet Search. In particular, standard measures of precision and recall presume a fixed query and a well-defined highly-structured result, typically an ordered list of documents sorted by some measure of relevance. Snippet Search is not intended to produce such a list, nor be driven by a single query. The overarching concepts of precision and recall still need to be considered, but in a broader arena, with possibly new definitions. Additional questions arise as well, such as the total time required to discover a particular fact and the extent of query modification encountered in pursuit of an information need.

One approach would adopt the view that automatic evaluation is impractical. Instead, an experiment that involves monitoring volunteer users given a fixed set of retrieval tasks could effectively shed light on the efficiency of Snippet Search in various settings.

References

[1] J.K. Baker. Trainable grammars for speech recognition. In D.H. Klatt and J.J. Wolf, editors, *Speech Communications Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550, 1979.

[2] D. C. Blair and M. E. Maron. An evaluation of retrieval effectiveness for a full-text document-retrieval system. *CACM*, 28(3):289–299, March 1985.

[3] A. Bookstein and D.R. Swanson. Probabilistic models for automatic indexing. *Journal of the American Society for Information Science*, 26(1):45–50, January–February 1975.

[4] John B. Carroll, Peter Davies, and Barry Richman. *The American Heritage Word Frequency Book.* Houghton Mifflin, New York, 1971.

[5] K. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 1989.

[6] ATT Corporation. Global regular expression program. Licensed software, 1979.

[7] Xerox Corporation. *Interlisp-D Reference Manual.* Xerox AIS, 1987.

[8] D. R. Cutting, P.-K. Halvorsen, J. O. Pedersen, and M. Withgott. Information theater versus information refinery. In *AAAI Spring Symposium on Text-based Intelligent Systems*, Stanford University, Stanford, CA, March 1990. Also available as Xerox PARC technical report SSL-89-101.

[9] D. R. Cutting and J. O. Pedersen. Optimizations for dynamic inverted index maintenance. In *Proceedings of SIGIR'90*, September 1990. Also available as Xerox PARC technical report SSL-90-10.

[10] D.R. Cutting, J. Pedersen, and P.-K. Halvorsen. An object-oriented architecture for text retrieval. In *Conference Proceedings of RIAO'91, Intelligent Text and Image Handling, Barcelona, Spain*, pages 285–298, April 1991. Also available as Xerox PARC technical report SSL-90-83.

[11] Ido Dagan and Alon Itai. A statistical filter for resolving pronoun references. In *Proceeedings of the 7th Israeli Symposium on Artificial Intelligence and Computer Vision*, 1990. To Appear.

[12] S. Dumais, G. Furnas, T. Landauer, S. Deerwester, and R. Harshman. Using latent semantic indexing to improve access to textual information. In *Proceeedings of CHI'88*, pages 281–285, 1988.

[13] Norbert Fuhr and Chris Buckley. Probabilistic indexing from relevance feedback. In Jean-Luc Vidick, editor, *Proceedings of SIGIR'90*, pages 45–61. ACM SIGIR, Press Universitaires de Bruxelles, September 1990.

[14] T. Fujisaki, F. Jelinek, J. Cocke, E. Black, and T. Nishino. A probabilistic method for sentence disambiguation. In *Proceedings of the International Workshop on Parsing Technologies*, August 1989.

[15] Donald Hindle. Noun classification from predicate-argument structures. In *Proceeding of the 28th meeting of the ACL, Pittsburg, Pennsylvania*, pages 268–275, 1990.

[16] H.P.Frei, M.Bärtschi, and J.-F. Jauslin. Caliban: Its user-interface and retrieval algorithm. Technical Report 62, Institut für Informatik, ETH, Zürich, April 1985.

[17] IBM Germany, Stuttgart. *Storage and Information Retrieval System (STAIRS)*, April 1972.

[18] L. Karttunen, K. Koskenniemi, and R. Kaplan. A compiler for two-level phonological rules. Report CSLI-87-108, Center for the Study of Language and Information, 1987.

[19] D. Knuth. *The Art of Computer Programming*, volume 3: Sorting and Searching. Addison-Wesley, 1973.

[20] J. M. Kupiec. Augmenting a hidden Markov model for phrase-dependent word tagging. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 92–98, Cape Cod, MA, 1989. Morgan Kaufmann.

[21] H.P. Luhn. Keyword-in-context for technical literature. ASDD Report RC-127, IBM Corporation, Yorktown Heights, N.Y., August 1959.

[22] Gary Marchionini. Information-seeking strategies of novices using a full-text electronic encyclopedia. *Journal of the American Society for Inforamtion Science*, 40(1):54–66, 1989.

[23] Michael Riley. Some applications of tree-based modelling to speech and language. In *Proceedings of the DARPA Speech and Natural Language Workshop, Cape Cod, Massachusetts*, pages 339–352, October 1989.

[24] I.C. Ross and J.W. Tukey. *Index to Statisitics and Probability: Permuted Titles, A–Microbiology*, volume Vol. 3 of *Information Access Series*. R&D Press, Los Altos, CA., 1975. Also available form the American Mathmatical Society, Providence, R.I.

[25] I.C. Ross and J.W. Tukey. *Index to Statisitics and Probability: Permuted Titles, Microbiology–Z*, volume Vol. 4 of *Information Access Series*. R&D Press, Los Altos, CA., 1975. Also available form the American Mathmatical Society, Providence, R.I.

[26] G. Salton. *Automatic Text Processing*. Addison-Wesley, 1989.

[27] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297, June 1990.

[28] G. Salton, A. Wong, and C.S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, November 1975.

[29] G. L. Steele, Jr. *Common Lisp, The Language.* Digital Press, second edition, 1990.

[30] C.J. van Rijsbergen. A theoretical basis for the use of cooccurrence data in retrieval. *Journal of Documentation*, 33(2):106–119, June 1977.

[31] P. Willett. Recent trends in hierarchical document clustering: A critical review. *Information Processing & Management*, 24(5):577–597, 1988.

[32] Mark Zimmerman. Texas. Public Domain software, 1989.