# Hierarchical Clustering of Words

**Akira Ushioda***
ATR Interpreting Telecommunications Research Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, Japan 619-02
email: ushioda@itl.atr.co.jp

## Abstract

This paper describes a data-driven method for hierarchical clustering of words in which a large vocabulary of English words is clustered bottom-up, with respect to corpora ranging in size from 5 to 50 million words, using a *greedy* algorithm that tries to minimize average loss of mutual information of adjacent classes. 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. Introducing word bits into the ATR Decision-Tree POS Tagger is shown to significantly reduce the tagging error rate. Portability of word bits from one domain to another is also disscussed.

## 1 Introduction

One of the fundamental issues concerning corpus-based NLP is the data sparseness problem. In view of the effectiveness of class-based n-gram language models against the data sparseness problem (Kneser and Ney 1993), 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.

Furthermore, clustering is much more useful if the clusters are of variable granularity, or hierarchical. We will consider 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 describe a method of creating binary tree representation of words and present results of evaluating and comparing the quality of the hierarchical clusters obtained from texts of very different sizes.

*Current address: Media Integration Laboratory, Fujitsu Laboratories Ltd., Kawasaki, Japan. Email: ushioda@flab.fujitsu.co.jp.

## 2 Word Bits Construction

Our word bits construction algorithm is a modification and an extension of the mutual information clustering algorithm proposed by Brown et al. (1992). We will first illustrate the difference between the original formulae and the ones we used, and then introduce the word bits construction algorithm. We will use the same notation as in Brown et al. to make the comparison easier.

### 2.1 Mutual Information Clustering Algorithm

Mutual information clustering method employs a bottum-up merging procedure with the average mutual information (AMI) of adjacent classes in the text as an objective function. In the initial stage, each word in the vocabularly of size $V$ 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. By storing the result of all the trial merges at the previous merging step, however, the time complexity can be reduced to $O(V^3)$ as shown below.

Suppose that, starting with V classes, we have already made V - k merges, leaving k classes, $C_k(1)$, $C_k(2)$, .. , $C_k(k)$. The AMI at this stage is given by the following equations.

$$I_k = \sum_{l,m} q_k(l,m) \tag{1}$$

$$q_k(l,m) = p_k(l,m) \log \frac{p_k(l,m)}{pl_k(l)pr_k(m)} \tag{2}$$

where $p_k(l,m)$ is the probability that a word in $C_k(l)$ is followed by a word in $C_k(m)$, and

$$pl_k(l) = \sum_m p_k(l,m), \quad pr_k(m) = \sum_l p_k(l,m).$$

In equation 1, $q_k$'s are summed over the entire $k \times k$ class bigram table in which (l,m) cell represents $q_k(l,m)$. In this merging step we investigate a trial merge of $C_k(i)$ and $C_k(j)$ for all class pairs $(i,j)$, and compute the AMI reduction $L_k(i,j) \equiv I_k - I_k(i,j)$ effected by this merge, where $I_k(i,j)$ is the AMI after the merge.

Suppose that the pair $(C_k(i), C_k(j))$ was chosen to merge, that is, $L_k(i,j) \leq L_k(l,m)$ for all

pairs $(l,m)$. In the next merging step, we have to calculate $L_{k-1}^{(i,j)}(l,m)$ for all the pairs $(l,m)$. Here we use the superscript $(i,j)$ to indicate that $(C_k(i), C_k(j))$ was merged in the previous merging step.

Now note that the difference between $L_{k-1}^{(i,j)}(l,m)$ and $L_k(l,m)$ only comes from the terms which are affected by merging the pair $(C_k(i), C_k(j))$.

Since $L_k(l,m) = I_k - I_k(l,m)$ and $L_{k-1}^{(i,j)}(l,m) = I_{k-1}^{(i,j)} - I_{k-1}^{(i,j)}(l,m)$, we have

$$L_{k-1}^{(i,j)}(l,m) - L_k(l,m) = \\ -(I_{k-1}^{(i,j)}(l,m) - I_k(l,m)) + (I_{k-1}^{(i,j)} - I_k).$$

Some part of the summation region of $I_{k-1}^{(i,j)}(l,m)$ and $I_k$ cancels out with a part of $I_{k-1}^{(i,j)}$ or a part of $I_k(l,m)$. Let $\hat{I}_{k-1}^{(i,j)}(l,m)$, $\hat{I}_k(l,m)$, $\hat{I}_{k-1}^{(i,j)}$, and $\hat{I}_k$ denote the values of $I_{k-1}^{(i,j)}(l,m)$, $I_k(l,m)$, $I_{k-1}^{(i,j)}$ and $I_k$, respectively, after all the common terms among them which can be canceled are canceled out. Then, we have

$$L_{k-1}^{(i,j)}(l,m) - L_k(l,m) = \\ -(\hat{I}_{k-1}^{(i,j)}(l,m) - \hat{I}_k(l,m)) + (\hat{I}_{k-1}^{(i,j)} - \hat{I}_k), \quad (3)$$

where

$$\begin{aligned} \hat{I}_{k-1}^{(i,j)}(l,m) &= q_{k-1}(l+m,i) + q_{k-1}(i,l+m) \\ \hat{I}_k(l,m) &= q_k(l+m,i) + q_k(i,l+m) + \\ &\quad q_k(l+m,j) + q_k(j,l+m) \\ \hat{I}_{k-1}^{(i,j)} &= q_{k-1}(i,l) + q_{k-1}(i,m) + \\ &\quad q_{k-1}(l,i) + q_{k-1}(m,i) \\ \hat{I}_k &= q_k(i,l) + q_k(i,m) + q_k(j,l) + \\ &\quad q_k(j,m) + q_k(l,i) + q_k(l,j) + \\ &\quad q_k(m,i) + q_k(m,j) \end{aligned}$$

Because equation 3 is expressed as the summation of a fixed number of q's, its value can be calculated in constant time, whereas the calculation of equation 1 requires $O(V^2)$ time. Therefore, the total time complexity is reduced by $O(V^2)$.

The summation regions of $\hat{I}$'s in equation 3 are illustrated in Figure 1. Brown et al. seem to have ignored the second term of the right hand side of equation 3 and used only the first term to calculate $L_{k-1}^{(i,j)}(l,m) - L_k(l,m)$ [1]. However, since the second term has as much weight as the first term, we used equation 3 to make the model complete.

Even with the $O(V^3)$ algorithm, the calculation is not practical for a large vocabulary of order $10^4$ or higher. Brown et al. proposed the following

[1] Actually, it is the first term of equation 3 times (-1) that appeared in their paper, but we believe that it is simply due to a misprint.

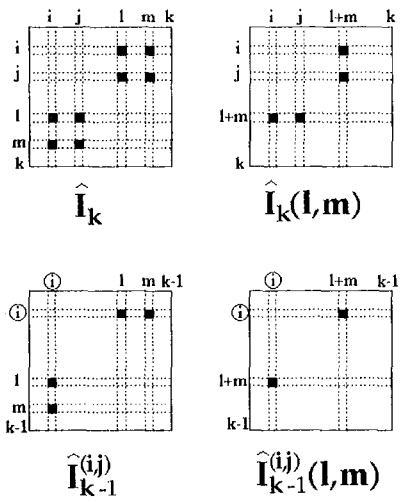


Figure 1: Summation Regions for $\hat{I}$'s

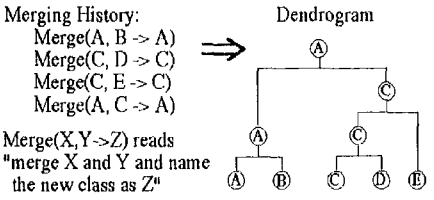


Figure 2: Dendrogram Construction

method, which we also adopted. We first make V singleton classes out of the V words in the vocabulary and arrange the classes in descending order of frequency, then define the *merging region* as the first $C+1$ positions in the sequence of classes. At each merging step, merging of only the classes in the merging region is considered, thus reducing the number of trial merges from $O(V^2)$ to $O(C^2)$. After each actual merge, the most frequent singleton class outside of the merging region is shifted into the region. With this algorithm, the time complexity is reduced to $O(C^2V)$.

### 2.2 Word Bits Construction Algorithm

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

1. *MI-clustering*: Make C classes using the mutual information clustering algorithm with the merging

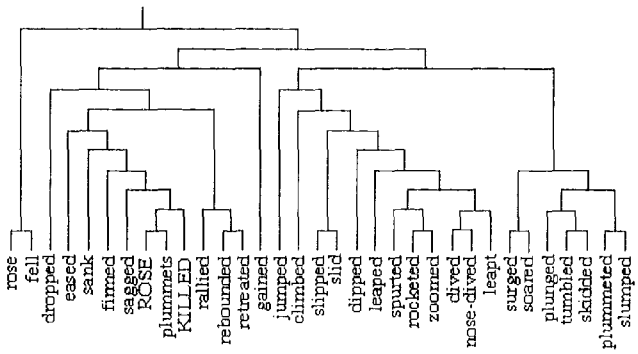


Figure 3: Sample Subtree for One Class

region constraint mentioned in (2.1).
2. *Outer-clustering*: Replace all words in the text with their class token[2] and execute binary merging without the merging region constraint until all the classes are merged into a singe 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.
(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 coresponding $D_{sub}$.

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 3 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.

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

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)
{ IsPrefix?(Word(0), "anti"), "no" }
. . . . . (Linguist's Questions)
{ Tag, "Prep-type5" } }

Figure 4: Example of an event

## 3 Experiments

We used plain texts from six years of the WSJ Corpus to create 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 which is based on SPATTER (Magerman 1994). The tagger employs a set of 443 syntactic tags. 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. Figure 4 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 three lines show questions about identity of words around the current word and tags for previous words. These questions are called *basic questions* and always used. The second type of questions, *word bits questions*, are on clusters and word bits such as *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.

Out of the set of events, a decision tree is constructed whose leaf nodes contain conditional probability distributions of tags, conditioned by the feature values. 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 (Black et al. 1996) for the tagging experiment. Both corpora use the ATR syntactic tag set. 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 5 shows the tagging error rates plotted against various clustering

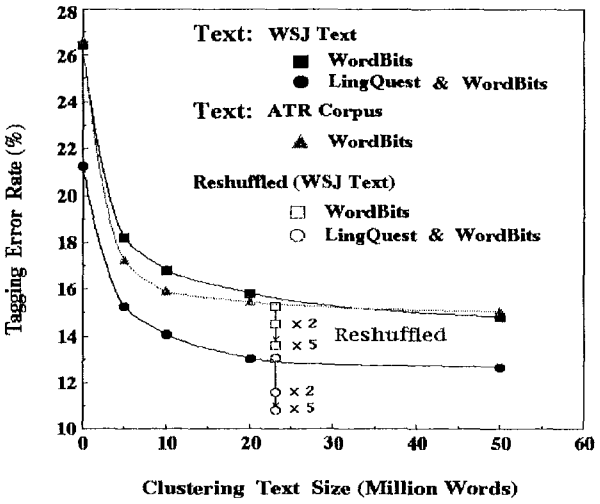


Figure 5: Tagging Error Rate

| 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

text sizes. Out of the three types of questions, basic questions and word bits questions are always 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[3] 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 the hierarchical clusters with increasing size of the clustering text. In Figure 5, introduction of linguistic questions[4] is also shown to significantly reduce the error rates for the WSJ corpus. The dependency of the error rates on the clustering text size is quite similar to the case in which no linguistic questions are used, indicating the effectiveness of combining automatically created word bits and handcrafted linguistic questions. Figure 5 also shows that reshuffling the classes several times just after step 1 (*MI-clustering*) of the word bits construction process further improves the word bits. One round of reshuffling corresponds to moving each word in the vocabulary from its original class to another class whenever the movement increases the AMI, starting from the most frequent word through the least frequent one. The figure shows the error rates with zero, two, and five rounds of reshuffling[5]. Overall high error rates are attributed to the very large tag set and the small training set. Another notable point in the figure 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.

[3]Since a distinctive bit-string is assigned to each word, the tagger also uses a 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.

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

## 4 Conclusion

We presented an algorithm for hierarchical clustering of words, and conducted a clustering experiment using large texts of varying sizes. High quality of the obtained clusters are confirmed by the POS tagging experiments. By introducing word bits into the ATR Decision-Tree POS Tagger, the tagging error rate is reduced by up to 43%. 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.

## Acknowledgements

We thank John Lafferty for his helpful suggestions.

## References

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.

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.

[5]The vocabulary used for the reshuffling experiments is the one used for a preliminary experiment and its size is 63850.