

BROWSING IN DIGITAL LIBRARIES: A PHRASE-BASED APPROACH

Craig G. Nevill-Manning*, Ian H. Witten† and Gordon W. Paynter†

* Biochemistry Department,
Stanford University,
Stanford, CA 94305-5307
cneville@stanford.edu

† Department of Computer Science,
University of Waikato,
Hamilton, New Zealand.
{ihw, gwp}@cs.waikato.ac.nz

A key question for digital libraries is this: how should one go about becoming familiar with a digital collection, as opposed to a physical one? Digital collections generally present an appearance which is extremely opaque—a screen, typically a Web page, with no indication of what, or how much, lies beyond: whether a carefully-selected collection or a morass of worthless ephemera; whether half a dozen documents or many millions. At least physical collections occupy physical space, present a physical appearance, and exhibit tangible physical organization. When standing on the threshold of a large library one gains a sense of presence and permanence that reflects the care taken in building and maintaining the collection inside. No-one could confuse it with a dung-heap! Yet in the digital world the difference is not so palpable.

The opacity of digital collections is frustrating. It would seem that precisely because the collection is available digitally it is amenable to automatic indexing, summarization, and visualization techniques, which ought to make browsing particularly easy and convenient. Studies of browsing have shown that it is a rich and fundamental human information behavior, a multifaceted and multidimensional activity (Chang and Rice, 1993). Research is progressing on an arsenal of techniques that head in quite different directions: physical-space metaphors like virtual-reality libraries, navigation metaphors like hypertext, information-space ones like topic clustering and visualization, people-oriented approaches based on formally-defined roles like network librarians or informal ones like intellectual encounter groups, market-oriented schemes like negotiation between agents, ethological ones like foraging, and agricultural ones like harvesting and berry-picking.

Despite the interest and activity in these exciting new possibilities, it is direct, explicit, searching that dominates the digital library scene today. Full-text retrieval makes it possible, in principle, to locate relevant information very efficiently in a huge collection. Whereas the efficiency of indexing and storage are straightforward to evaluate, the task of formulating appropriate

queries and retrieving only relevant documents remains problematic. Retrieval of matching documents involves both *precision* (not returning irrelevant documents) and *recall* (not overlooking relevant documents) (Salton, 1989), and there has been much research on how to maximize precision and recall given a particular query (Harman, 1992–96). In practice, the question posed at the beginning of this article tends to be answered by making a selection of queries more or less haphazardly to gain a feeling for what the collection contains.

This paper suggests a new way of getting to grips with the content of a collection. The idea is to build a hierarchical structure of phrases that appear frequently, and present them interactively to the user. Figure 1 gives an illustration, which will be discussed more fully in due course. Briefly, users can select any word from the lexicon of the collection (the word *index* has been selected in the left-hand column), see which phrases it appears in (center column), select one of them (*indexing and retrieval*) and see the larger phrases in which it appears (right-hand column), and so on (the screen continues to the right). This is reminiscent of the permuted title or keyword-in-context (KWIC) indexes of days gone by. However, there are two crucial differences. First, we identify a *hierarchical* structure of phrases. This greatly reduces the size of the index and allows the user to home in on useful information in logarithmic time. Second, we restrict the phrases to those that occur more than a preset number of times—usually twice or more. This shifts attention from individual items towards the content of the collection as a whole.

Phrase browsing allows users to gain a feeling for the kind of topics that are treated in the collection. As a bottom-up, lexical, approach, it lies at the opposite end of the spectrum to holistic, semantic, methods like document clustering. Both kinds of technique are important, and future solutions will incorporate a variety of different approaches, offering different things to people with different cognitive styles, or just variety for variety's sake.

This paper describes the technique. We have developed an algorithm, called SEQUITUR, that infers a hierarchical structure of phrases from a sequence of discrete symbols. It is extremely efficient and can easily process large chunks of text (up to around 50 Mbyte in less than half an hour of compute time). The phrases are displayed in a browsable form in an interface that we call PHIND—phrase hierarchy for identifying noteworthy documents. The first section describes the phrase-identification algorithm itself. It works incrementally, modifying the grammar as each new symbol of the sequence is received. Two properties are maintained in the grammar: digram uniqueness and rule utility;

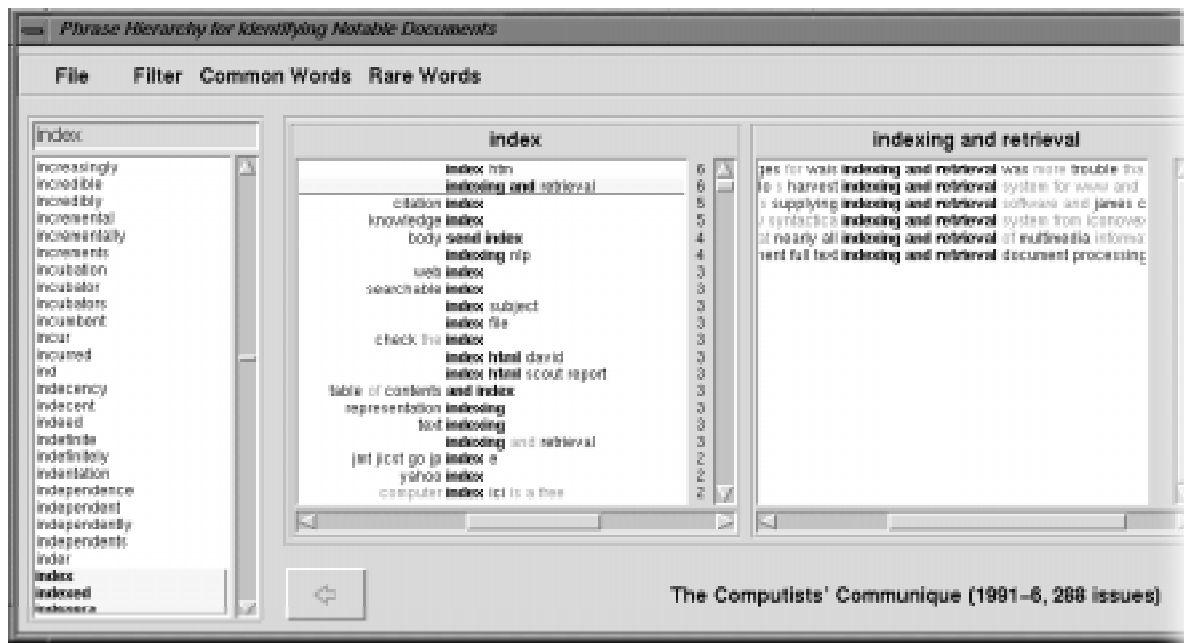


Figure 1 Part of a typical browsing screen for the Computists' Communique collection

these drive the grammar modification process. Following that, we examine two examples of phrase identification in text, one based on characters as tokens and the other on words. Section 3 describes the PHIND interactive browsing interface to document collections, and introduces many issues that arise when displaying SEQUITUR phrases for interactive use. Finally, we discuss the pros and cons of this approach to browsing the content of document collections.

1 The SEQUITUR algorithm

The basic insight of the phrase-finding method is that any phrase which appears more than once can be replaced by a grammatical rule that generates the phrase, and that this process can be continued recursively. The result is a hierarchical representation of the original sequence. It is not a grammar, for the rules are not generalized and are capable of generating only one string. (It does provide a good basis for going on to infer a grammar, but that is beyond the scope of this paper.) A scheme that resembles the one developed here arose from the area of language acquisition (Wolff, 1980). However, the algorithm we describe takes time linear in the length of the input sequence, whereas Wolff's is quadratic. This has allowed us to investigate sequences containing several million tokens. Nevill-Manning (1996) gives a more comprehensive description of the SEQUITUR algorithm and its applications.

SEQUITUR forms a grammar from a sequence based on repeated phrases in it. The key difference from conventional grammatical inference techniques, and from dictionary-based text compression schemes (see e.g. Bell *et al.*, 1990), is that a hierarchical structure is formed from the sequence. Each repetition gives rise to a rule in the grammar, and is replaced by a non-terminal symbol, producing a more concise representation of the sequence. It is this pursuit of brevity that drives the algorithm to form and maintain the grammar, and as a by-product, provide a hierarchical structure for the sequence.

We illustrate the algorithm using characters as phrase structure elements, although when applying the method to browsing we

generally use words. At the left of Figure 2a is a sequence that contains the repeating string *bc*. Note that the sequence is already a grammar—a trivial one with a single rule. To compress it, a new rule $A \rightarrow bc$ is formed, and both occurrences of *bc* are replaced by *A*. The new grammar is shown at the right the Figure.

The sequence in Figure 2b shows how rules can be reused in longer rules. It is formed by concatenating two copies of the sequence in Figure 2a. Since it represents an exact repetition, compression can be achieved by forming the rule $A \rightarrow abcdbc$ to replace both halves of the sequence. Further gains can be made by forming rule $B \rightarrow bc$ to compress rule *A*. This demonstrates the advantage of treating the sequence, rule *S*, as part of the grammar—rules may be formed in rule *A* in an analogous way to rules formed from rule *S*. These rules within rules constitute the grammar's hierarchical structure.

The grammars in Figures 2a and 2b share two properties:

- p_1 : no pair of adjacent symbols appears more than once in the grammar;
- p_2 : every rule is used more than once.

p_1 can be restated as “every digram in the grammar is unique,” and will be referred to as *digram uniqueness*. p_2 ensures that each rule is useful, and will be called *rule utility*. These two constraints exactly characterize the grammars that SEQUITUR generates.

Figure 2c shows what happens when these properties are violated. The first grammar contains two occurrences of *bc*, so p_1 does not hold. This introduces redundancy because *bc* appears twice. In the second grammar, *B* is used only once, so p_2 does not hold. If it were removed, the grammar would become more concise.

The grammars in Figures 2a and 2b are the only ones for which both properties hold for each sequence. However, there is not always a unique grammar with these properties. For example, the sequence in Figure 2d can be represented by both of the grammars on its right, and they both obey p_1 and p_2 . We deem either grammar to be acceptable.

	Sequence	Grammar		Sequence	Grammar
a	$S \rightarrow \text{abcdbc}$	$S \rightarrow \text{aAdA}$ $A \rightarrow \text{bc}$	b	$S \rightarrow \text{abcdbcabcdbc}$	$S \rightarrow \text{AA}$ $A \rightarrow \text{aBdB}$ $B \rightarrow \text{bc}$
c	$S \rightarrow \text{abcdbcabcdbc}$	$S \rightarrow \text{AA}$ $A \rightarrow \text{abcdbc}$ $S \rightarrow \text{CC}$ $A \rightarrow \text{bc}$ $B \rightarrow \text{aA}$ $C \rightarrow \text{BdA}$	d	$S \rightarrow \text{aabaaab}$	$S \rightarrow \text{AaA}$ $A \rightarrow \text{aab}$ $S \rightarrow \text{AbAab}$ $A \rightarrow \text{aa}$

Figure 2 Example sequences and grammars that reproduce them

- (a) a sequence with one repetition
- (b) a sequence with a nested repetition
- (c) two grammars that violate the two constraints
- (d) two different grammars for the same sequence that obey the constraints

SEQUITUR's operation consists of ensuring that both properties hold. When describing the algorithm, the properties act as *constraints*. The algorithm operates by enforcing the constraints on a grammar: when the digram uniqueness constraint is violated, a new rule is formed, and when the rule utility constraint is violated, the useless rule is deleted. The next two sections describe how this is performed.

Digram uniqueness

When a new symbol is observed, it is appended to rule S , the top-level rule. The last two symbols of rule S —the new symbol and its predecessor—form a new digram. If it occurs elsewhere in the grammar, the first constraint has been violated. To restore it, a new rule is formed with the digram on the right-hand side, headed by a new non-terminal. The two original digrams are replaced by this non-terminal. However, the appearance of a duplicate digram does not always result in a new rule. If the new digram appears as the right-hand side of an existing rule, then no new rule need be created: the digram is replaced by the non-terminal that heads the existing rule.

The hierarchy is formed and maintained by an iterative process. Changes ripple through the grammar, forming and matching longer rules higher in the hierarchy.

Rule utility

So far, it seems that the right-hand side of any rule in the grammar will only ever be two symbols long. However, longer rules are formed by the effect of the rule utility constraint, which ensures that every rule is used more than once. When a new symbol is appended to the top-level rule, the new digram that it creates may begin with a non-terminal symbol—which must of course be defined elsewhere in the grammar. Suppose this new digram appears only once in the rest of the grammar. Then a new rule will be defined to replace the digram. The fact that the non-terminal symbol is only used in this new rule violates the rule utility constraint. Therefore the non-terminal is removed from the grammar, its definition being incorporated into the new rule that has just been formed.

This is the mechanism for forming long rules: form a short rule temporarily, and if subsequent symbols continue the match, allow a new rule to supersede the shorter one.

2 Phrase identification in text

SEQUITUR can operate on any sequence of tokens. To convey a feeling for what it produces, we give two examples. The first uses characters as tokens, and shows how the algorithm successfully identifies some lexical features. The second uses words as tokens, and shows how phrases are discovered by the algorithm.

Character-based hierarchies

Figure 3 shows parts of three hierarchies inferred from the text of the Bible in English, French, and German. The hierarchies are formed without any knowledge of the preferred structure of words and phrases, but nevertheless capture many meaningful regularities. In Figure 3a, the word *beginning* is split into *begin* and *ning*—a root word and a suffix. Many words and word groups appear as distinct parts in the hierarchy (spaces have been made explicit by replacing them with bullets). The same algorithm produces the French version in Figure 3b, where *commencement* is split in an analogous way to *beginning*—into the root *commence* and the suffix *ment*. Again, words such as *Au*, *Dieu* and *cieux* are distinct units in the hierarchy. The German version in Figure 3c correctly identifies all words, as well as the phrase *die Himmel und die*. In fact, the hierarchy for *the heaven and the* in Figure 3a bears some similarity to the German equivalent.

Each version of the Bible contains between four and five million characters. Forming a hierarchy from such a sequence is accomplished in about two minutes.

Word-based phrases

In a second experiment, SEQUITUR was invoked on a large body of technical reports, part of the 1.9 Gbyte corpus comprising the Computer Science Technical Report collection of the New Zealand Digital Library (Witten *et al.*, 1996). The reports were presented as a sequence of words, and all words were mapped, somewhat arbitrarily, to lower case before processing. A 28 Mb sample was chosen, which included 500 technical reports from nine sites. Table 1 shows the size of the sample and the resulting grammar (which took 12 minutes to produce on an aging Sun Sparc).

In any hierarchy produced by SEQUITUR, the rule headed by S , the start symbol, expands to reproduce the entire sequence. This initial rule has rather a different character to the others. Essentially, all other rules express regularities in the original sequence, in that their contents must occur at least twice. The top-level rule receives all the leftovers; the unique sequences that do not recur. In the grammar produced for the sample of Table 1, the initial rule contained 1.5 million symbols—about one third of the number in the original sequence. There were 280,000 other

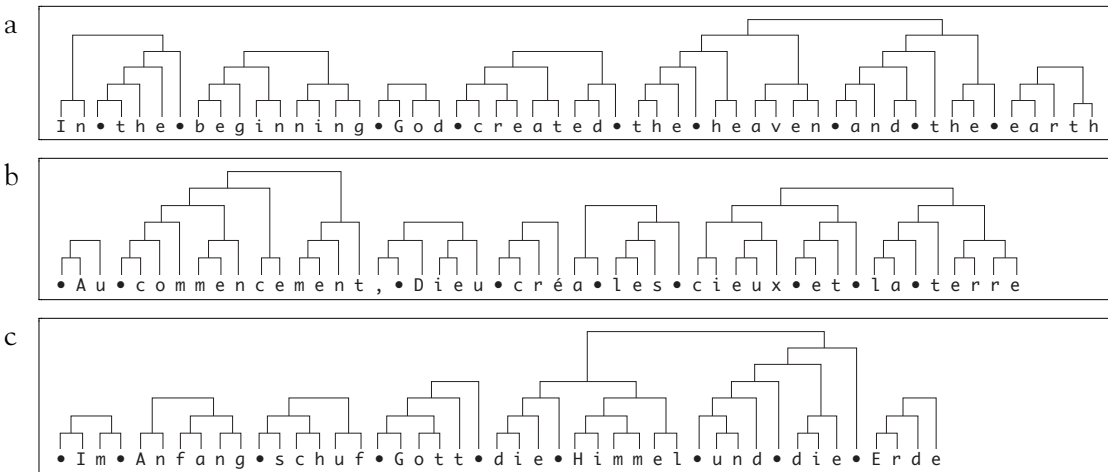


Figure 3 Hierarchies for Genesis 1:1 in (a) English, (b) French, and (c) German

rules, having an average of only 2.6 symbols (terminals and non-terminals) in their right-hand side. In total, the grammar contained 2.3 million symbols, or about half of the original number of words.

Figure 4 shows some of the phrases produced, namely, the ones that contain *grammar*. Although this string occurs 5,000 times in the original sample, it appears in only 60 rules. The first column of Figure 4 lists several of these, recast into phrases. For example, the first entry corresponds to the rule

$A \rightarrow \text{context-free grammars}$

Note that this rule is included because it contains the string *grammar*, even though it appears in inflected form: the first column of Figure 4 was obtained by searching the entire grammar for the character string *grammar*, not for the lexical token *grammar*.

Some of the rules contain non-terminals; these have been expanded into the corresponding sequence of words for display. Note that in this example, the digram uniqueness constraint seems to have been violated because the digram *context-free grammar* appears twice in the first column, in the second and last entries. However, this is not so. In fact the last entry corresponds to the rules

$B \rightarrow C \text{ grammar}$
 $C \rightarrow a \text{ context-free}$

This sub-optimal phrase identification is an unfortunate, but inevitable, consequence of the fast algorithm we use: in fact, the problem of finding a dictionary that produces the smallest phrase-structure representation is NP-complete (Storer and Szymanski, 1982).

The first column gives a lexical picture of how the word *grammar* is used. In the second column, we focus on the first rule—rule *A* above—and list those places in the grammar where it appears. The entries flanked by ellipses indicate extracts from rule *S*, the top-level rule, and so rule *A* occurs in only two rules:

$D \rightarrow \text{probabilistic } A$
 $S \rightarrow \dots E A \dots F A \dots$

where

$E \rightarrow \text{the set of}$
 $F \rightarrow \text{described by}$

Column two focuses attention on the phrase *probabilistic context-free grammars* as the only context in which *context-free grammars* appears more than once. Expanding this phrase leads to the third column, which introduces another rule,

$G \rightarrow \text{probabilistic context-free grammars (PCFGs)}$

as well as two further extracts from the top-level rule. Note that these two extracts were not reported above as containing *context-free grammars* since the phrase occurred in this larger context—this illustrates how SEQUITUR's phrases focus attention on how words are used and avoid cluttering up the display with correct but uninteresting uses.

The new rule *G* occurs in only two places, both in the top-level rule (fourth column of Figure 4). However, it also leads to an interesting discovery: that *PCFGs* is being used as a synonym for *probabilistic context-free grammars*. The implication that this acronym is used in place of the phrase is potentially extremely useful: the user may decide to perform another search on the abbreviation, because it is likely that once it has been defined, the acronym alone will be used. It is interesting to consider how this information would have been obscured in a different method of presentation—it is hard to imagine such a discovery taking place in the absence of a phrase-browsing technique, because the co-occurrence of the phrase and the abbreviation usually has no special significance.

3 Interactive browsing

So far we have seen an explanation of how the phrase hierarchy is created, and a description of how it might facilitate general browsing around a collection of information. The descriptions above have considered the phrase hierarchy in general. However, many detailed decisions must be made in order to operationalize this general idea. In this section we describe PHIND, an interactive system that presents the phrase hierarchy in a way that facilitates browsing.

The screen display in Figure 1 shows a hierarchy based (for variety's sake) on a different corpus, the *Computists' Communique* (www.computists.com). This is an on-line AI research news magazine, operating since 1991, which includes grant and funding opportunities, industry news, Internet and Web information, on-line resources, research discussion lists, software offers, software development resources, and career and entrepreneurial tips.

Sample	500 technical reports from nine FTP sites 4.5 million words, 50,000 word vocabulary
Grammar	280,000 rules of average size (rhs) 2.6 symbols 1.5 million symbols in the top-level rule

Table 1 Size of the Computer Science Technical Report sample and resulting grammar

rules containing <i>grammar</i>	rules containing <i>context-free grammars</i>	rules containing <i>probabilistic context-free grammars</i>	rules containing <i>probabilistic context-free grammars (PCFGs)</i>
context-free ~s context-free ~ ~ procedure tree ~s functional ~ systemic ~ systemic ~s logic ~s attribute ~s with a corpus based ~ a context-free ~ ...	probabilistic ~ ...the set of ~... ...described by ~...	~ (PCFGs) ...recent interest in ~ for language modeling... ...language model used in particular ~ will have to deal with this problem...	...researchers have become interested in ~ for language modeling the fact that a PCFG... ...brief aside let us come back to the ~ that we referred to...

Figure 4 Hierarchy of phrases for searching a large text base

When used with PHIND, SEQUITUR operates in word mode, using complete words as tokens. Punctuation is removed and words are folded to lower case. The vocabulary of the collection is displayed alphabetically in the leftmost column of Figure 1. Users can select a word (either with the mouse or by typing), and all phrases in which it appears are displayed along with the number of times each phrase occurs. In Figure 1 the user has selected *index* from the vocabulary and the phrases it appears in are listed in the next column to the right. For example, *index htm* appears six times. Note that this particular phrase appears as an artifact of word parsing: it emanates from the filename *index.htm*—as of course does *index.html*, further down the list. It is encouraging that these junk entries consume far less space in the list than they would in a conventional query for the term *index*.

Each phrase can be selected and expanded in turn. The user has selected the phrase *indexing and retrieval*, which also appears six times in the corpus. In this particular case, each of these six phrases occurs exactly once and cannot be expanded any further—in fact they are all flanked by ellipses that would be revealed by scrolling the third column horizontally.

In general, the user can traverse the grammar, extending and hence specializing the query term. Every word is the root of a tree structure whose leaves are the occurrences of that word in the top-level rule. Occurrences in other rules are internal nodes corresponding to phrases that contain the word. Those phrases are themselves used elsewhere in the grammar, either in the top-level rule or in other rules for longer phrases. It is possible to stop at any internal node and use that phrase as a query term, or continue following the tree to a leaf and retrieve the corresponding document (although the final step of document retrieval is not yet implemented in PHIND).

We have incorporated a number of additional interactive facilities into the browser: stemming, and mechanisms for dealing with rare words and common words. These greatly enhance the interactive interface, and we discuss them in turn.

Stemming

PHIND can be used to browse the grammar exactly as it was generated from the corpus. However, some properties of SEQUITUR's grammars, and of word usage in general, mean that strict adherence to the phrase structure is often counter-productive. For example, SEQUITUR treats different words as completely different symbols even though they may be closely related lexically. PHIND overcomes this problem in a rather simplistic manner by treating each selection from the vocabulary list as the stem of a word.

In Figure 1 the user has selected *index*, but PHIND has searched for phrases including *indexed*, *indexers*, *indexes*, and *indexing* as well.

Rare words

Other features help the user explore the document collection conveniently. One concerns *hapax legomena*. Generally, between 30% and 50% of the words in the vocabulary of any collection appear only once. These words can never generate a phrase hierarchy, and since PHIND is a tool for gaining a feel for the general content of a corpus they are unlikely to be of interest to the user—even though they would be highly significant if used in any particular query. By default, PHIND labels any word that appears less than a small, preset number of times a *rare word*. Rare words are not included in the vocabulary list and are printed in red whenever they appear in the other columns. The user can change the threshold for rare word detection so that there are no rare words, or so that words that occur fewer than a certain number of times are considered rare; this can be done interactively.

In Figure 1, the only rare word visible is *syntactica*, which appears on the fourth line of the *indexing and retrieval* column: in fact it appears in red although this is not apparent in the Figure. Incidentally this word is a brand name in "Syntactica indexing and retrieval systems"; a possibly interesting feature of the collection that we would surely never have noticed without such a browsing tool.

Common words

Words that occur very frequently can also obscure information. Words like *a* and *the* cause problems because SEQUITUR often uses them to form rules, but as far as the user is concerned they add little meaning to the phrase. Nobody really wants to know that the most common use of the word *index* is in the phrase *the index*. By default, PHIND labels as *common words* the one hundred most frequently occurring words in the collection and displays them in gray—many examples are visible in the second and third columns of Figure 1. Users can set the threshold for common words interactively, and can set it to zero to disable this facility.

A word that is neither common nor rare is an *interesting word*. An *interesting phrase* is a one that contains an interesting word other than the term (or terms) that were used to locate the phrase. SEQUITUR weeds out profitless phrases like *indexing and* by only displaying interesting phrases. When the user performs a search for *index* the phrase *indexing and* is not returned because *and* is not an interesting word. Conversely, the phrase *knowledge index* is displayed because *knowledge* is an interesting word. Phrases that are not interesting are expanded until they subsume at least one further interesting word. For example, when *indexing and* is expanded it generates several interesting phrases, including *indexing and retrieval* and *automatic indexing and fact extraction from*.

Relationship between SEQUITUR and PHIND rulesets

Forcing PHIND to look for interesting phrases has a radical effect on the results of most searches. SEQUITUR creates its grammar by making digrams out of symbols that occur together, and therefore the most frequently appearing symbols—our “common words”—tend to form the most frequently used phrases. The most common phrases are the most likely to be uninteresting, in which case they become re-expanded. As a result, the phrases that PHIND displays are often several words long, whereas short phrases like *indexing and* occur frequently in SEQUITUR’s output. Moreover, there are many phrases that are expanded right up to their appearance in the top-level rule. Consequently, PHIND’s hierarchies are shallower than those of SEQUITUR.

In the example of Figure 1, around 60% of appearances of the word *index* in the SEQUITUR grammar are in rules other than the top-level one. However, only 25% of these appearances are accounted for by PHIND rules. The remaining 35% of appearances in SEQUITUR’s rules are in spurious rules that do not add anything meaningful because they differ from another rule only by the inclusion of common words.

In general, we have observed that increasing the size of the collection increases the depth of the PHIND hierarchy, the number of phrases returned by each query, and the average length of the phrases, but none noticeably more than the others.

4 Discussion

Pros

The intention of this work is to give the user a good idea of the subject matter of the text in a large collection, and present it in manageable chunks determined by the branching factor of each rule. This is especially evident at the very top level, where, in the example of Figure 4, the list of rules involving the word *grammar* provides a plausible taxonomy of concepts involving grammars.

The hierarchy produced by our technique contrasts with methods such as keyword-in-context (KWIC) displays, where all

occurrences of a search term are displayed along with the surrounding context. KWIC indexes do not scale, because the amount displayed for any given search term depends linearly on the size of the collection: this is presumably why they have fallen out of use. In our hierarchical approach, the amount displayed probably grows logarithmically with collection size (although we have not yet been able to establish this result theoretically).

Here are two examples that underscore this point. First, consider the discovery of the acronym *PCFGs* that was made when analyzing Figure 4. The fact that this acronym co-occurs with the phrase several times would not be visually apparent in a KWIC display, and would not receive much prominence. Its embodiment in a rule, however, ensures that the SEQUITUR-based method puts it at the top of the list of occurrences. Second, the word *index* occurs 487 times in the collection used to create Figure 1, in 200 separate news articles. A KWIC display of each of these occurrences would be less than useful. The display would have been cluttered up with many junk occurrences of *index htm* and *index html*, and the key phrase *indexing and retrieval* would be unlikely to be spotted, as would the other key phrases *citation index*, *knowledge index*, *indexing nlp*, *web index*, and so on.

The technique is potentially very efficient. Again returning to Figure 4, once the word *grammar* has been found in the lexicon—which can be accomplished quickly using a trie data structure—SEQUITUR’s internal pointers between different instances of the same symbol can be used to retrieve the rules in time linear in the number of phrases. When a rule is picked, the places in which the non-terminal appears can be accessed in the same way. Once the hierarchy has been formed, all pointers necessary to traverse it are in place, and there is no need for any search.

In practice, the ability to adjust thresholds for rare words and common words interactively is very useful when trying to get to know the content of a collection.

Cons

Although SEQUITUR’s phrases give a good general idea of the structure of the grammar and the frequency of the phrases, there are several situations where phrases are given too much importance—or too little. Phrases receive artificially high frequency when an author quotes sections of a paper in its abstract, or—worse still—when the title of a paper is repeated in the header of every page. It is usually obvious when this has happened because phrases become very long and occur at the same level of the grammar. We have encountered similar problems with the headers of news articles, and with references and bibliographies.

SEQUITUR’s grammars often result in phrase boundary conflicts. For example, the phrase *indexing and retrieval* occurs twice in the first column of Figure 1. (The first occurrence is made up of the symbols *indexing and* and *retrieval*, and the second of *indexing and retrieval*.) Due to these conflicts, important phrases may be overlooked because they are not listed as prominently as they should be.

Another problem results from our definition of common words as the most frequently-occurring terms in the collection. Sometimes words are inappropriately classed as “common.” For example, in one subcollection of the Computer Science Technical Report library, we found that *neural* was among the one hundred most frequently used words.

Finally, although the technique is very efficient, it does require a lot of main memory. Forming the phrases in the first place is inherently memory-intensive: it grows linearly with the size of the

grammar, and for the Table 1 example is about 60 Mbyte—twice the size of the corpus. Once the hierarchical grammar is in place, the PHIND interface could use direct disk access, although our present implementation does not and therefore makes considerable main-memory demands on the host computer.

Conclusion

The thrust of this research is to build systems that let users become familiar with the content of a digital library by browsing a hierarchical structure of phrases that are repeated frequently within the collection. Despite our syntactic approach to phrase identification, the structures that are obtained in practice frequently correspond to plausible conceptual hierarchies. This permits large corpora of text to be browsed efficiently, with access to a particular document requiring a number of steps that varies with the logarithm of the size of the corpus.

The method shows promise for collections of up to 50 Mbyte, but still poses significant practical problems before it can be adopted on a wider scale. We plan to investigate ways to scale the hierarchical inference by building multiple hierarchies and merging them. On the browsing side, we are developing a disk-rather than memory-based system that can run efficiently on client-class machines.

We believe that in the context of large information bases such as the New Zealand Digital Library, this interface will obviate the “query and hope” approach to browsing, and allow users to develop an intuition that would otherwise be very difficult to acquire.

References

- Bell, T.C., Cleary, J.G. and Witten, I.H. (1990) *Text compression*. Prentice Hall, Englewood Cliffs, New Jersey.
- Chang, S.J. and Rice, R.E. (1993) “Browsing: a multidimensional framework,” *Annual Review of Information Science and Technology*, 28, 231–276.
- Harman, D.K.E. (1992–96) “Proc. TREC Text Retrieval Conference,” Gaithersburg, MD: National Institute of Standards Special Publication, 500-207, 500-215, 500-225, 500-236.
- Nevill-Manning, C.G. (1996) “Inferring sequential structure,” D.Phil. thesis, Computer Science Department, University of Waikato, New Zealand.
- Salton, G. (1989) *Automatic Text Processing: the transformation, analysis and retrieval of information by computer*. Reading, Mass.: Addison Wesley.
- Storer, J.A. and Szymanski, T.G. (1982) “Data compression via textual substitution.” *J Association for Computing Machinery* 29(4), 928–951.
- Witten, I.H., Nevill-Manning, C.G., and Cunningham, S.J. (1996) “Building a digital library for computer science research: technical issues,” *Proc. Australasian Computer Science Conference*, Melbourne, Australia, 534-542.
- Wolff, J.G. (1980) “Language acquisition and the discovery of phrase structure,” *Language and Speech*, 23(3), 255–269.