Hierarchical Clustering of Words and Application to NLP Tasks

Akira Ushioda*

Fujitsu Laboratories Ltd.
Kawasaki, Japan
email: ushioda@flab.fujitsu.co.jp

Abstract

This paper describes a data-driven method for hierarchical clustering of words and clustering of multiword compounds. A large vocabulary of English words (70,000 words) is clustered bottom-up, with respect to corpora ranging in size from 5 million to 50 million words, using mutual information as an objective function. The resulting hierarchical clusters of words are then naturally transformed to a bit-string representation of (i.e. *word bits* for) all the words in the vocabulary. Evaluation of the word bits is carried out through the measurement of the error rate of the ATR Decision-Tree Part-Of-Speech Tagger. The same clustering technique is then applied to the classification of multiword compounds. In order to avoid the explosion of the number of compounds to be handled, compounds in a small subclass are bundled and treated as a single compound. Another merit of this approach is that we can avoid the data sparseness problem which is ubiquitous in corpus statistics. The quality of one of the obtained compound classes is examined and compared to a conventional approach.

1 Introduction

One of the fundamental issues concerning corpus-based NLP is that we can never expect to know from the training data all the necessary *quantitative* information for the words that might occur in the test data if the vocabulary is large enough to cope with a real world domain. In view of the effectiveness of class-based n-gram language models against the data sparseness problem (Kneser and Ney 1993, Ueberla 1995), it is expected that classes of words are also useful for NLP tasks in such a way that statistics on classes are used whenever statistics on individual words are unavailable or unreliable. An ideal type of clusters for NLP is the one which guarantees mutual substitutability, in terms of both syntactic and semantic soundness, among words in the same class (Harris 1951, Brill and Marcus 1992). Take, for example, the following sentences.

(a) He went to the house by car.
(b) He went to the apartment by bus.
(c) He went to the ? by ? .
(d) He went to the house by the sea.

Suppose that we want to parse sentences using a statistical parser and that sentences (a) and (b) appeared in the training and test data, respectively. Since (a) is in the training data, we know that the prepositional phrase *by car* is attached to the main verb *went*, not to the noun phrase *the house*. Sentence (b) is quite similar to (a) in meaning, and identical to (a) in sentence structure. Now if the words *apartment* and *bus* are *unknown* to the parsing system

*A part of this work is done when the author was at ATR Interpreting Telecommunications Research Laboratories, Kyoto, Japan.

(i.e. never occurred in the training data), then sentence (b) must look to the system very much like (c), and it will be very hard for the parsing system to tell the difference in sentence structure between (c) and (d). However, if the system has access to a predefined set of classes of words, and if *car* and *bus* are in the same class, and *house* and *apartment* are in another class, it will not be hard for the system to detect the similarity between (a) and (b) and assign the correct sentence structure to (b) without confusing it with (d). The same argument holds for an example-based machine translation system. In that case, an appropriate translation of (b) is expected to be derived with an example translation of (a) if the system has an access to the classes of words. Therefore, it is desirable that we build clustering of the vocabulary in terms of *mutual substitutability.*

Furthermore, clustering is much more useful if the clusters are of variable granularity. Suppose, for example, that we have two sets of clusters, one is finer than the other, and that word-1 and word-2 are in different finer classes. With finer clusters alone, the amount of information on the association of the two words that the system can obtain from the clusters is minimal. However, if the system has a capability of falling back and checking if they belong to the same coarser class, and if that is the case, then the system can take advantage of the class information for the two words. When we extend this notion of two-level word clustering to many levels, we will have a tree representation of all the words in the vocabulary in which the root node represents the whole vocabulary and a leaf node represents a word in the vocabulary. Also, any set of nodes in the tree constitutes a partition (or clustering) of the vocabulary if there exists one and only one node in the set along the path from the root node to each leaf node.

In the following sections, we will first describe a method of creating a binary tree representation of the vocabulary and present results of evaluating and comparing the quality of the clusters obtained from texts of very different sizes. Then we will extend the paradigm of clustering from word-based clustering to compound-based clustering. In the above examples we looked only at the mutual substitutability of words; however, a lot of information can also be gained if we look at the substitutability of word compounds for either other word compounds or single words. We will introduce the notion of compound-classes, propose a method for constructing them, and present results of our approach.

2 Hierarchical Clustering of Words

Several algorithms have been proposed for automatically clustering words based on a large corpus (Jardino and Adda 91, Brown et al. 1992, Kneser and Ney 1993, Martin et al. 1995, Ueberla 1995). They are classified into two types. One type is based on shuffling words from class to class starting from some initial set of classes. The other type repeats merging classes starting from a set of singleton classes (which contain only one word). Both types are driven by some objective function, in most cases by perplexity or average mutual information. The merit of the second type for the purpose of constructing hierarchical clustering is that we can easily convert the history of the merging process to a tree-structured representation of the vocabulary. On the other hand, the second type is prone to being trapped by a local minimum. The first type is more robust to the local minimum problem, but the quality of classes greatly depends on the initial set of classes, and finding an initial set of good quality is itself a very difficult problem. Moreover, the first approach only provides a means of partitioning the vocabulary and it doesn't provide a way of constructing a hierarchical clustering of words. In this paper we adopt the merging approach and propose an improved method of constructing hierarchical clustering. An attempt is also made to combine the two types of clustering and some results will be shown. The combination is realized by the construction of clusters using the merging method followed by the reshuffling of words from class to class.

Our word bits construction algorithm (Ushioda 1996) is a modification and an extension

of the mutual information (MI) clustering algorithm proposed by Brown et al. (1992). The reader is referred to (Ushioda 1996) and (Brown et al. 1992) for details of MI clustering, but we will first briefly summarize the MI clustering and then describe our hierarchical clustering algorithm.

2.1 Mutual Information Clustering Algorithm

Suppose we have a text of T words, a vocabulary of V words, and a partition π of the vocabulary which is a function from the vocabulary V to the set C of classes of words in the vocabulary. Brown et al. showed that the likelihood $L(\pi)$ of a bigram class model generating the text is given by the following formula.

$$L(\pi) = -H + I \tag{1}$$

Here H is the entropy of the 1-gram word distribution, and I is the average mutual information (AMI) of adjacent classes in the text and is given by equation 2.

$$I = \sum_{c_1,c_2} Pr(c_1c_2) \log \frac{Pr(c_1c_2)}{Pr(c_1)Pr(c_2)} \tag{2}$$

Since H is independent of π, the partition that maximizes the AMI also maximizes the likelihood $L(\pi)$ of the text. Therefore, we can use the AMI as an objective function for the construction of classes of words.

The mutual information clustering method employs a bottum-up merging procedure. In the initial stage, each word is assigned to its own distinct class. We then merge two classes if the merging of them induces minimum AMI reduction among all pairs of classes, and we repeat the merging step until the number of the classes is reduced to the predefined number C. Time complexity of this basic algorithm is $O(V^5)$ when implemented straightforwardly, but it can be reduced to $O(V^3)$ by storing the result of all the trial merges at the previous merging step.

Even with the $O(V^3)$ algorithm, however, the calculation is not practical for a large vocabulary of order 10^4 or higher. Brown et al. proposed the following method, which we also adopted. We first make V singleton classes out of the V words in the vocabulary and arrange the classes in the descending order of frequency, then define the *merging region* as the first $C + 1$ positions in the sequence of classes. So initially the $C + 1$ most frequent words are in the merging region. Then do the following.

1. Merge the pair of classes in the merging region merging of which induces minimum AMI reduction among all the pairs in the merging region.
2. Put the class in the $(C + 2)^{nd}$ position into the merging region and shift each class after the $(C + 2)^{nd}$ position to its left.
3. Repeat 1. and 2. until C classes remain.

With this algorithm, the time complexity becomes $O(C^2V)$ which is practical for a workstation with V in the order of 100,000 and C up to 1,000.

2.2 Word Bits Construction Algorithm

The simplest way to construct a tree-structured representation of words is to construct a dendrogram as a byproduct of the merging process, that is, to keep track of the order of merging and make a binary tree out of the record. A simple example with a five word vocabulary is shown in Figure 1. If we apply this method to the above $O(C^2V)$ algorithm straightforwardly, however, we obtain for each class an extremely unbalanced, almost left branching subtree. The

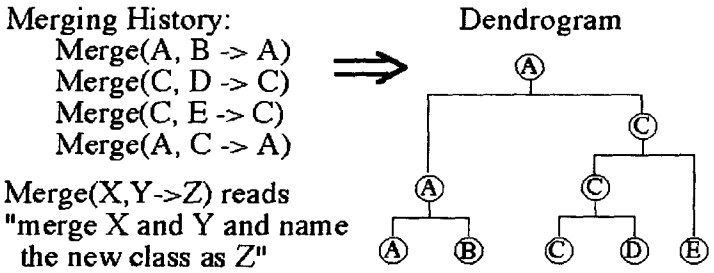

Figure 1: Dendrogram Construction

reason is that after classes in the merging region are grown to a certain size, it is much less expensive, in terms of AMI, to merge a singleton class with lower frequency into a higher frequency class than merging two higher frequency classes with substantial sizes.

A new approach we adopted incorporates the following steps.

1. MI-clustering: Make C classes using the mutual information clustering algorithm with the merging region constraint mentioned in (2.1).

2. Outer-clustering: Replace all words in the text with their class token[1] and execute binary merging without the merging region constraint until all the classes are merged into a single class. Make a dendrogram out of this process. This dendrogram, D_{root}, constitutes the upper part of the final tree.

3. Inner-clustering: Let $\{C(1), C(2), ..., C(C)\}$ be the set of the classes obtained at step 1. For each i ($1 \leq i \leq C$) do the following.

   (a) Replace all words in the text except those in C(i) with their class token. Define a new vocabulary V' = $V_1 \cup V_2$, where V_1 = {all the words in C(i)}, V_2 = $\{C_1, C_2, ..., C_{i-1}, C_{i+1}, C_C\}$, and C_j is a token for C(j) for $1 \leq j \leq C$. Assign each element in V' to its own class and execute binary merging with a merging constraint such that only those classes which only contain elements of V_1 can be merged. This can be done by ordering elements of V' with elements of V_1 in the first $|V_1|$ positions and keep merging with a merging region whose width is $|V_1|$ initially and decreases by one with each merging step.

   (b) Repeat merging until all the elements in V_1 are put in a single class.

   Make a dendrogram D_{sub} out of the merging process for each class. This dendrogram constitutes a subtree for each class with a leaf node representing each word in the class.

4 Combine the dendrograms by substituting each leaf node of D_{root} with the corresponding D_{sub}.

This algorithm produces a balanced binary tree representation of words in which those words which are close in meaning or syntactic feature come close in position. Figure 2 shows an example of D_{sub} for one class out of 500 classes constructed using this algorithm with a vocabulary of the 70,000 most frequently occurring words in the Wall Street Journal Corpus. Finally, by tracing the path from the root node to a leaf node and assigning a bit to each branch with zero or one representing a left or right branch, respectively, we can assign a bit-string (*word bits*) to each word in the vocabulary.

[1] In the actual implementation, we only have to work on the bigram table instead of the whole text.

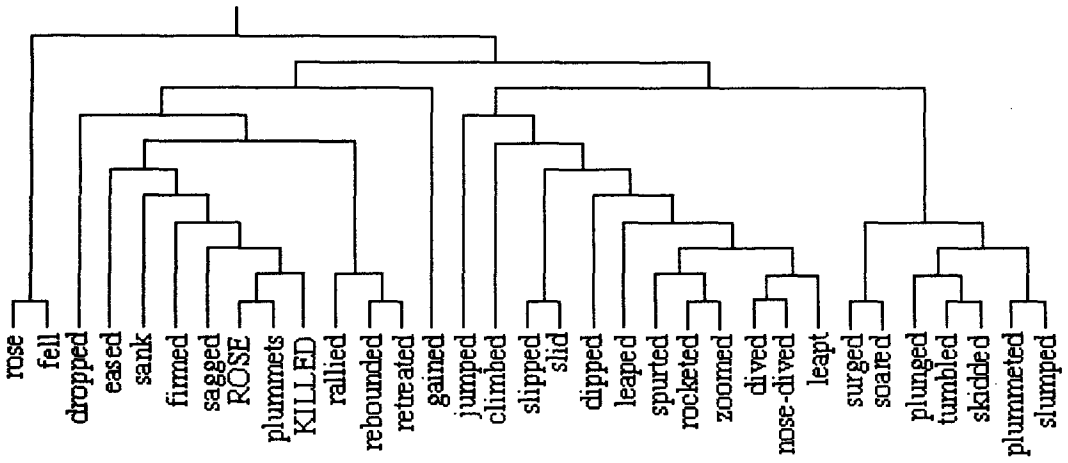

Figure 2: Sample Subtree for One Class

3 Word Clustering Experiments

We performed experiments using plain texts from six years of the Wall Street Journal Corpus to create clusters and word bits. The sizes of the texts are 5 million words (MW), 10MW, 20MW, and 50MW. The vocabulary is selected as the 70,000 most frequently occurring words in the entire corpus. We set the number C of classes to 500. The obtained hierarchical clusters are evaluated via the error rate of the ATR Decision-Tree Part-Of-Speech Tagger.

Then as an attempt to combine the two types of clustering methods discussed in Section 2, we performed an experiment for incorporating a word-reshuffling process into the word bits construction process.

3.1 Decision-Tree Part-Of-Speech Tagging

The ATR Decision-Tree Part-Of-Speech Tagger is an integrated module of the ATR Decision-Tree Parser which is based on SPATTER (Magerman 1994). The tagger employs a set of 441 syntactic tags, which is one order of magnitude larger than that of the University of Pennsylvania Treebank Project. Training texts, test texts, and held-out texts are all sequences of word-tag pairs. In the training phase, a set of *events* are extracted from the training texts. An *event* is a set of feature-value pairs or question-answer pairs. A feature can be any attribute of the context in which the current word *word(0)* appears; it is conveniently expressed as a question. Tagging is performed left to right. Figure 3 shows an example of an event with a current word *like*. The last pair in the event is a special item which shows the *answer*, i.e., the correct tag of the current word. The first two lines show questions about identity of words around the current word and tags for previous words. These questions are called *basic questions*. The second type of questions, *word bits questions*, are on clusters and word bits such as *is the current word in Class 295?* or *what is the 29th bit of the previous word's word bits?*. The third type of questions are called *linguist's questions* and these are compiled by an expert grammarian. Such questions could concern membership relations of words or sets of words, or morphological features of words.

Out of the set of events, a decision tree is constructed. The root node of the decision tree represents the set of all the events with each event containing the correct tag for the corresponding word. Probability distribution of tags for the root node can be obtained by calculating relative frequencies of tags in the set. By asking a value of a specific feature on each event in the set, the set can be split into N subsets where N is the number of possible values for the feature. We can then calculate conditional probability distribution of tags for

Event-128:

{

⟨ word(0), "like" ⟩ ⟨ word(-1), "flies" ⟩ ⟨ word(-2), "time" ⟩ ⟨ word(1), "an" ⟩ ⟨ word(2), "arrow" ⟩

⟨ tag(-1), "Verb-3rd-Sg-type3" ⟩ ⟨ tag(-2), "Noun-Sg-type14" ⟩

. . . . . (Basic Questions)

⟨ Inclass?(word(0), Class295), "yes" ⟩ ⟨ WordBits(Word(-1), 29), "1" ⟩

. . . . . (WordBits Questions)

⟨ IsMember?(word(-2), Set("and","or","nor")), "no" ⟩ ⟨ IsPrefix?(Word(0), "anti"), "no" ⟩

. . . . . (Linguist's Questions)

⟨ Tag, "Prep-type5" ⟩

}

Figure 3: Example of an event

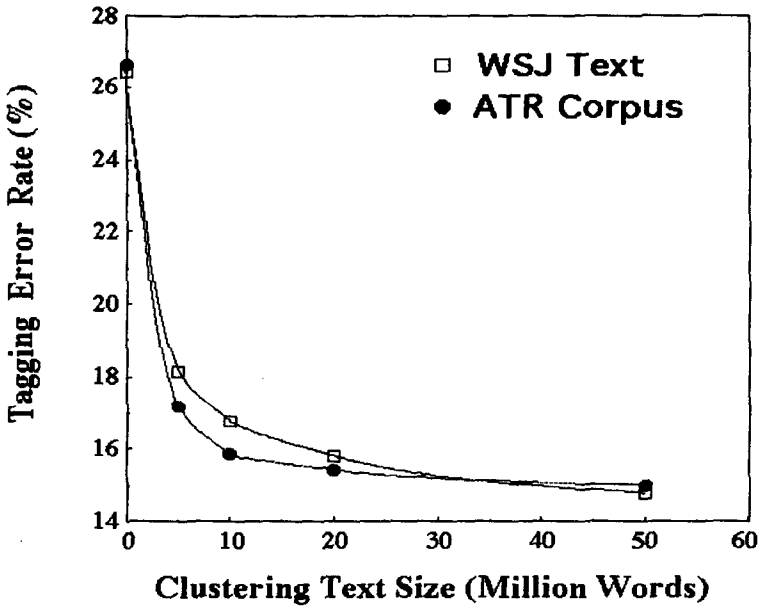

Figure 4: Tagging Error Rate

each subset, conditioned on the feature value. After computing for each feature the entropy reduction incurred by splitting the set, we choose the best feature which yields maximum entropy reduction. By repeating this step and dividing the sets into their subsets we can construct a decision tree whose leaf nodes contain conditional probability distributions of tags. The obtained probability distributions are then smoothed using the held-out data. The reader is referred to (Magerman 1994) for the details of smoothing. In the test phase the system looks up conditional probability distributions of tags for each word in the test text and chooses the most probable tag sequences using beam search.

We used WSJ texts and the ATR corpus for the tagging experiment. The WSJ texts are re-tagged manually using the ATR syntactic tag set. The ATR corpus is a comprehensive sampling of Written American English, displaying language use in a very wide range of styles and settings, and compiled from many different domains (Black et al. 1996). Since the ATR corpus is still in the process of development, the size of the texts we have at hand for this experiment is rather minimal considering the large size of the tag set. Table 1 shows the sizes of texts used for the experiment. Figure 4 shows the tagging error rates plotted against various clustering text sizes. Out of the three types of questions, basic questions and word bits

Text Size (words)	Training	Test	Held-Out
WSJ Text	75,139	5,831	6,534
ATR Text	76,132	23,163	6,680

Table 1: Texts for Tagging Experiments

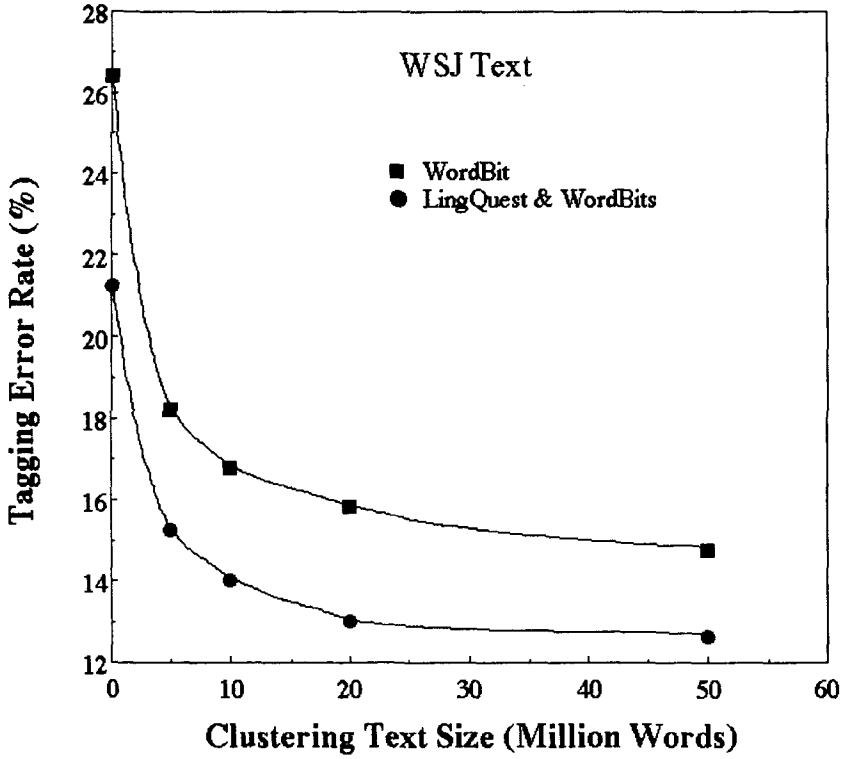

Figure 5: Comparison of WordBits with LingQuest & WordBits

questions are used in this experiment. To see the effect of introducing word bits information into the tagger, we performed a separate experiment in which a randomly generated bit-string is assigned to each word[2] and basic questions and word bits questions are used. The results are plotted at zero clustering text size. For both WSJ texts and ATR corpus, the tagging error rate dropped by more than 30% when using word bits information extracted from the 5MW text, and increasing the clustering text size further decreases the error rate. At 50MW, the error rate drops by 43%. This shows the improvement of the quality of clusters with increasing size of the clustering text. Overall high error rates are attributed to the very large tag set and the small training set. One notable point in this result is that introducing word bits constructed from WSJ texts is as effective for tagging ATR texts as it is for tagging WSJ texts even though these texts are from very different domains. To that extent, the obtained hierarchical clusters are considered to be portable across domains.

Figure 5 contrasts the tagging results using only word bits against the results with both word bits and linguistic questions[3] for the WSJ text. The zero clustering text size again corresponds

[2]Since a distinctive bit-string is assigned to each word, the tagger also uses the bit-string as an ID number for each word in the process. In this control experiment bit-strings are assigned in a random way, but no two words are assigned the same word bits. Random word bits are expected to give no class information to the tagger except for the identity of words.

[3]The linguistic questions we used here are still in the initial stage of development and are by no means

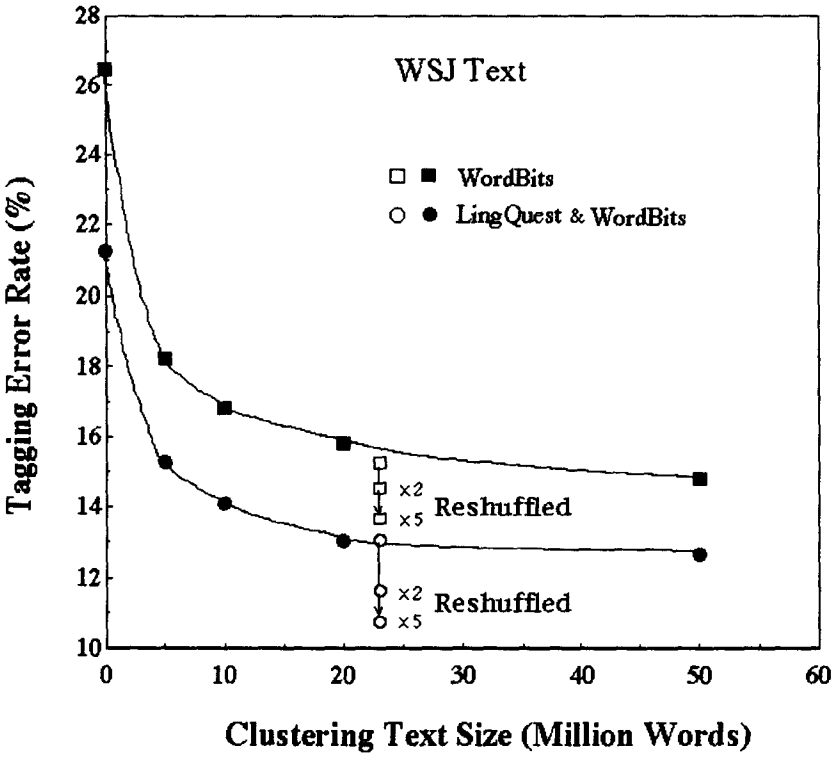

Figure 6: Effects of Reshuffling for Tagging

to a randomly generated bit-string. Introduction of linguistic questions is shown to significantly reduce the error rates for the WSJ corpus. Note that the dependency of the error rates on the clustering text size is quite similar in the two cases. This indicates the effectiveness of combining automatically created word bits and hand-crafted linguistic questions in the same platform, i.e., as features. In Figure 5 the tagging error rates seem to be approaching saturation after the clustering text size of 50MW. However, whether no further improvement can be obtained by using texts of greater size is still an unsolved question.

3.2 Reshuffling

One way to improve the quality of word bits is to introduce a reshuffling process just after step 1 (*MI-clustering*) of the word bits construction process (cf. § 2.2). The reshuffling process we adopted is quite simple.

1. Pick a word from the vocabulary. Move the word from its current class to another class if that movement increases the AMI most among all the possible movements.

2. Repeat step 1 starting from the most frequent word through the least frequent word.

This constitutes one *round* of reshuffling. After several rounds of reshuffling, the word bits construction process is resumed from step 2 (*Outer-clustering*).

Figure 6 shows the tagging error rates with word bits obtained by zero, two and five rounds of reshuffling[4] with a 23MW text. Tagging results presented in Figure 5 are also shown as a reference. Although the vocabulary used in this experiment is slightly different from the other

comprehensive.

[4]The vocabulary used for the reshuffling experiment shown in Figure 6 is the one used for a preliminary experiment and its size is 63850.

experiments, we can clearly see the effect of reshuffling for both the word-bits-only case and the case with word bits and linguistic questions. After five rounds of reshuffling, the tagging error rates become much smaller than the error rates using the 50MW clustering text with no reshuffling. It is yet to be determined if the effect of reshuffling increases with increasing amount of clustering text.

4 From Word Clustering to Compound Clustering

We showed in section 3 that the clusters we obtained are useful for Part-Of-Speech tagging. However, the clusters we have worked on so far have all been clusters of words, and the Part-Of-Speech tagging task has been limited to individual words. For many other NLP tasks, however, similarities among phrases or multiword compounds are more important than those among individual words. Let's turn back to the motivation of the clustering work discussed in the *Introduction*. Consider the following sentences.

(e) The music sent Mary to sleep.

(f) The music sent Professor Frederic K. Thompson to sleep.

Suppose that we want to translate sentence (f) to some language by an example-based machine translation system with example data including sentence (e) and its translation. In this case, what the system has to detect is that both "Mary" and "Professor Frederic K. Thompson" represent a human. The similarity between "Mary" and "Frederic" as being first names doesn't help in this case. Similarly, the detection of a correspondence between "CBS Inc." and "American Telephone & Telegraph Co." might be necessary in another case. This observation leads us to construct *classes of compounds* rather than classes of just words. Individual words can also be in the same class as multiword compounds, but we will generically call such a class a *class of compounds* in this paper. While several methods have been proposed to automatically extract compounds (Smadja 1993, Su et al. 1994), we know of no successful attempt to automatically make classes of compounds.

The obvious problem we face when we construct classes of compounds is that the possible number of compounds is too large if we try to handle them individually. However, if we represent compounds by a series of word-classes[5] instead of a series of words, we can constrain the explosion of the number of compounds. One way of looking at this approach is to bundle quite similar compounds in a small subclass and treat them as a single compound. For example, in the experiment described in Section 3, it was found that some word class, say WC129, contains almost exclusively first names, and another class, say WC246, contains almost exclusively family names. Then the chain of classes "WC129_WC246" represents one pattern of human names, or one group of two-word compounds representing human names. There are of course many other patterns, or class chains, of different lengths which represent human names. Therefore, our aim is to collect all the different class chains which are syntactically and semantically similar and put them in one compound-class.

In the following subsection, we describe one approach to this goal which is completely automatic.

4.1 Compound Clustering Method

Our compound clustering method consists of the following three steps.

1. Identification of Class Chains

   First, we replace each word in a large text with its word-class. We then use mutual information as a measure of "stickiness" of two classes, and identify which class pair

[5] We use the term *word-class* for a class of words to make a clear distinction from a *compound-class*.

should be chained. Let MI(C1,C2) be mutual information of adjacent classes C1 and C2 in the text. Then we form chain "C1_C2" if

$$MI(C1, C2) = \log \frac{Pr(c_1 c_2)}{Pr(c_1)Pr(c_2)} \geq *TH* \quad (3)$$

for some threshold *TH*.

If it is found, in the series of three classes "C1 C2 C3" in the text, that (C1,C2) forms a chain and (C2,C3) also forms a chain, then we simply form one large chain C1_C2_C3. In a similar way we form a chain of maximum length for any series of classes in the text.

2. Construction of Reduced Text and New Vocabulary

   Each class chain identified is then replaced in the text with a token which represents the chain. We call such a token a class chain token. After the scan through the text with this replacement operation of a class chain with its token, the text is represented by a series of word-classes and class chain tokens. The word classes remaining in the text are the ones which don't form a chain in their context. Those word classes are then converted back to their corresponding words in the text [6].

   The resulting text is the same as the original text except that a multiword compound which matches one of the extracted class chains is represented by a class chain token. We call this text the *reduced text.* Out of the reduced text, a new vocabulary is created as a set of words and tokens whose frequency in the reduced text is more than or equal to some threshold.

3. Compound Clustering

   We conduct *MI-clustering* (step 1 of the word bits construction process) using the reduced text and the new vocabulary. The classes we obtained , which we call compound-classes, contain words and class chain tokens. Each class chain token in a compound-class is then *expanded.* This means that all the multiword compounds that are represented by the class chain token in the text are put into the compound-class. After expanding all the tokens, the tokens are removed from the compound-classes. This results in compound-classes containing words and multiword compounds. It is also possible to construct hierarchical clustering of compounds if we follow all the steps in the word bits construction process after this step.

4.2 Compound Clustering Experiment

We used plain texts from two years (1987 and 1988) of Wall Street Journal Corpus to create compound clusters. The total volume amounts to about 40 MW of text. The word-classes used in this experiment are taken from the result of MI clustering with the 50MW text followed by five rounds of reshuffling. The quality of the compound clusters depends on the threshold *TH* in equation 3. We used *TH*=3 following "a very rough rule of thumb" used for word-based mutual information in (Church and Hanks, 1990).

Out of the 40MW text, 7621 distinct class chains and 287,656 distinct multiword compounds are extracted. To construct a new vocabulary, we selected the words and tokens whose frequency in the reduced text is more than four. The size of the new vocabulary is 60589 and it contains 4685 class chain tokens. Some of the compound-classes that were obtained are shown in Figure 7. The compounds are listed in descending order of frequency in each class, and the lists are truncated at an arbitrary point.

[6] The conversion of a word-class to a word is not a one-to-one mapping, but with the context in the text the conversion is unique. In the actual implementation, the text is represented by a series of (word, word-class) pairs and no conversion is actually carried out.

Figure 7: Examples of Compound Classes

COMPOUND CLASS 171:
President_Reagan Mr._Reagan Mr._Bush Mr._Dukakis Judge_Bork Ronald_Reagan George_Bush Michael_Dukakis Treasury_Secretary_James_Baker Mr._Holmes Vice_President_George_Bush Gov._Dukakis Gen._Noriega Mrs._Thatcher someone_who Mrs._Aquino Mr._Roh Gen._Secord Mr._Lawson Adm._Poindexter anyone_who Mr._Dole Lt._Col._North Jimmy_Carter Sen._Dole Mr._Mulroney Mr._Quayle Sen._Bentsen Mr._Chirac Mr._Gephardt Mr._Marcos Vice_President_Bush Sen._Quayle Mr._Carter Mr._Chun Prime_Minister_Margaret_Thatcher Judge_Greene Mr._Brady President_Carter President_Chun Judge_Kennedy Sen._Proxmire Robert_Bork Rep._Rostenkowski Mr._Kohl Robert_Holmes Judge_Pollack Mr._Kemp Prime_Minister_Yasuhiro_Nakasone Mr._Kennedy President_Aquino

COMPOUND CLASS 179:
General_Motors_Corp. Drexel_Burnham_Lambert_Inc. Ford_Motor_Co. International_Business_Machines_Corp. General_Electric_Co. Shearson_Lehman_Brothers_Inc. Chrysler_Corp. First_Boston_Corp. Merrill_Lynch_&_Co. Morgan_Stanley_&_Co. Shearson_Lehman_Hutton_Inc. News_Corp. American_Telephone_&_Telegraph_Co. PaineWebber_Inc. Prudential-Bache_Securities_Inc. Texaco_Inc. McDonnell_Douglas_Corp. Dean_Witter_Reynolds_Inc. Time_Inc. AMR_Corp. CBS_Inc. American_Express_Co. Campeau_Corp. BankAmerica_Corp. Du_Pont_Co. Allegis_Corp. General_Dynamics_Corp. Digital_Equipment_Corp. Kohlberg_Kravis_Roberts_&_Co. Exxon_Corp. Chase_Manhattan_Corp. USX_Corp. Nikko_Securities_Co. Lockheed_Corp.

COMPOUND CLASS 221:
common_stock preferred_stock cash_flow bank_debt long-term_debt foreign_debt subordinated_debt senior_debt balance_sheet short-term_debt balance_sheets cost_overruns corporate_debt debt_load convertible_preferred_stock international_debt debt_outstanding Class_B_stock debt_ratings cumulative_preferred_stock corporate_IOUs current_delivery preference_stock ozone_layer buffer_stock unsecured_debt convertible_preferred external_debt debt_offering current_contract blood_clots Class_B_common cumulative_convertible_preferred_stock corporate_governance Class_B_common_stock unsold_balance secured_debt debt_issue cumulative_preferred municipal_debt convertible_exchangeable_preferred_stock cash_hoard debt_rating 65-day_supply cash_balance senior_subordinated_debt senior_secured_debt

COMPOUND CLASS 256:
Fed SEC Reagan_administration IRS Pentagon Justice_Department Navy Commerce_Department FDA Army FCC FDIC Federal_Reserve_Board State_Department Bundesbank EPA FAA IMF Labor_Department Agriculture_Department FBI NASD Defense_Department Federal_Home_Loan_Bank_Board British_government NRC Finance_Ministry Japanese_government FTC UAW Kremlin PRI Transportation_Department PLO Federal_Trade_Commission CFTC Canadian_government NSC GAO Teamsters Carter_Hawley INS GSA Environmental_Protection_Agency ANC Labor_Party AFL-CIO FASB NFL Federal_Aviation_Administration ACLU

Compound-class-171 consists of names with title many of which are politicians' names. Compound-class-179 contains multiword company names. Compound-class-221 consists of multiword compound nouns from several specific semantic domains including money, surgery and natural environment, but most of the frequent compounds are money-related terms. Compound-class-256 is worth special attention in the sense that although single words and multiword compounds are mixed almost evenly, most of the single words are abbreviations of organizations, mostly public organizations, and the multiword compounds also almost exclusively represent public organizations. Another point to note here is that the pattern of *case* is not uniform in this list. Although both "Defense Department" and "British government" represent political organizations, the former consists of only capitalized words and the latter doesn't.

In order to measure the performance of this compound clustering method, a consistency check is performed for one class. The objective is to check what proportion of the identified members of the class actually *deserves* to be included in the class. Because this kind of judgement is very difficult in general, we must choose a class whose membership is quite clear to identify. By this criterion we chose compound-class-179 because it is quite easy to decide if some compound is a correct company name or not. From the 40MW text, we randomly chose 3000 occurrences of multiword compounds which are members of compound-class-179. By manual analysis, it was found that 133 identified compounds were wrong. The precision is therefore 95.6 %. Most of the errors are due to the truncation of correct company names. For example, from the string "North American Philips Corp.", only "Philips Corp." was extracted. Although "Philips Corp." is itself a correct company name, we treated this instance as an error because our judgement was occurrence-based. There was only one instance where a compound irrelevant to company names was extracted (a person name). For a control experiment which we will describe shortly, all the incorrect compounds are corrected by hand and a standard file is created which contains all the correct 2999 occurrences of company names.

One merit of the current approach is that the identification of a compound-class is carried out in time linear with the text size. Therefore, by associating a word with its word-class as a feature in the lexicon, and by storing class chain patterns and their membership to compound classes, we can carry out a real time identification of the compound-classes without actually storing the compounds in the lexicon.

As a control experiment to the above experiment, we conducted word-based compound extraction and compared the result with the above result. Instead of mutual information of adjacent classes, mutual information of adjacent words are calculated for all the bigrams in the text. Then using various MI threshold values, words are *chained* in a similar way as described in Section 4.1, and compounds are identified. We then evaluated how many of the occurrences of company names in the standard file are identified in the word-based compound extraction experiment. We varied the MI threshold values from 1.0 to 6.0 with a step of 0.5, but the precision of the word-based approach with respect to the standard file was always below 50 %.

The main reason of the superiority of the class-based approach against the word-based one is associated with the data sparseness problem. Most of the previously proposed methods to extract compounds or to measure word association using mutual information (MI) either ignore or penalize items with low co-occurrence counts (Church and Hanks 1990, Su, Wu and Chang 1994), because MI becomes unstable when the co-occurrence counts are very small. Take for example a class chain "WC129_WC246" discussed above. Figure 8 shows some examples of compounds matching the pattern "WC129_WC246" in the 40MW text. Each column shows, from left to right, word-based MI for the word bigram (WORD-1,WORD-2), co-occurrence frequency of the word bigram, the first word, the second word, class-based MI for the class bigram (CLASS-1, CLASS-2), co-occurrence frequency of the class bigram, the word-class of WORD-1, and the word-class of WORD-2. Note that the numbers for class-based entries are

Figure 8: Examples of Compounds for Names

WORD-MI	BI-COUNT	WORD-1	WORD-2	CLASS-MI	CL-BI-COUNT	CLASS-1	CLASS-2
16.104915	1	Takako	Doi	3.941235	52087	129	246
15.881772	3	Mandy	Patinkin	3.941235	52087	129	246
14.783159	3	Hideo	Sakamaki	3.941235	52087	129	246
12.280086	10	Curt	Bradbury	3.941235	52087	129	246
11.048669	3	Matthew	Kennelly	3.941235	52087	129	246
9.358209	1	Marsha	Gardner	3.941235	52087	129	246
7.994606	7	Ralph	Whitehead	3.941235	52087	129	246
5.073718	1	George	Hartman	3.941235	52087	129	246
4.328457	1	Daniel	Owen	3.941235	52087	129	246
3.914939	3	Charles	Walker	3.941235	52087	129	246
3.319351	1	Robert	Fischer	3.941235	52087	129	246
2.939145	1	Robert	Lucas	3.941235	52087	129	246
2.236354	2	Edward	Baker	3.941235	52087	129	246
1.119861	1	Robert	Shultz	3.941235	52087	129	246
1.072005	1	Robert	Hall	3.941235	52087	129	246
1.069133	1	George	Jackson	3.941235	52087	129	246
0.771154	1	Richard	Baker	3.941235	52087	129	246
0.218531	1	John	Jackson	3.941235	52087	129	246

the same for all the compounds because we collected compounds with the same class chain. Although all the compounds are compounds of a first name and a family name, the word-based MI varies considerably. This is because frequencies of first names and family names vary considerably while frequencies of pairs of first names and family names in the list are very small. For example, "John" and "Jackson" are very common first and second names, but the name "John Jackson" appeared only once in the text. Therefore the word-based MI becomes very small. On the other hand, because "Takako" and "Doi" were very rare names in WSJ news articles in 1987 and 1988, the MI becomes very high even though "Takako Doi" appeared only once in the text. In contrast, the class-based MI is very stable because the co-occurrence frequency of the two classes is as high as 52087. When we examined frequencies of all the compounds in the text that match "WC129_WC246", it turned out that more than 80 % of the compounds appeared less than five times in the text. This shows how the data sparseness problem is critical for the purpose of compound extraction.

5 Conclusion

We presented an algorithm for hierarchical clustering of words, and conducted a clustering experiment using large texts ranging in size from 5MW to 50MW. High quality of the obtained clusters is confirmed by the effect of introducing word bits into the ATR Decision-Tree Part-Of-Speech Tagger. The hierarchical clusters obtained from WSJ texts are also shown to be useful for tagging ATR texts which are from quite different domains than WSJ texts. The word-classes thus obtained are then used to identify and cluster multiword compounds. It is shown that by using statistics on classes instead of on words, the data sparseness problem is avoided and the reliability of mutual information is increased. As a result, class-based compounds identification and extraction becomes more reliable than word-based methods. This approach

also provides a way of automatically clustering compounds, which has rarely been attempted.

Acknowledgements

We thank John Lafferty and Christopher Manning for their helpful comments, suggestions and discussion with us. Special thanks are to Eric Visser for reviewing a draft of this paper.

References

Brill, E. and Marcus, M. (1992) "Automatically Acquiring Phrase Structure Using Distributional Analysis." *Darpa Workshop on Speech and Natural Language*, Harriman, N.Y.

Black, E., Eubank, S., Kashioka, H., Magerman, D., Garside, R., and Leech, G. (1996) "Beyond Skeleton Parsing: Producing a Comprehensive Large-Scale General-English Treebank With Full Grammatical Analysis". *Proceedings of the 16th International Conference on Computational Linguistics.*

Brown, P., Della Pietra, V., deSouza, P., Lai, J., Mercer, R. (1992) "Class-Based n-gram Models of Natural Language". *Computational Linguistics*, Vol. 18, No 4, pp. 467–479.

Church, K. and Hanks, P. (1990) "Word Association Norms, Mutual Information, And Lexicography". *Computational Linguistics*, Vol. 16, No 1, pp. 22–29.

Harris, Z. (1951) *Structural Linguistics.* Chicago, University of Chicago Press.

Jardino, M. and Adda, G. (1991) "Automatic word classification using simulated annealing", *ICASSP 91.*

Martin, M., Liermann, J. and Ney, H. (1995) "Algorithms for Bigram and Trigram Word Clustering", *Proceedings of European Conference on Speech Communication and Technology.*

Kneser, R. and Ney, H. (1993) "Improved Clustering Techniques for Class-Based Statistical Language Modelling". *Proceedings of European Conference on Speech Communication and Technology.*

Magerman, D. (1994) *Natural Language Parsing as Statistical Pattern Recognition.* Doctoral dissertation. Stanford University, Stanford, California.

Smadja, D. (1993) "Retrieving Collocations From Text: Xtract". *Computational Linguistics*, Vol. 19, No 1, pp. 143–177.

Su K.-Y., Wu M.-W. and Chan J.-S. (1994) "A Corpus-based Approach to Automatic Compound Extraction". *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics.*

Ueberla, J. (1995) "More Efficient Clustering of N-Grams for Statistical Language Modeling". *Proceedings of European Conference on Speech Communication and Technology.*

Ushioda, A. (1996) "Hierarchical Clustering of Words". *Proceedings of the 16th International Conference on Computational Linguistics.*