Chapter 7
Text Operations

with Nivio Ziviani

7.1 Introduction

As discussed in Chapter 2, not all words are equally significant for representing the semantics of a document. In written language, some words carry more *meaning* than others. Usually, *noun* words (or groups of noun words) are the ones which are most representative of a document content. Therefore, it is usually considered worthwhile to preprocess the text of the documents in the collection to determine the terms to be used as *index terms*. During this preprocessing phase other useful text operations can be performed such as elimination of stopwords, stemming (reduction of a word to its grammatical root), the building of a thesaurus, and compression. Such text operations are discussed in this chapter.

We already know that representing documents by sets of index terms leads to a rather imprecise representation of the semantics of the documents in the collection. For instance, a term like '*the*' has no meaning whatsoever by itself and might lead to the retrieval of various documents which are unrelated to the present user query. We say that using the set of all words in a collection to index its documents generates too much *noise* for the retrieval task. One way to reduce this noise is to reduce the set of words which can be used to refer to (i.e., to index) documents. Thus, the preprocessing of the documents in the collection might be viewed simply as a process of controlling the size of the vocabulary (i.e., the number of distinct words used as an index terms). It is expected that the use of a controlled vocabulary leads to an improvement in retrieval performance.

While controlling the size of the vocabulary is a common technique with commercial systems, it does introduce an additional step in the indexing process which is frequently not easily perceived by the users. As a result, a common user might be surprised with some of the documents retrieved and with the absence of other documents which he expected to see. For instance, he might remember that a certain document contains the string '*the house of the lord*' and notice that such a document is not present among the top 20 documents retrieved in

response to his query request (because the controlled vocabulary contains neither 'the' nor 'of'). Thus, it should be clear that, despite a potential improvement in retrieval performance, text transformations done at preprocessing time might make it more difficult for the user to interpret the retrieval task. In recognition of this problem, some search engines in the Web are giving up text operations entirely and simply indexing all the words in the text. The idea is that, despite a more noisy index, the retrieval task is simpler (it can be interpreted as a full text search) and more intuitive to a common user.

Besides document preprocessing, other types of operations on documents can also be attempted with the aim of improving retrieval performance. Among these we distinguish the construction of a thesaurus representing conceptual term relationships and the clustering of related documents. Thesauri are also covered in this chapter. The discussion on document clustering is covered in Chapter 5 because it is an operation which might depend on the current user query.

Text normalization and the building of a thesaurus are strategies aimed at improving the precision of the documents retrieved. However, in the current world of very large digital libraries, improving the efficiency (in terms of time) of the retrieval process has also become quite critical. In fact, Web search engines are currently more concerned with reducing query response time than with improving precision and recall figures. The reason is that they depend on processing a high number of queries per unit of time for economic survival. To reduce query response time, one might consider the utilization of text compression as a promising alternative.

A good compression algorithm is able to reduce the text to 30–35% of its original size. Thus, compressed text requires less storage space and takes less time to be transmitted over a communication link. The main disadvantage is the time spent compressing and decompressing the text. Until recently, it was generally understood that compression does not provide substantial gains in processing time because the extra time spent compressing/decompressing text would offset any gains in operating with compressed data. Further, the use of compression makes the overall design and implementation of the information system more complex. However, modern compression techniques are slowly changing this understanding towards a more favorable view of the adoption of compression techniques. By modern compression techniques we mean good compression and decompression speeds, fast random access without the need to decode the compressed text from the beginning, direct searching on the compressed text without decompressing it, among others.

Besides compression, another operation on text which is becoming more and more important is *encryption*. In fact, due to the fast popularization of services in the Web (including all types of electronic commerce), key (and old) questions regarding security and privacy have surfaced again. More than ever before, impersonation and unauthorized access might result in great prejudice and financial damage to people and organizations. The solution to these problems is not simple but can benefit from the operation of encrypting text. Discussing encrypted text is beyond the scope of this book but an objective and brief introduction to the topic can be found in [498].

In this chapter, we first discuss five preprocessing text operations including thesauri. Following that, we very briefly summarize the problem of document clustering (which is discussed in detail in Chapter 5). Finally, a thorough discussion on the issue of text compression, its modern variations, and its main implications is provided.

7.2 Document Preprocessing

Document preprocessing is a procedure which can be divided mainly into five text operations (or transformations):

(1) Lexical analysis of the text with the objective of treating digits, hyphens, punctuation marks, and the case of letters.

(2) Elimination of stopwords with the objective of filtering out words with very low discrimination values for retrieval purposes.

(3) Stemming of the remaining words with the objective of removing affixes (i.e., prefixes and suffixes) and allowing the retrieval of documents containing syntactic variations of query terms (e.g., connect, connecting, connected, etc).

(4) Selection of index terms to determine which words/stems (or groups of words) will be used as an indexing elements. Usually, the decision on whether a particular word will be used as index term is related to the syntactic nature of the word. In fact, noun words frequently carry more semantics than adjectives, adverbs, and verbs.

(5) Construction of term categorization structures such as a thesaurus, or extraction of structure directly represented in the text, for allowing the expansion of the original query with related terms (a usually useful procedure).

In the following, each of these phases is discussed in detail. But, before proceeding, let us take a look at the logical view of the documents which results after each of the above phases is completed. Figure 1.2 is repeated here for convenience as Figure 7.1. As already discussed, by aggregating the preprocessing phases, we are able to move the logical view of the documents (adopted by the system) from that of a full text to that of a set of high level indexing terms.

7.2.1 Lexical Analysis of the Text

Lexical analysis is the process of converting a stream of characters (the text of the documents) into a stream of words (the candidate words to be adopted as index terms). Thus, one of the major objectives of the lexical analysis phase is the identification of the words in the text. At first glance, all that seems to be involved is the recognition of spaces as word separators (in which case, multiple

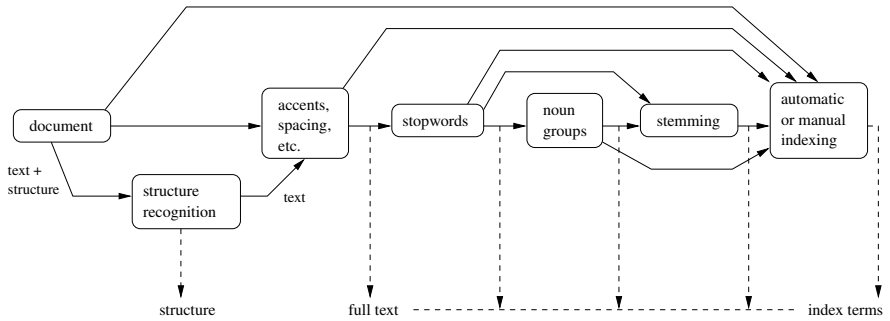

Figure 7.1 Logical view of a document throughout the various phases of text preprocessing.

spaces are reduced to one space). However, there is more to it than this. For instance, the following four particular cases have to be considered with care [261]: digits, hyphens, punctuation marks, and the case of the letters (lower and upper case).

Numbers are usually not good index terms because, without a surrounding context, they are inherently vague. For instance, consider that a user is interested in documents about the number of deaths due to car accidents between the years 1910 and 1989. Such a request could be specified as the set of index terms {deaths, car, accidents, years, 1910, 1989}. However, the presence of the numbers 1910 and 1989 in the query could lead to the retrieval, for instance, of a variety of documents which have a reference to either of these two years. The problem is that numbers by themselves are just too vague. Thus, in general it is wise to disregard numbers as index terms. However, we have also to consider that digits might appear mixed within a word. For instance, '510B.C.' is a clearly important index term. In this case, it is not clear what rule should be applied. Furthermore, a sequence of 16 digits identifying a credit card number might be highly relevant in a given context and, in this case, should be considered as an index term. A preliminary approach for treating digits in the text might be to remove all words containing sequences of digits unless specified otherwise (through regular expressions). Further, an advanced lexical analysis procedure might perform some date and number normalization to unify formats.

Hyphens pose another difficult decision to the lexical analyzer. Breaking up hyphenated words might be useful due to inconsistency of usage. For instance, this allows treating 'state-of-the-art' and 'state of the art' identically. However, there are words which include hyphens as an integral part. For instance, gilt-edge, B-49, etc. Again, the most suitable procedure seems to adopt a general rule and specify the exceptions on a case by case basis.

Normally, punctuation marks are removed entirely in the process of lexical analysis. While some punctuation marks are an integral part of the word (for

instance, '510B.C.'), removing them does not seem to have an impact in retrieval performance because the risk of misinterpretation in this case is minimal. In fact, if the user specifies '510B.C' in his query, removal of the dot both in the query term and in the documents will not affect retrieval. However, very particular scenarios might again require the preparation of a list of exceptions. For instance, if a portion of a program code appears in the text, it might be wise to distinguish between the variables 'x.id' and 'xid.' In this case, the dot mark should not be removed.

The case of letters is usually not important for the identification of index terms. As a result, the lexical analyzer normally converts all the text to either lower or upper case. However, once more, very particular scenarios might require the distinction to be made. For instance, when looking for documents which describe details about the command language of a Unix-like operating system, the user might explicitly desire the non-conversion of upper cases because this is the convention in the operating system. Further, part of the semantics might be lost due to case conversion. For instance, the words *Bank* and *bank* have different meanings — a fact common to many other pairs of words.

As pointed out by Fox [261], all these text operations can be implemented without difficulty. However, careful thought should be given to each one of them because they might have a profound impact at document retrieval time. This is particularly worrisome in those situations in which the user finds it difficult to understand what the indexing strategy is doing. Unfortunately, there is no clear solution to this problem. As already mentioned, some Web search engines are opting for avoiding text operations altogether because this simplifies the interpretation the user has of the retrieval task. Whether this strategy will be the one of choice in the long term remains to be seen.

7.2.2 Elimination of Stopwords

As discussed in Chapter 2, words which are too frequent among the documents in the collection are not good discriminators. In fact, a word which occurs in 80% of the documents in the collection is useless for purposes of retrieval. Such words are frequently referred to as *stopwords* and are normally filtered out as potential index terms. Articles, prepositions, and conjunctions are natural candidates for a list of stopwords.

Elimination of stopwords has an additional important benefit. It reduces the size of the indexing structure considerably. In fact, it is typical to obtain a compression in the size of the indexing structure (for instance, in the size of an inverted list, see Chapter 8) of 40% or more solely with the elimination of stopwords.

Since stopword elimination also provides for compression of the indexing structure, the list of stopwords might be extended to include words other than articles, prepositions, and conjunctions. For instance, some verbs, adverbs, and adjectives could be treated as stopwords. In [273], a list of 425 stopwords is illustrated. Programs in C for lexical analysis are also provided.

Despite these benefits, elimination of stopwords might reduce recall. For instance, consider a user who is looking for documents containing the phrase '*to be or not to be.*' Elimination of stopwords might leave only the term *be* making it almost impossible to properly recognize the documents which contain the phrase specified. This is one additional reason for the adoption of a full text index (i.e., insert all words in the collection into the inverted file) by some Web search engines.

7.2.3 Stemming

Frequently, the user specifies a word in a query but only a variant of this word is present in a relevant document. Plurals, gerund forms, and past tense suffixes are examples of syntactical variations which prevent a perfect match between a query word and a respective document word. This problem can be partially overcome with the substitution of the words by their respective stems.

A *stem* is the portion of a word which is left after the removal of its affixes (i.e., prefixes and suffixes). A typical example of a stem is the word *connect* which is the stem for the variants *connected, connecting, connection,* and *connections*. Stems are thought to be useful for improving retrieval performance because they reduce variants of a same root word to a common concept. Furthermore, stemming has the secondary effect of reducing the size of the indexing structure because the number of distinct index terms is reduced.

While the argument supporting stemming seems sensible, there is controversy in the literature about the benefits of stemming for retrieval performance. In fact, different studies lead to rather conflicting conclusions. Frakes [273] compares eight distinct studies on the potential benefits of stemming. While he favors the usage of stemming, the results of the eight experimental studies he investigated do not allow us to reach a satisfactory conclusion. As a result of these doubts, many Web search engines do not adopt any stemming algorithm whatsoever.

Frakes distinguishes four types of stemming strategies: affix removal, table lookup, successor variety, and n-grams. Table lookup consists simply of looking for the stem of a word in a table. It is a simple procedure but one which is dependent on data on stems for the whole language. Since such data is not readily available and might require considerable storage space, this type of stemming algorithm might not be practical. Successor variety stemming is based on the determination of morpheme boundaries, uses knowledge from structural linguistics, and is more complex than affix removal stemming algorithms. N-grams stemming is based on the identification of digrams and trigrams and is more a term clustering procedure than a stemming one. Affix removal stemming is intuitive, simple, and can be implemented efficiently. Thus, in the remainder of this section we concentrate our discussion on algorithms for affix removal stemming only.

In affix removal, the most important part is suffix removal because most variants of a word are generated by the introduction of suffixes (instead of pre-

fixes). While there are three or four well known suffix removal algorithms, the most popular one is that by Porter because of its simplicity and elegance. Despite been simpler, the Porter algorithm yields results comparable to those of the more sophisticated algorithms.

The Porter algorithm uses a suffix list for suffix stripping. The idea is to apply a series of rules to the suffixes of the words in the text. For instance, the rule

$$s \longrightarrow \phi \tag{7.1}$$

is used to convert plural forms into their respective singular forms by substituting the letter s by nil. Notice that to identify the suffix we must examine the last letters in the word. Furthermore, we look for the longest sequence of letters which matches the left hand side in a set of rules. Thus, application of the two following rules

$$\begin{aligned} sses &\longrightarrow ss \\ s &\longrightarrow \phi \end{aligned} \tag{7.2}$$

to the word *stresses* yields the stem *stress* instead of the stem *stresse*. By separating such rules into five distinct phases, the Porter algorithm is able to provide effective stemming while running fast. A detailed description of the Porter algorithm can be found in Appendix B.

7.2.4 Index Terms Selection

If a full text representation of the text is adopted then all words in the text are used as index terms. The alternative is to adopt a more abstract view in which not all words are used as index terms. This implies that the set of terms used as indices must be selected. In the area of bibliographic sciences, such a selection of index terms is usually done by a specialist. An alternative approach is to select terms for index terms automatically.

Distinct automatic approaches for selecting index terms can be used. A good approach is the identification of noun groups (as done in the Inquery system [120]) which we now discuss.

A sentence in natural language text is usually composed of nouns, pronouns, articles, verbs, adjectives, adverbs, and connectives. While the words in each grammatical class are used with a particular purpose, it can be argued that most of the semantics is carried by the noun words. Thus, an intuitively promising strategy for selecting index terms automatically is to use the nouns in the text. This can be done through the systematic elimination of verbs, adjectives, adverbs, connectives, articles, and pronouns.

Since it is common to combine two or three nouns in a single component (e.g., *computer science*), it makes sense to cluster nouns which appear nearby in the text into a single indexing component (or concept). Thus, instead of simply