Word Segmentation and Recognition for Web Document Framework

Chi-Hung Chi, Chen Ding, Andrew Lim
School of Computing
National University of Singapore
Lower Kent Ridge Road, Singapore 119260

ABSTRACT

It is observed that a better approach to Web information understanding is to base on its document framework, which is mainly consisted of (i) the title and the URL name of the page, (ii) the titles and the URL names of the Web pages that it points to, (iii) the alternative information source for the embedded Web objects, and (iv) its linkage to other Web pages of the same document. Investigation reveals that a high percentage of words inside the document framework are "compound words" which cannot be understood by ordinary dictionaries. They might be abbreviations or acronyms, or concatenations of several (partial) words. To recover the content hierarchy of Web documents, we propose a new word segmentation and recognition mechanism to understand the information derived from the Web document framework. A maximal bi-directional matching algorithm with heuristic rules is used to resolve ambiguous segmentation and meaning in compound words. An adaptive training process is further employed to build a dictionary of recognisable abbreviations and acronyms. Empirical results show that over 75% of the compound words found in the Web document framework can be understood by our mechanism. With the training process, the success rate of recognising compound words can be increased to about 90%.

1. INTRODUCTION

Table of content (TOC) is always an important source of information for traditional document understanding. It gives readers the outline of a document as well as the hierarchical linkage among its chapters or sections. This is true even for short articles where the headings and sub-headings can still reflect an abstraction about its content. In a Web document, there exists a new source of information, called the **document framework**, which contains higher information entropy than the possible "table of content" in a traditional document. This framework is the skeleton of the Web document structure. A vertex in the graph is a component Web page with attributes of its title and URL name of the page together with alternate information source for all its embedded objects. It can be a section or sub-section of the document. An edge in the graph shows the referral relationship of one Web page to the next one. It gives the relationships among sections (or Web pages) of a Web document. Although most of the framework information might be invisible to the Web surfers, they are more important to Web document understanding than the traditional TOC does because of the following reasons.

There are usually more "headings" and "sub-headings" (in the form of page titles and URLs) in a Web document than in a traditional one. A common practice in Web design is to fit what can be displayed on the monitor screen into a Web page. This often results in the number of pages in a Web-based document being far more than its number of sections and sub-sections in a traditional document. Each of these pages needs an unique description for its title and URL address name.

The information entropy of the document framework of a Web page is usually much higher than that of the TOC of a traditional document. While a section heading is usually consisted of only a few words, the document framework of a Web page is made up of at least three components: title and URL name of the Web page, titles and URL names of the Web pages it is hyperlinked to, and alternate information for all embedded objects inside the page. As a result, its information content is richer than that of TOC.

The size of Web pages further increases the importance of its framework in Web information understanding. We conducted an experiment to find the size ranges of Web pages. For each Web page under study, its size was measured as the word count of the page, excluding the image information and the structural tags. It is found that for the 1000 Web pages used in the experiment, about 60% of them have size between 100 to 1000 words. This page size range is quite small and is not as much as that in traditional documents. Consequently, understanding the Web document framework becomes more important.

Diagnose and Repair Your Web Site
http://www8.zdnet.com/pcmag/features/Webdiag/_open.htm
1.1 You expend a lot of effort creating the perfect Web site. Shouldn't you make sure it actually works?
http://www8.zdnet.com/pcmag/features/Webdiag/_intro.htm
1.2 What They Do
http://www8.zdnet.com/pcmag/features/Webdiag/_intro1.htm
1.3 So Which One
http://www8.zdnet.com/pcmag/features/Webdiag/_intro2.htm
2. Shareware Link Checkers
http://www8.zdnet.com/pcmag/features/Webdiag/sb1.htm
3.1 Mercury Interactive Corp.: Astra SiteManager
http://www8.zdnet.com/pcmag/features/Webdiag/rev1.htm
3.2 Coast Software Inc.: Coast WebMaster
http://www8.zdnet.com/pcmag/features/Webdiag/rev2.htm
3.3 InContext Systems: WebAnalyzer
http://www8.zdnet.com/pcmag/features/Webdiag/rev3.htm

Figure 1: Web Document Framework of the Article "Diagnose and Repair Your Web Site"

CIKM '99 11/99 Kansas City, MO, USA

Figure 1 gives an example of the Web document framework for an article in an on-line PC magazine. It shows that both the title and URL address contain information about the main topic of the corresponding Web page. The length of the title is usually longer than that of the typical section headings. And the URL address, which consists of the domain name and path directories, shows information about the document hierarchy. To illustrate this, let us look at the last entry in Figure 1. The title of the entry is "`InContext Systems: WebAnalyzer`", which is a Web diagnose product mentioned in the article. The html file name "`rev3.htm`" in the URL address indicates that it is the third product review in the article. The path directory of the URL address reveals the information hierarchy of the Web site clearly. This article "`Webdiag`" is in the "`features`" section of the PC magazine "`pcmag`" under the Web site "`www8.zdnet.com`". Each html file name also corresponds to a chapter of this article. This shows the importance of the title and URL address of a Web page to generate its "TOC".

Further investigation found that understanding the Web document framework is not as easy as it might appear. This is due to the occurrence of compound words - words that are made up of multiple (partial) words through concatenation and cannot be found in ordinary dictionaries. From our experiment, we found that the occurrence frequencies of compound words in URLs and titles are actually quite high, more than 90% in URL addresses and about 44% in titles.

There are at least three reasons for the occurrence of compound words in the Web document framework. The first reason is about the use of delimiters in the URL address. Except for a few pre-defined delimiters such as "." and "/", most delimiters found in traditional documents are not allowed in the URL address. This applies to both the domain name and the path directory part. For example, the URL "`http://www.diamond.bob's.publishing.com/introduction to the company/`" is not a valid one because it violates the definition to the URL address in the HTTP standard [3]. As a result, it is common for people to ignore the delimiters and use compound words instead. In the above instance, one possible form of the URL address might be "`http://www.diamondbobspublishing.com/intro/`". The second reason is due to the limited number of words allowed in an URL address. By incorporating compound words in the URL address, it is hoped that more information can be conveyed to both Web surfers and Web administrator. The third reason is that compound words are often used as a unique way to stand out Web pages.

The popular use of compound words in Web authoring makes Web information understanding and retrieval difficult. Compound words are seldom used in Web queries and stored as Web document indexes. Content understanding is also difficult because they cannot be found in the dictionary. To recover the content hierarchy of Web documents, we propose a new word segmentation and recognition mechanism to understand the information derived from the Web document framework. A maximal bi-directional matching algorithm with heuristic rules is used to resolve ambiguous segmentation and meaning in compound words. An adaptive training process is further employed to build a dictionary of recognizable abbreviations and acronyms. Since the compound words not only appear in the title and URL address but they are also in the hyperlinks and embedded object file names, we will take all these possible information categories into our consideration. Empirical results show that over 75% of the compound words found in the document framework can be understood by our matching algorithm. With the training process, the success rate of recognizing compound words can be increased to about 90%.

2. RELATED WORK

Words in traditional English-like sentences are separated with blank spaces and punctuation marks; they have little concern (if any) about the issue of word segmentation. In World Wide Web, the situation is quite different. However, after separated by the delimiters, compound words without explicit boundaries are still left in the document content, just like the case for oriental languages. A typical analogy is the Chinese language.

A good survey for Chinese word segmentation can be found in Wu and Tseng's paper [7]. A Chinese sentence is composed of a string of characters which does not have delimiters to separate words. The absence of boundaries poses a problem to Chinese text retrieval. Various solutions to word segmentation for Chinese documents have been proposed. They are mainly divided into two approaches: (1) statistical approach, (2) rule-based approach. In the statistical approach, the occurrence frequencies of characters and words, and their syntactic tags are collected. The co-occurrence frequencies and other statistical measurements derived from these data are also used to determine which characters should be grouped together to form a word. Sproat [4] used the association strength between characters to determine the word boundaries. Sun [5] proposed a tagging-based, first-order Markov model to perform word segmentation. Fan and Tsai [2] suggested to use the statistical relaxation method that is usually used in the field of image processing. In the rule-based approach, the maximal matching algorithm is a typical mechanism. In this algorithm, starting at a certain character in the string, the longest valid word is always segmented. Chen and Liu [1] adopted the matching algorithm with six different heuristic rules to resolve the ambiguities. Yeh and Lee [6] presented a unification-based system for identifying words; all information items including lexicon, lexical rules, ambiguity-resolution rules and execution results are stored in the knowledge base.

Although similarities exist between the oriental word segmentation and the compound word segmentation for Web documents, they are fundamentally different problems. We still take Chinese as an example. Apparently, most (if not all) Chinese characters can be found in the dictionary; each of them can be considered as an independent information container. However, in the Web environment, the situation is different. Every author may have a different style to form a compound word; every specific domain may also have its unique rules for the abbreviations and acronyms. It is much more difficult to find two identical compound words from different Web site domains. In other words, the composition pattern for one compound word might not be applicable for another directly. Consequently, while research in Chinese word segmentation gives good foundation to the word segmentation problem of the Web document framework, their

techniques need to be modified and re-evaluated before they can be applied here.

3. BASIC APPROACH TO WEB WORD SEGMENTATION AND RECOGNITION

To address the word segmentation and recognition problem for web document framework, we adopt some basic principles from the rule-based approach. In this approach, words will be identified by matching them with a well-prepared dictionary. The dictionary should have a sufficient amount of word entries. However, independent of the size of the dictionary, the occurrence of compound words makes it very difficult to match every component string exactly with the dictionary entries. The author of a Web document often uses some affixes to compose a compound word. A typical instance is "CompuServe". The first part of this word "Compu" is the prefix of the word "computer". Another example is "edutainment". The prefix "edu" (for "education") and the suffix "tainment" (for "entertainment") are combined to form the word. The dictionary simply cannot cover all possible prefixes and suffixes and their potential combinations. As a result, the problem of understanding web document framework can be formulated into two sub-problems. Given a compound word, how the component strings can be identified and segmented. This is not easy because these components might not be matched exactly with the dictionary entries. The other problem is that after the identification, how these affixes can be associated with the dictionary entries. Again, this is not trivial because they can be acronyms corresponding to some phrases, abbreviations for some special words, or just meaningless and can be neglected.

To address the first problem, we propose to use a combined approach of forward and backward maximal matching. Given a compound word, there are various ways to segment it into multiple strings. One heuristic that we adopt in our word segmentation is to give preference to longer words. The longer the word, the less likely is the case where the characters come together independently, without forming a word. This heuristic has also been verified empirically by many Chinese word segmentation systems. The basic principle of the maximal matching is to scan the input string from the beginning and to match the prefix strings with words in a given dictionary. If more than one word is found in the dictionary, the longest word will be selected. Since the scanning starts from the beginning of the compound word, it is called the forward maximal matching. This mechanism is good for prefix strings. For possible suffix strings that are left from the forward matching mechanism, the backward maximal matching is used. It is similar to the forward matching except that it scans the string from the end back to the beginning, i.e., in the backward direction. After the two scans, possible candidates for the word segmentation are obtained. Heuristic rules are then used to select the most appropriate one. Fine-tuning and adjustments are also made to the combined algorithm because the component strings of compound words might not match with any word entry in the dictionary. This will be discussed later in the next few sections.

For the second problem, we suggest to maintain a table of frequently occurred acronyms and abbreviations in a given domain through a small training corpus. These acronyms or abbreviations must be recognizable and have been accepted by the specific domain. This table will be used together with the normal dictionary in the word segmentation process. To maintain the table up-to-date, new acronyms and abbreviations will be inserted in the table manually through some feedback process. Note that when a table optimized for one domain is used in another domain, its effectiveness might be reduced.

4. SEGMENTATION AND RECOGNITION ALGORITHM

In this section, we are going to describe the word segmentation and recognition algorithm for Web document framework, together its heuristic rules and training process.

4.1. Terminology and Definitions

Before we go into the discussion of the algorithm, some terms need to be defined precisely.

Definition 1: The *document framework WD* of a Web page *W* is consisted of the title and the URL address of the page, together with all hyperlink addresses, embedded object file names and their alternate texts in the page.

Definition 2: A *word* is a consecutive string of characters separated by the blank space or some other delimiters.

- Delimiter = {d | d ∈ Character ^ d ∉ ALPHABET ^ d ∉ DIGIT}
- ALPHABET = a | b | ... | z | A | B | ... | Z
- DIGIT = 0 | 1 | 2 | ... | 9

For example, given a URL address "http://www.iscs.nus.edu.sg/students/", all the words in it are "http", "www", "iscs", "nus", "edu", "sg" and "students". While some words (such as "students") can be found in a dictionary, the others might not be. Hence, we have following definition:

Definition 3: A *dictionary word* is a consecutive string of characters that exists as an entry in a given dictionary.

For example, the words in the URL "http://www.ausedcar.com/car/" are "http", "www", "ausedcar", "com" and "car". The sub-string "car" in the word "ausedcar" is a dictionary word. The word "car" itself is also a dictionary word. In other words, a dictionary word can be a word or the sub-string of a compound word in the Web document framework.

Definition 4: Given a Web document framework *WD*, its *word set S* is the set of all the words extracted from *WD*.

Definition 5: The *Reference Base RB* of an item x in a Web document framework *WD* is defined as a set of four possible elements $\{C_1(x), C_2(x), C_3(x), C_4(x)\}$ that are constructed by the following relationships:

$$SeqCont = \begin{cases} title\ text & if\ x = URL\ Address\ in\ WD \\ anchor\ text\ of\ the\ link & if\ x = hyperlink\ in\ WD \\ alternate\ text\ of\ the\ object & if\ x = embedded\ object\ in\ WD \end{cases}$$

$C_1(x)$ = concatenation of all words as *SeqCont*

$C_2(x)$ = concatenation of all words not in the stop-list as *SeqCont*

$C_3(x)$ = concatenation of the first character of all words as *SeqCont*

$C_4(x)$ = concatenation of the first character of all words not in the stop-list as *SeqCont*

Definition 6: The Reference Based Set *RBS* of a Web document framework *WD* is defined as the collection set of the reference bases associated with all items in *WD*.

Definition 7: A word w in the word set S of a Web document framework *WD* is said to be a *RB-based word* if it is a (sub-)string of any element in its corresponding reference based set *RBS(WD)*.

As an example, the word "welles" is extracted from the URL address "http://www.wellesbowen.com/". The title of the Web page is "Toledo Real Estate Welles Bowen Realtors". Then,

- *RB*("http://www.wellesbowen.com") = {C_1, C_2, C_3, C_4}
- C_1="toledorealestatewellesbowenrealtors",
- C_2="toledorealestatewellesbowenrealtors",
- C_3="trewbr", and
- C_4="trewbr"

"welles" is the sub-string of C_1 and C_2, so it is a *RB*-based word.

Definition 8: A word w in the word set of a Web document framework *WD* is said to be a *non-dictionary word* if it does not match in a dictionary and is not a *RB*-based word in *RBS(WD)*

Based on this definition, the string "edu", "com", and "nus" are all non-dictionary words. Note that a non-dictionary word can be a sub-string of a dictionary word.

With these definitions, we can now proceed to discuss the data and control flow of the word segmentation algorithm, together with its heuristic rules and training process for word recognition.

4.2. Design Considerations

Given a Web page, the document framework, which includes the title, URL address, hyperlinks and embedded object file names, is extracted. After the removal of the commonly-used words in the stop list, each extracted word will be matched against the dictionary. If no matching is found, this word will be considered as a compound word and it will be sent for segmentation. The matching process, with both forward and backward matching will be executed to obtain all the possible segmentation candidates. Then, the maximal heuristic rules will be applied to disambiguate and select the most appropriate segmentation choice from the candidates. Any matched affix (sub-)string will then be removed from the input word set and the procedure will be repeated for the remaining strings until the input word set is empty. For those compound words that fail in the recognition process, they might be set for manual recognition (with table updating, will be discussed in Section 4.5). The flow diagram of the word segmentation and recognition for Web document framework is shown in Figure 2.

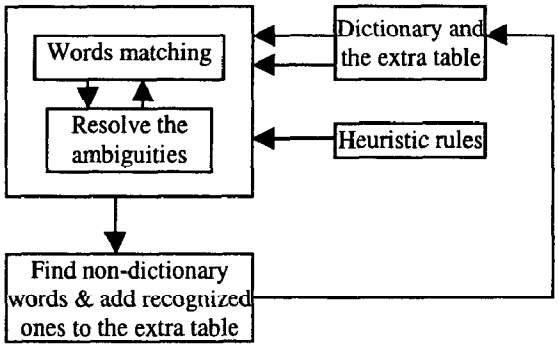

Figure 2: Flow Diagram of Web Segmentation and Recognition Algorithm

The input of our algorithm is the extracted words in the word set S of the Web document framework *WD*. These words come from the title, URL address, hyperlinks and embedded object file names. During our experimentation, we observed that compound words can sometimes be segmented at the position of the turning point from uppercase character to lowercase or vice verse. For instance, a word "AIBonline" can be separated into "AIB" and "online". Thus, before the matching process, such preprocessing for the change of letter case can be performed. If a match is found, the compound word will be segmented according to this pattern and it will not go through the combined forward and backward matching process. Similar situation applies when there are numeric characters in a compound word.

In our segmentation algorithm, the concepts of reference base, reference based set, and reference based word (i.e. Definition 5-7) are introduced. These concepts are mainly used to describe compound words and abbreviations that are commonly formed by concatenation of words or first characters of words in the context respectively. Words in the reference base set are usually not found in the dictionary directly. However, from their reference bases, hints about their meanings can be deduced. Given a *RB*-based word, the segmentation for this word will be mapped to its counterpart in its associated context. Its entry will also be updated into the dictionary. Note that this context mapping for the *RB*-based words will be processed before the matching algorithm. This is to ensure that the matching algorithm will not segment them wrongly. We will use "AIBonline" as the example. Its context in the title is "American Institute of Baking". Consequently, the string "AIB" will be mapped to the word collection of "American", "institute" and "baking".

With the basic understanding of our segmentation process, let us go into details on the coding of the matching algorithm and its heuristic rules in the next two sub-sections.

4.3. Pseudo-Code for Matching Algorithm

```
Procedure Forward-Scan (word w)
count := 0;
start_position := 0;
current_position := 1;
while (current_position != end of w) {
    current_string := sub-string s from the start_position to the
            current_position of string w;
        match current_string against the dictionary D;
    if (match) {
        push current_string onto the stack candidate_word;
        move current_position by one character forward;
    } else {
        if (current_string is the prefix of some entry in D) {
            move current_position by one character forward;
        } else {
                if (candidate_word != empty) {
                pop item from stack candidate_word;
                push item into array
                candidate_segment[count];
                start_position := start_position +
                WordLength(item);
                current position := start_position + 1;
            } else {
                remove the last character from current_string;
                push current_string onto array candidate-
            segment[count];
                start_position := current_position - 1;
            }
            count ++;
        }
    }
    if (current_position = end of w) {
        if (current_string = top of stack candidate-word) {
            pop item from stack candidate-word;
            push item into array candidate-segment[count];
        } else {
                push current_string into stack candidate-
segment[count];
        }
        count ++;
    }
    while (candidate_word != empty) {
        pop item from stack candidate-word;
        push item into array candidate-segment[count];
        count ++;
        }
}
```

Figure 3: Forward Matching Algorithm

The matching algorithm is made up of two components: a forward matching process and a backward matching process. The pseudo-code for the forward matching algorithm is given in Figure 3. To illustrate how the algorithm works, let us assume the matching for a compound word "compuserve". The steps involved in the matching are given as follow:

1. "c" matches with the dictionary. Since there are lots of entries in the dictionary that start with the character 'c', the pointer will move one position forward.
2. "co" matches with the dictionary. Since the number of dictionary entries that start with the string "co" is larger than zero, the pointer will move one position forward.
3. "com" matches with the dictionary. Since the number of dictionary entries that start with the string "com" is larger than zero, the pointer will move one position forward.
4. "comp" matches with the dictionary. Since the number of entries that start with the string "comp" is larger than zero, the pointer will move one position forward.
5. "compu" matches with the dictionary. Since the number of entries that starts with the string "compu" is larger than zero, the pointer will move one position forward.
6. "compus" matches with the dictionary. There is no dictionary entry that can match it. So a separation point is found. "compu" will be taken as a non-dictionary, segmented candidate and will be removed from the testing string.
7. "s" matches with the dictionary. Since there are lots of entries in the dictionary that start with the character 's', the pointer will move one position forward.
8. "se" matches with the dictionary. Since the number of entries that start with the string "se" is larger than zero, the pointer will move one position forward.
9. "ser" matches with the dictionary. Since the number of entries that start with the string "ser" is larger than zero, the pointer will move one position forward.
10. "serv" matches with the dictionary. Since the number of entries that start with the string "serv" is larger than zero, the pointer will move one position forward.
11. "serve" finds a match in the dictionary. Hence, it will be taken as a dictionary word and be removed from the testing string.
12. The input word string is empty and this causes the matching procedure to end.

So the word "compuserve" will be segmented into "compu" and "serve". Sometimes, more than one match can be found for a sub-string. For example, the sub-string "ban" in the word "bannerad" can be a dictionary word itself or a sub-string of the dictionary word "banner". When it happens, all the possible matches will be stored. As a result, several different segmentations might exist after the forward matching. If there is at least one choice in the segmentation candidate set that consists of dictionary words and *RB*-based words only, backward matching will not performed. Otherwise, the input word will continue to be segmented with the backward matching algorithm.

The backward matching algorithm is similar to the forward matching one, except that the word is scanned from the end back to the beginning. In this case, a reverse dictionary (i.e. with character order of each dictionary entry reversed) is used. It is important to note that the results of the two matching algorithms are often different. For example, the word "imaginet" will be segmented as "imagine" and "t" by forward matching, and as "imagi" and "net" by backward matching.

4.4. Heuristic Rules

In word segmentation, one of the most powerful and commonly used disambiguation rule is the heuristic of maximal matching. There are different variations for the maximal matching. After detailed investigation on the feasibility and applicability of these variations in the context of Web documents and the composition pattern of English words, we propose the following maximal matching rules. The priorities/importance of these rules are in the order of their listing sequence (i.e. smallest number implies highest priority).

Rule 1: If there exists a segmentation candidate that contains words in the reference base set *RBS* only, this candidate will be chosen as the final result.

This rule suggests that the context information is a key factor to perform word segmentation and recognition for Web document framework. For example, given an URL address "http://www.diamondbobpublishing.com" and its associated title "Diamond Bob's Publishing House". After the reference base *RB* of the URL address is computed, a string match between the word "diamondbobpublishing" in the URL address and its title is found. As a result, the compound word "diamondbobpublishing" can easily be segmented to the candidate set {"diamond", "bob" and "publishing"} from the title information. Note that in this case, no forward or backward matching will be required.

Rule 2: If there exists a segmentation candidate that contains *RB*-based words and dictionary words only, this candidate will be retained as the correct segmentation.

A simple example is the word "abbottrealty". "abbott" is a person's name and it appears in its reference base as a *RB*-based word. "realty" is a dictionary word. Hence, this candidate will be chosen.

Rule 3: If the component strings of two segmentation candidates are all dictionary words, then the one with the least number of words will be chosen for segmentation.

This rule is to enforce that a segmentation choice with only dictionary words will be given higher priority in selection. Furthermore, if there are more than one such segmentation choices, the longest matching one (i.e. the least number of words), which follows the maximal heuristic rule, will be retained. For example, "americanet" can be segmented as "America" and "net" which are all dictionary words. The sub-string "america" can be a dictionary word itself and can also be a sub-string of "American". If only the longest matching rule is used, the segmentation will become "American" and "et", which contains a non-dictionary word. Another example is the word "apart". "apart" itself is a dictionary word. But it can also be segmented into two dictionary words - "a" and "part". According to our rule, the first segmentation choice will be chosen. That is "apart" will be treated as one single word.

Rule 4: If there exists a segmentation candidate that consists of *RB*-based words and non-dictionary words only, it will be retained as the correct segmentation.

Rule 5: If the component strings of two segmentation candidates include dictionary words and non-dictionary words, the one with the least number of non-dictionary words will be chosen for segmentation. If the numbers of non-dictionary words for all the candidates are the same, then the result from backward matching will be retained.

These two rules simply imply that dictionary and *RB*-based words are given higher priority in the selection process; segmentation result from the backward matching is also prior to the one from the forward matching. As an example, one segmentation choice for the word "prowrestling" is "prow", "rest" and "ling". Among them, "prow" and "rest" are dictionary words, and "ling" is a non-dictionary word. Another segmentation choice is "pro" and "wrestling", which is obtained from the backward matching. "pro" is non-dictionary word and "wrestling" is a dictionary word. Since the numbers of non-dictionary words are the same, the second candidate, which is the result from backward matching will be chosen.

Rule 6: If the component strings of two segmentation candidates are all non-dictionary words only, the one with the least number of words will be chosen for segmentation. If their numbers are the same, the result from the backward matching will be retained.

Finally, if ambiguities in segmentation are still not resolved after the applications of these heuristic rules, the final segmentation choice will be picked randomly. This should not have much impact to the overall performance. Next, we are going to describe the training process for the setting up and maintenance of an additional table for abbreviations and acronyms.

4.5. Training Process

It is pointed out in the previous section that due to the existence of abbreviations, acronyms, and compound words that are made up from affixes, standard dictionary often cannot give the best segmentation result in the matching process. These words simply cannot be found in the dictionary. The decision for the points of segmentation in the affix case is also difficult to make. The concept of reference base might be able to help in the segmentation process, but this depends on the co-occurrence of the compound words and their associated context in the Web document framework. To make up for this, we propose to establish an additional table for those frequently occurred, recognizable non-dictionary words. This table will be used simultaneously with the standard dictionary in the matching process. It is hoped that the combined dictionary will improve the effectiveness of word segmentation and web document understanding. Just like the multilingual document translation, the table will be somewhat domain-specific; application of a table from one domain to another domains will result in ineffectiveness.

To establish such a table, we propose the following training process with sample corpus. Given a domain area, sample Web documents are selected and they will be segmented by our algorithm to find out all compound words in them. Then, all the non-dictionary words are extracted from the result. Those frequently occurred ones will be recognized by some area-relevant knowledge. They are mainly abbreviations of dictionary words or acronyms that have commonly been accepted in the area. For example, "gov" is the abbreviation of "government", "org" is the abbreviation of "organization", "edu" is the abbreviation of "education" and "com" is the abbreviation of "commerce". These abbreviations are widely accepted in the World Wide Web. In the field of computer science, "sys" is the abbreviation of "system" and "img" is the abbreviation of "image". The recognized, non-dictionary words will finally be added into the table. This training process will be iterated several times until the content of the table is stabilized.

Finally, as we mentioned previously, the table will also be kept update by the feedback of the matching of compound words in their associated reference base set.

5. EXPERIMENTAL STUDY

To test our segmentation and matching algorithm and the associated heuristic rules, we implemented a word segmentation system for Web document framework. The word dictionary used in our system is constructed primarily from the system dictionary in Digital UNIX system and it has about thirty thousand word entries. For the input testing data, Internet search engines are used to collect Web pages for the analysis. This is to ensure the randomness of the data samples. Furthermore, Web pages are also collected from different domain areas to study the domain-sensitivity of the training process. Four domain areas are chosen; they are "finance", "network", "chemistry" and "biology". In each domain test, the name of the area is used as the input to the search engine and the top 50 Web pages are collected as the training corpus for the setting up of the additional table. Then the next 50 more pages will be used for the testing of the algorithm. In the experiment, the following four parameters are measured:

- N_1 = Number of words in a Web document framework (i.e. title, URL addresses, hyperlinks, and embedded object names).
- N_2 = Number of compound words in a Web document framework.
- N_3 = Number of words of a Web document framework after segmentation.
- N_4 = Number of recognisable words of a Web document framework.

With these data, the followings can be computed:

- *CP* (Percentage of compound words) = N_2 / N_1
- *WI* (Increased word rate) = $(N_3 - N_2) / N_2$
- *RE* (Recognition rate) = N_4 / N_3

Among the above measurements, *CP* gives a hint on the importance of segmentation for the compound words. The value of WI implies the average number of words a compound word will be segmented into. Finally, *RE* can measure the effectiveness of our algorithm. Note that since common words in the stop list (e.g. common words such as "the", "of", etc. and Web-specific words such as "www", "http" etc.) are removed from the Web document framework before segmentation, their word count is not included in the above measurement.

5.1. Results

Due to the limited space of the paper and the consistency of the results among different domain areas, we will mainly use the "network" area as an example for discussion.

Figure 4 gives the distribution trend for the number of words extracted from the title, URL address, hyperlinks and embedded object file names in the Web document framework. In our testing corpus, the framework for more than 70% of the Web documents has word count ranging from 10 to 200. This is quite substantial, especially when it is compared with our previous experimental result that more than 60% of the Web pages have an average page word count (including all kinds of information categories) of 100 to 1000 only. This confirms our argument that the Web document framework is an important source of information for Web document understanding. The fact that the average entropy of the words from the Web document framework is usually higher than the words in the Web pages makes the segmentation problem more important in Web document understanding.

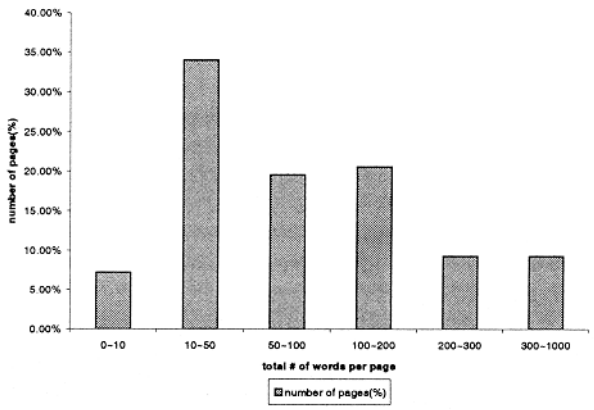

Figure 4: Distribution of Total Number of Words per Page (Network)

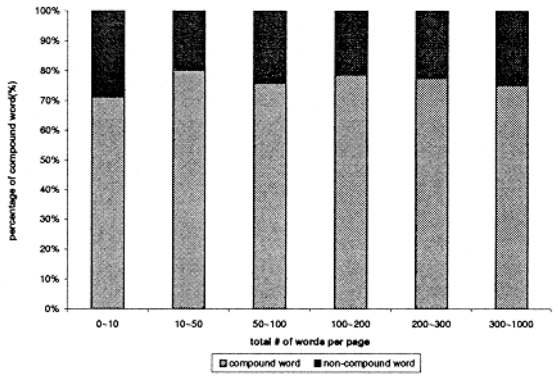

Figure 5: Percentage of Compound Words in Web Document Framework (Network)

Figure 5 shows the percentage of compound words in the Web document framework. The graph clearly agrees with what we predict: there is a very high percentage of compound words in the Web document framework. No matter whether the total word count of the framework is less than 10 or around 1000, the percentage of compound word is often over 70%, which is very high. This figure verifies our claim on the importance of understanding compound words in Web document framework.

A compound word is the concatenation of several separate strings; thus the understanding of a compound word implies to segment it back to the original component strings. After the segmentation for a set of compound words, the total word count will be raised. The *WI* parameter is such a measurement. Figure 6 plots the distribution graph of *WI* for different ranges of total word count. It shows that there is an average of 60% increase in the word count, which is quite a lot. This confirms with our argument that compound words are very concise and have higher information entropy. Furthermore, the increase in word count *WI* is quite independent of the framework size (except for very small page sizes). This is reasonable because the use of compound

words is more the style of the Web designers than the constraint of the Web page size.

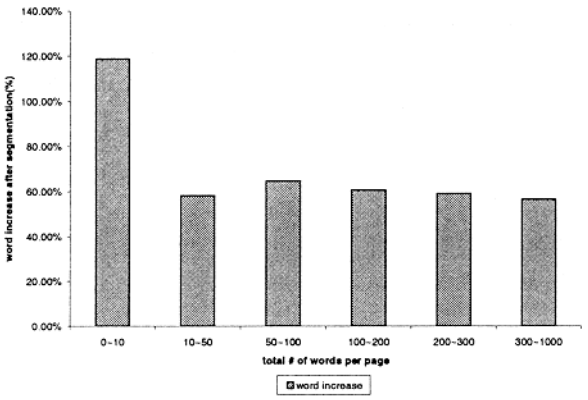

Figure 6: Distribution of Increase in Word Count (Network)

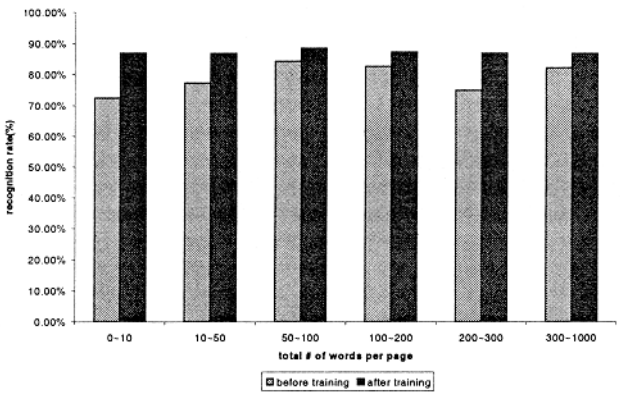

Figure 7: Distribution of Recognition Rate (Network)

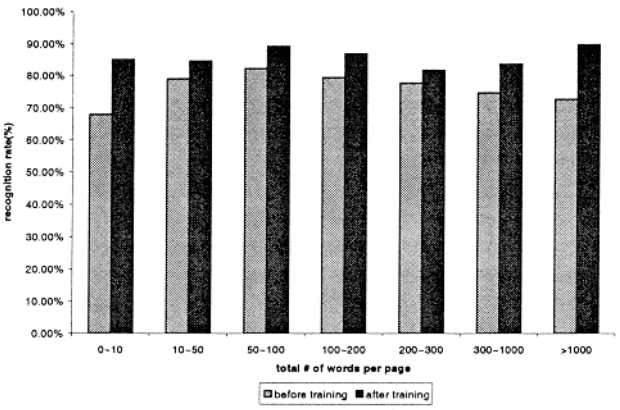

Figure 8: Distribution of Recognition Rate (Biology)

Figure 7 and Figure 8 give the recognition rate of our algorithm before and after training in the domain areas of network and biology. The first series of each column is the recognition rate before the training process and the second series is the value after training. On average, the recognition ranges from 70% to 80%, which is reasonably high. This shows the applicability and potentials of our proposal. Furthermore, it is quite insensitive to the Web document size, which is expected. With training, an additional of a few percents to 15% increase in the recognition rate is obtained. This justifies the use of the additional table. Note that the effect of training is actually expected to be higher. After detailed investigation, we found out that this lower than expected result is due to the inaccurate query result returned to the Web surfers. Quite a number of "top Web pages" returned are actually not related the query subject. Since the second mapping table is trained from this corpus, a less effective, domain specific table results.

6. CONCLUSION

Web document framework, which is mainly consisted of the title, URL address, hyperlinks, and embedded object files names, is an important source of information for Web document understanding. From the statistical data, it is found that a high percentage of the words are actually compound words, which cannot be understood by the standard dictionary. So the issues of segmenting the compound words, followed by partial word recognition are the two main concerns that we focus in this paper. A combined forward and backward matching, together with its heuristic rules, is proposed to approach these problems. In particular, the concept of reference base is introduced. Our experimental results verify that the maximal matching algorithm, with its heuristic method, is very effective. Considering the frequent occurrences of abbreviations or acronyms in compound words of Web documents, a training process is also applied to the algorithm to improve the recognition rate for such words. With all these techniques working together, an average of about 90% recognition rate is obtained, which is high enough to justify its practical applicability and potentials.

References

[1] Keh-Jiann Chen and Shing-Huan Liu, "Word Identification for Mandarin Chinese Sentences," *Proceedings of COLING-92*, 1992, pp. 101-107.

[2] Charng-Kang Fan and Wen-Hsiang Tsai, "Automatic Word Identification in Chinese Sentences by the Relaxation Techniques," *Computer Processing of Chinese & Oriental Languages*, Vol. 4, No. 1, Nov 1998.

[3] Hypertext Transfer Protocol - HTTP/1.0 http://www.ics.uci.edu/pub/ietf/http/rfc1945.html.

[4] Richard Sproat and Chilin Shih, "A Statistical Method for Finding Word Boundaries in Chinese Text," *Computer Processing of Chinese & Oriental Languages*, Vol. 4, No. 4, March 1990, pp. 336-349.

[5] M. S. Sun, T. B. Y. Lai, S. C. Lun, and C. F. Sun, "Some Issues on the Statistical Approach to Chinese Word Identification," *Proceedings of the 1992 International Conference on Chinese Information Processing*, Vol. 1, 1992, pp. 246-253.

[6] Ching-Long Yeh and His-Jian Lee, "Rule-based Word Identification for Mandarin Chinese Sentences - A Unification Approach," *Computer Processing of Chinese & Oriental Languages*, Vol. 5, No. 2, March 1991.

[7] Zimin Wu and Gwyneth Tseng, "Chinese Text Segmentation for Text Retrieval: Achievements and Problems," *Journal of the American Society for Information Science*, 44(9), 1993, pp. 532-542.