

Word and Compound Clustering for Natural Language Processing

Akira Ushioda

CMU-LTI-00-165

School of Computer Science
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee
Jaime Carbonell, Chair
John Lafferty, Co-chair
Robert Carpenter
Christopher Manning

Submitted in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy

©2000 by Akira Ushioda



School of Computer Science

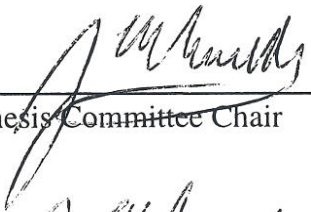
**Doctoral Thesis
in the field of
Language Technologies**

*Word and Compound Clustering
for Natural Language Processing*

Akira Ushioda

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy


ACCEPTED:



Thesis Committee Chair

9/22/00

Date




Department Head

9/22/00

Date

APPROVED:



Dean

Date

Abstract

This thesis addresses the problem of automatically constructing word clusters from plain texts. While the speech community has done much work on word clustering, their goal has been to improve language modeling and less attention has been paid to natural language processing tasks. Furthermore, in previous clustering work, the domain was often very limited and vocabulary size was small. (Some exceptions involve a series of works by the IBM group.)

This thesis explores the utility of clustering techniques to large scale NLP tasks with very large vocabularies. The criterion function employed in this work to measure the clustering quality is the cross-entropy of the empirical word distribution function with regard to the probability function of a class bigram language model. The main focus of this research is to show that the improvement of clusters in terms of cross-entropy is reflected in the improvement of NLP tasks, especially for rare events.

In order to support the above assumption, various methods to improve the clustering quality are investigated, and the resulting clusters are evaluated through practical NLP tasks including Part-Of-Speech tagging and proper name identification and classification.

Because rare events are very common in large scale NLP tasks, this work should be interesting to researchers in many NLP fields including corpus linguistics, broad-coverage terminology compilation and lexicography, text categorization and summarization, machine translation, as well as information retrieval and language modeling for speech recognition.

Acknowledgements

I am very grateful to my advisors: Jaime Carbonell, for his guidance, encouragement and enduring patience, and John Lafferty for his insight, valuable suggestions, and stimulating discussions.

I would like to thank Bob Carpenter (SpeechWorks International) and Chris Manning (Stanford University) for their encouragement and helpful suggestions and also for providing me working knowledge on many areas of Natural Language Processing while they were faculty members at Carnegie Mellon University.

I am grateful to Alex Waibel for helpful suggestions and discussions on statistical modelling and for giving me a great opportunity to work in his speech technology group.

I would also like to thank my former colleagues at CMU: Gerald Penn, Thomas Polzin, and Massimo Paolucci, for many valuable discussions and above all for making three years of my life at CMU very pleasant.

I would also like to thank Lauri Lafferty for closely reading a thesis draft with a special focus on correcting English expressions.

Finally, my deepest gratitude goes to my wife Akiyo for all her love, support, endless patience and encouragement. I dedicate this thesis to her.

Contents

1	Introduction	1
2	Cluster Analysis	5
2.1	Hierarchical Clustering	6
2.1.1	Agglomerative Hierarchical Clustering	7
2.1.2	Divisive Hierarchical Clustering	9
2.2	Optimization Method	9
2.3	Soft Clustering	11
3	Word Clustering	13
3.1	Previous Work: Automatic Clustering of Words	13
3.2	Word Clustering for NLP Tasks	21
3.3	MI Clustering and Hierarchical Word Clustering	22
3.3.1	Mutual Information Clustering Algorithm	22
3.3.2	Hierarchical Word Clustering Algorithm	27
3.3.3	Example of Clusters	29
3.4	Improvement of Clustering Quality	34
4	Evaluation of Clustering Quality	45
4.1	Introduction	45
4.2	Decision-Tree Part-Of-Speech Tagger	46
4.3	HMM Tagger	50
4.3.1	Tagging Model	50
4.3.2	Experiments on the WSJ Corpus	52
4.4	Domain Dependence	55
4.5	Split & Merge Clustering	60
4.5.1	POS Tagging of Unknown Words	62
4.5.2	Number of Classes	63
4.6	Comparison with Previous Work on Part-Of-Speech Tagging	67
4.7	Conclusion	72
5	From Word Clustering to Compound Clustering	74
5.1	Introduction	74
5.2	Compound Clustering Method	74
5.3	Improvement of Compound Clusters	76
5.4	Evaluation of Compound Clusters	76

5.4.1 Compound Clustering Experiment	77
6 Conclusion	87

List of Figures

2.1	Example of Dendrogram	6
3.1	Summation Region (1)	24
3.2	Summation Region (2)	24
3.3	Summation Region (3)	26
3.4	Dendrogram Construction	27
3.5	Sample Subtree for One Class	29
3.6	Distribution of Class Sizes	31
3.7	Distribution of Word Bits Length	33
3.8	Example Subtrees	35
3.9	Reduction of Average Mutual Information	41
4.1	Example of an Event	47
4.2	Tagging Error Rate	48
4.3	Comparison of WordBits with LingQuest & WordBits	49
4.4	Effects of Reshuffling for Tagging	50
4.5	POS Tagging Accuracy for Unknown Words with Varied Training Sizes and Clusters	53
4.6	POS Tagging Accuracy for All Words with Varied POS-Tagged Training Text Sizes	55
4.7	POS Training Text Size Dependencies	56
4.8	POS Tagging Accuracy for Unknown Words with AP-Clusters	58
4.9	Domain Dependency	59
4.10	Threshold Value vs. Number of Classes	61
4.11	Example of Split&Merge Clustering Performance	64
4.12	AMI of AP Newswire with WSJ Clusters	65
4.13	POS Tagging Performance for Unknown Words	66
4.14	POS Tagging Accuracy with Varying Cluster Sizes	68

List of Tables

2.1	Comparison of Clustering Methods	6
3.1	Large Classes	30
3.2	Examples of Singleton Classes	31
3.3	Examples of Good Classes	32
3.4	Examples of Bad Classes	33
3.5	Example of Misspelled Words	34
3.6	Example of Hyphenated Words	36
3.7	Example of Data Size Dependency (1)	38
3.8	Example of Data Size Dependency (2)	39
3.9	Reshuffling	40
3.10	Class Purification	42
3.11	Example of Crossvalidation	44
4.1	Texts for Tagging Experiments	47
4.2	Comparison of POS Tagging Accuracies	71
5.1	Compound Class 171 (high frequency part)	78
5.2	Compound Class 171 (low frequency part)	79
5.3	Compound Class 179	80
5.4	Compound Class 221	81
5.5	Compound Class 256	82
5.6	Examples of Compounds for Names	84

Chapter 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 one which guarantees mutual substitutability, in terms of both syntactic and semantic soundness, among words in the same class (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 (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 (b) 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 access to the classes of words. Therefore, it is expected that building clusters of the vocabulary in terms of *mutual substitutability* improves performance of NLP tasks. The mutual information (MI) clustering algorithm proposed by Brown et al. (1992) is a promising candidate for a way of constructing such word clusters from *plain texts*. Since the amount of plain texts available on-line nowadays is several orders of magnitude greater than that of human-annotated corpora, automatic learning methods from plain texts such as MI clustering are quite attractive *if* it is the case that the more texts we use the better results we obtain. This assumption, however, has yet to be investigated.

Furthermore, clustering should be much more useful if the clusters are of variable granularity. Suppose, for example, that we have two sets of clusters, one finer than the other, and that word-1 and word-2 are in different finer classes. With finer clusters alone, the amount of information about the association of the two words that the system can obtain from the clusters is minimal. However, if the system is capable of falling back and checking if they belong to the same coarser class, 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. We will call such multi-level clusters hierarchical clusters.

Hierarchical clustering is one direction to extend the conventional *flat* clustering (i.e. a simple partition of the vocabulary). Another important direction to extend the paradigm

of clustering is to move from word-based clustering to compound-based clustering (Ushioda 1996b). 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. Indeed, for many NLP tasks, similarities among phrases or multiword compounds are more important than those among individual words. Consider the following sentences.

(e) The music put Mary to sleep.

(f) The music put 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* or simply a *compound class*. 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 for large scale, broad coverage NLP, such as for processing news articles.

This thesis will explore the utility of the clustering technique to large scale NLP tasks with very large vocabularies. The criterion function used in this work to measure the clustering quality is the cross-entropy of the empirical word distribution function with regard to the probability function of a class bigram language model. The main focus of this research is to show that the improvement of clusters in terms of cross-entropy is reflected in the improvement of NLP tasks, especially for rare events. In order to show that, various methods to improve the clustering quality will be investigated, and the resulting clusters will be evaluated through practical NLP tasks including Part-Of-Speech tagging and proper name identification and classification.

The organization of the thesis is as follows. In Chapter 2, previous work on data-

driven automatic clustering methods will be reviewed. Chapter 3 first describes previous work on automatic clustering of words, then presents the basic framework of the word clustering methods investigated in this thesis work. Chapter 4 quantitatively evaluates cluster quality using Part-Of-Speech tagging as a representative example of NLP tasks to which the clustering technique is directly applied. Chapter 5 extends the paradigm of clustering from word-based clustering to compound-based clustering, and shows how word-based clusters can be used to efficiently create compound clusters. Chapter 6 summarizes the thesis and suggests future directions of this research.

Chapter 2

Cluster Analysis

This chapter reviews previous work on automatic clustering methods. So much work has been reported on clustering that it is out of the scope of this thesis to cover the whole range of clustering methods. Instead, we will give here a brief overview of two major types of clustering, namely hierarchical clustering and optimization, and another newer method called soft clustering which is designed to overcome a major defect of conventional clustering techniques. Cluster analysis has been widely used in many fields of study including biology, sociology, psychology, business, computer science, medicine, and many others. Some of the characteristics of each type of clustering method are summarized in Table 2.1. ⟨Partition?⟩ checks whether the method produces a partition of the data, that is, whether there is no overlap between classes. ⟨Revocable?⟩ designates whether the membership of an element to a class can be changed during the course of the clustering process. ⟨Global Criterion?⟩ checks whether a global clustering criterion which drives the clustering process must always be defined, where “global” means that every individual is involved in the calculation of the criterion.¹ In the table, “-” means either the answer is not fixed or the question is not applicable. Representative clustering methods of each type will be described in the following sections.

¹In optimization and soft clustering such a global criterion is always necessary, whereas in some hierarchical clustering methods local information is enough. For example, in the single link method, whether to merge two classes can be determined by comparing the local between-class distances for all the pairs of classes, and it is not necessary to define a global criterion.

	Hierarchical	Optimization	Soft
(a) ⟨Partition?⟩	YES	YES	NO
(b) ⟨Revocable?⟩	NO	YES	-
(c) ⟨Global Criterion?⟩	-	YES	YES

Table 2.1: Comparison of Clustering Methods

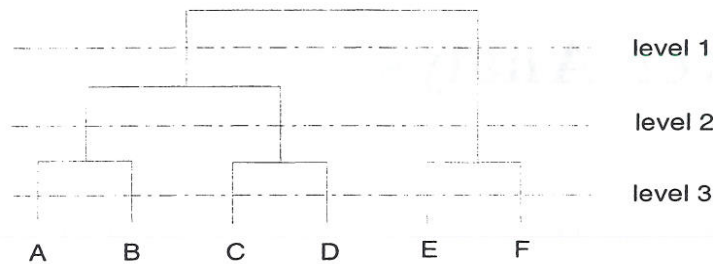


Figure 2.1: Example of Dendrogram

2.1 Hierarchical Clustering

Hierarchical clustering produces a multi-level partition of the data in which the whole entities are partitioned into subgroups, each of which is further partitioned into its subgroups, and each of these is further partitioned, and so on. A hierarchical cluster can be represented as a tree called dendrogram which shows how the entities are grouped.

Figure 2.1 shows a simple example of a dendrogram. At level-1 the whole entities are divided into two groups, or classes, $\{A,B,C,D\}$ and $\{E,F\}$. At level-2 there are three classes $\{A,B\}$, $\{C,D\}$, $\{E,F\}$, and at level-3 all individuals constitute a class of their own, or, a singleton.

Hierarchical clustering methods are classified into two types, agglomerative methods and divisive methods, depending on whether the clustering process is bottom-up or top-down. Agglomerative methods proceed by successive merges of the classes and divisive methods repeat partitions of classes into their subclasses. In both cases the process is irrevocable. This property enables the process to run fairly fast, but of course at the cost of resulting cluster quality since a poor early partition or merge decision cannot be modified in the later clustering steps.

2.1.1 Agglomerative Hierarchical Clustering

Agglomerative hierarchical clustering methods start with n singleton classes, where n is the total number of individuals to be clustered, and end with one class with n elements. The process of merging is based on computation of similarities (or dissimilarities) between two entities, where an entity is either an individual element or a class of individuals. Since an individual can be seen as a singleton class, the similarities are in general defined on a pair of classes. At each merging step, a pair of classes which are the most similar (or least dissimilar), according to the defined similarity measure, is identified, and these classes are merged to form a new class which replaces the two merged classes. Therefore, the resulting dendrogram is a binary tree. In general we can think of a process in which more than two classes are merged at once, but in practice the binary similarity measure is used almost exclusively, largely due to computational restrictions. Agglomerative hierarchical methods are classified according to the similarity measure adopted. Well-known methods of this type in the literature include the single link method, the complete link method, the centroid cluster method, the group average method, and Ward's method.

The single link method defines dissimilarity (or distance) between two classes as the distance between their most similar members (Sneath 1957). If individuals are represented as nodes of a graph with edges being formed between nodes in the same class, and an edge between the most similar pair of nodes in two classes C_i and C_j , then this method produces a minimum spanning tree. Besides the simplicity, this method has an advantage of invariance to monotonic transformations of the similarity data, which means that the clustering result is not affected by any data transformation that retains the relative order of similarity values between individuals. This method also possesses a well-known defect called *chaining* which is a tendency to prefer incorporating new entities into existing clusters to initiating new clusters. However, this very tendency can be advantageous if the clusters have an elongated shape in the feature space, because most other methods tend to erroneously chop elongated clusters.

The opposite of the single link method is the complete link method which defines the distance between two classes as the distance between their most dissimilar members (Sokal and Michener 1958).

Between the above two methods lie centroid cluster method and the group average method (Sokal and Michener 1958). The centroid cluster method defines the distance between two classes as the distance between the two class centroids. In this method it is assumed that each individual is represented by a vector \mathbf{v} of features. If i^{th} class consists of c_i elements $\mathbf{v}_1^i, \mathbf{v}_2^i, \dots, \mathbf{v}_{c_i}^i$, then its centroid $\bar{\mathbf{v}}^i$ is given by

$$\bar{\mathbf{v}}^i = \frac{1}{c_i} \sum_{k=1}^{c_i} \mathbf{v}_k^i \quad (2.1)$$

and the distance between two classes i and j is

$$d(i, j) = \| \bar{\mathbf{v}}^i - \bar{\mathbf{v}}^j \|$$

In the group average method the distance between two classes is defined as the average of the distances between all pairs of individuals in the two classes. That is,

$$d(i, j) = \frac{1}{c_i c_j} \sum_{m=1}^{c_i} \sum_{n=1}^{c_j} d(x_m^i, x_n^j),$$

where x_p^k is the p^{th} element of class k . Note that only distances between individuals need to be known to obtain the distance between classes; a co-ordinate of each individual need not be known or even defined. This is in contrast to the centroid cluster method. When co-ordinates or vectors of individuals are given, Euclidean distance can be used to define the distance between classes:

$$d(x_m^i, x_n^j) = \| \mathbf{v}_m^i - \mathbf{v}_n^j \|$$

In Ward's method, the sum of squared errors (SSE) E , given below, is introduced as a criterion of the quality of clusters (Ward 1963):

$$E = \sum_{i=1}^c \sum_{k=1}^{c_i} \| \mathbf{v}_k^i - \bar{\mathbf{v}}^i \|^2, \quad (2.2)$$

where c is the number of classes and $\bar{\mathbf{v}}^i$ is the centroid of class i defined in equation 2.1.

At each clustering step SSE is calculated and a pair of classes is merged so that merging the pair minimizes the increase in SSE.

2.1.2 Divisive Hierarchical Clustering

The first step in divisive hierarchical clustering methods is to divide the whole data set into two classes, followed by successive partition of classes into two subclasses. When there are n individuals, there are 2^{n-1} ways to split the whole set into two classes in the first step. Therefore it is not practical to consider all the possible partitions except in the case of a very small n . Two basic types of divisive methods are monothetic methods and polythetic methods.

Monothetic methods primarily deal with binary data. Each class is divided into two subclasses depending on whether the member possesses a specific feature or not. Association analysis is of this type. In association analysis a class is divided in terms of presence or absence of an appropriate feature for that class such that dissimilarity between the two subclasses is maximized in terms of the division criterion associated with that feature.

In polythetic methods decision rules for partitioning are based on the values of multiple features. The splinter group method proposed by MacNaughton-Smith et. al (1964) is of this type. In this method the most dissimilar element to the rest of the class is picked as the first element of the splinter group, and other elements which are more similar to the splinter group than the rest of the class are added to the splinter group one by one until moving the item begins to increase the predefined total cost.

In the minimum diameter clustering, a class diameter is defined to be the largest distance between any pair of individuals in the class, and the class with the largest diameter is divided into two subclasses so that the diameter of the larger of the two subclasses is as small as possible.

2.2 Optimization Method

Optimization methods produce a flat partition of the data. An initial cluster is created first, and then individuals are moved from class to class to optimize some predefined clustering criterion. Although optimization methods only construct a one-level partition of the data by itself, we can always create a hierarchical cluster by repeatedly applying optimization methods to each class.

Variations of the optimization methods come from variations in the type of optimiza-

tion criteria. One of the most widely used criteria is the sum of squared errors given in equation 2.2, and the k-means method (MacQueen 1967) is a representative of this kind. In the k-means method, the number c of classes is predefined and initial guesses are made for c class centroids. This can be done by a random choice or by more elaborate means. Then the following steps are repeated until there are no changes:

- assign each item to the nearest class centroid
- replace each class centroid with the mean of all the elements in the class

Some other well-known criteria can be derived from the scatter matrix commonly used in multiple discriminant analysis (Wilks 1962). Consider the problem of partitioning a set S of N individuals into c disjoint classes C_1, C_2, \dots, C_c . We assume that each item is represented by a p -dimensional feature vector $\mathbf{v} = \begin{pmatrix} v_1 \\ \dots \\ v_p \end{pmatrix}$. Let c_i be the number of elements in C_i and let $\bar{\mathbf{v}}^i$ be the mean of elements in C_i . The total mean vector $\bar{\mathbf{v}}$ is given by

$$\bar{\mathbf{v}} = \frac{1}{N} \sum_{\mathbf{v} \in S} \mathbf{v} = \frac{1}{N} \sum_{k=1}^c c_k \bar{\mathbf{v}}^k$$

Then the total scatter matrix \mathbf{T} of S is given by

$$\mathbf{T} = \sum_{\mathbf{v} \in S} (\mathbf{v} - \bar{\mathbf{v}})(\mathbf{v} - \bar{\mathbf{v}})^T$$

The total scatter matrix can be rewritten as a sum of two terms as follows:

$$\begin{aligned} \mathbf{T} &= \sum_{i=1}^c \sum_{\mathbf{v} \in C_i} ((\mathbf{v} - \bar{\mathbf{v}}^i) + (\bar{\mathbf{v}}^i - \bar{\mathbf{v}}))((\mathbf{v} - \bar{\mathbf{v}}^i) + (\bar{\mathbf{v}}^i - \bar{\mathbf{v}}))^T \\ &= \sum_{i=1}^c \sum_{\mathbf{v} \in C_i} (\mathbf{v} - \bar{\mathbf{v}}^i)(\mathbf{v} - \bar{\mathbf{v}}^i)^T + \sum_{i=1}^c c_i (\bar{\mathbf{v}}^i - \bar{\mathbf{v}})(\bar{\mathbf{v}}^i - \bar{\mathbf{v}})^T \\ &+ \sum_{i=1}^c \left(\sum_{\mathbf{v} \in C_i} (\mathbf{v} - \bar{\mathbf{v}}^i) \right) (\bar{\mathbf{v}}^i - \bar{\mathbf{v}})^T + \sum_{i=1}^c (\bar{\mathbf{v}}^i - \bar{\mathbf{v}}) \left(\sum_{\mathbf{v} \in C_i} (\mathbf{v} - \bar{\mathbf{v}}^i) \right)^T \\ &= \mathbf{W} + \mathbf{B}, \end{aligned} \tag{2.3}$$

where

$$\begin{aligned} \mathbf{W} &= \sum_{k=1}^c \mathbf{W}_k \\ \mathbf{W}_k &= \sum_{\mathbf{v} \in C_k} (\mathbf{v} - \bar{\mathbf{v}}^k)(\mathbf{v} - \bar{\mathbf{v}}^k)^T \end{aligned} \tag{2.4}$$

and

$$\mathbf{B} = \sum_{k=1}^c c_k (\bar{\mathbf{v}}^k - \bar{\mathbf{v}})(\bar{\mathbf{v}}^k - \bar{\mathbf{v}})^T$$

\mathbf{W}_k is the within-class scatter matrix of class k which is a measure of how much elements of k^{th} class are dispersed within the class. \mathbf{B} is the between-class scatter matrix. Since the total scatter \mathbf{T} does not depend on how the data are clustered, a desirable cluster is the one for which the within-class dispersion is minimum and/or the between-class dispersion is maximum. Several clustering criteria based on the scatter matrix have been proposed. One criterion is to minimize the trace of \mathbf{W} , which is the sum of the variation on each feature. Another simple scalar measure of the variability within classes is the determinant of the within-class scatter matrix. Detailed discussion on advantages and disadvantages of scatter-based optimization criteria is given in Duda and Hart (1973).

Another type of optimization methods is reshuffling. Starting with some initial cluster, increase in the optimization criterion by the reallocation of an individual from its current class to another class is calculated for each individual, and each individual is moved to another class so as to maximize the increase in the criterion.

2.3 Soft Clustering

In all of the methods described so far, it is assumed that any individual either is or is not a member of a particular class. This type of clustering is called *hard* or *crisp* clustering. In *soft* clustering, in contrast, each individual has a degree of membership in each class. The fuzzy-k-means method (Bezdek 1981), which is an extension of the k-means method, is an example of this type. The (hard) k-means method can be reformulated as the problem of minimizing the within-class sum-of-squared error criterion J_H in equation 2.5 under conditions 2.6 through 2.8:

$$J_H = \sum_{k=1}^n \sum_{i=1}^c u_{ik} \| \mathbf{v}_k - \bar{\mathbf{v}}^i \|^2 \quad (2.5)$$

$$\forall i, k \quad u_{ik} \in \{0, 1\} \quad (2.6)$$

$$\forall k \quad \sum_{i=1}^c u_{ik} = 1 \quad (2.7)$$

$$\forall i \quad 0 < \sum_{k=1}^n u_{ik} < n \quad (2.8)$$

$\mathbf{v}_1, \dots, \mathbf{v}_n$ are feature vectors of n individuals, $\mathbf{v}^1, \dots, \mathbf{v}^c$ are c class centroids, and u_{ik} corresponds to \mathbf{v}_k 's membership to class i . Conditions 2.6 and 2.7 imply that each individual belongs to one and only one class, and condition 2.8 ensures that there is no empty class.

In the fuzzy-k-means method, 2.5 and 2.6 are modified as follows:

$$J_S = \sum_{k=1}^n \sum_{i=1}^c u_{ik}^\psi \|\mathbf{v}_k - \mathbf{m}_i\|^2 \quad (2.9)$$

$$\forall i, k \quad u_{ik} \in [0, 1], \quad (2.10)$$

where $\psi \in (1, \infty)$ is a fuzzy exponent (Bezdek 1981), and \mathbf{m}_i is the fuzzy mean of class i . Note that the degree of class membership u_{ik} can take any real value between 0 and 1.

The necessary conditions for minimizing J_S are given by Bezdek (1981) as follows:

$$u_{ik} = \frac{\|\mathbf{v}_k - \mathbf{m}_i\|^{-\frac{2}{\psi-1}}}{\sum_{j=1}^c \|\mathbf{v}_k - \mathbf{m}_j\|^{-\frac{2}{\psi-1}}} \quad (2.11)$$

$$\mathbf{m}_i = \frac{\sum_{k=1}^n u_{ik}^\psi \mathbf{v}_k}{\sum_{k=1}^n u_{ik}^\psi} \quad (2.12)$$

Similar to the case of the hard k-means method, the optimum class centroids and membership functions can be obtained in the following iterations:

- obtain initial set of class centroids $\mathbf{m}_1, \dots, \mathbf{m}_c$
- repeat until changes in class centroids are less than some threshold value
 - update degrees of class membership using equation 2.11
 - update class centroids using equation 2.12

The fuzzy exponent ψ determines the degree of fuzziness (i.e., the degree of overlap between classes) of the final configuration.

Lee (1997) also proposed centroid-based soft clustering which uses *free energy* as a clustering criterion to create hierarchical clusters of words. Saul and Pereira (1997) and Rooth et al. (1999) formalized soft clustering of words in the framework of Expectation-Maximization (EM) algorithm. Details of these methods will be given in the next chapter.

Chapter 3

Word Clustering

This chapter first describes previous work on automatic clustering of words. Following that, the basic framework of the word clustering methods investigated in this thesis work is presented. Finally, examples of word clusters obtained by various clustering methods are illustrated and compared. Quantitative evaluation of cluster quality is presented in the next chapter.

3.1 Previous Work: Automatic Clustering of Words

Several algorithms have been proposed for automatically constructing partitions of the vocabulary based on a corpus. They are basically classified into three types. One type is based on shuffling words from class to class starting with some initial set of classes (Kneser and Ney 1993, Ueberla 1995, Martin, Liermann and Ney 1995). The optimization methods discussed in the previous chapter use this type. A second type is agglomerative hierarchical clustering. Bottom-up merging of classes is repeated starting from a set of singleton classes, which contain only one word (Brown et al. 1992). The last type, divisive hierarchical clustering, is a top-down method which iteratively partitions classes into their subclasses starting with some initial configuration (McMahon and Smith 1996, Pereira, Tishby and Lee 1993, Li and Abe 1996). The majority of these algorithms have been developed for the purpose of improving language modeling for speech recognition. The task of language models is to estimate probability value for each word in a predefined vocabulary according to the context in which the word appears. In the n-gram model, which has been the most popular and successful model, a context is defined as the sequence

of the $(n - 1)$ previous words. In general, as the value of n increases, the predictive power of the language model increases, but at the same time the reliability of the model as a whole decreases because the data becomes more sparse. Word clusters are used to avoid the problem of data sparseness, or, in other words, to generalize the linguistic information in the training data.

Work by Kneser and Ney (1993) is a typical example of shuffling-based clustering. Starting with some initial partition of the vocabulary, a shuffling process is repeated until some termination criterion is met. Kneser and Ney used a single class containing all the words in the vocabulary as an initial partition in the case of an English vocabulary. In the process of shuffling, they allowed moving a word to a newly created empty class. With the conventional bigram-based maximum likelihood criterion, however, this would result in a set of singleton classes (except for a possible effect of stacking in a local optimum), which corresponds to a word bigram model. To avoid this, they introduced a modified optimization criterion which they call the leaving-one-out likelihood. The leaving-one-out likelihood is essentially the same as the maximum likelihood except that instead of using the entire training corpus $w_1 w_2 \dots w_N$, the training corpus with the single event (w_{i-1}, w_i) removed is used for training, and only one sample (w_{i-1}, w_i) is used as the *held-out* part to simulate unseen events. This process of slicing out the held-out event is repeated N times so that all N partitions with one held-out event are considered in the optimization. This leaving-one-out method is a special type of cross-validation. By using the leaving-one-out likelihood, the process stops creating a new class at a certain point with the optimum number of classes. This method can find an optimum number of classes automatically, but at the expense of introducing another parameter, a discount constant, which should depend on the data size as well as the types of the data and therefore must be empirically determined.

Ueberla (1995) extended Kneser and Ney's algorithm to deal with clustering using higher order n -grams. He also proposed a heuristic to speed up the algorithm. Time complexity becomes a crucial problem when the data size or the vocabulary size becomes large. The time complexity of Kneser and Ney's algorithm is $O(I * V * (V + C^2))$, where V is the vocabulary size, C is the number of classes, and I is the number of iterations

of shuffling. With the heuristic version of the algorithm proposed by Ueberla, the time complexity becomes $O(I(V^2 + V * C + C^2))$. According to Ueberla, C^2 is dominant in $(V + C^2)$ because C *crucially determines the quality of the resulting language model and one would therefore like to choose it as big as possible*. In that case, the dominant term before the improvement is IVC^2 and the dominant term after the improvement is IV^2 , and the latter is much smaller. Ueberla's clustering experiment with one million words of WSJ corpus shows that the quadratic growth of execution time with respect to the number of classes can be reduced to nearly linear time by his heuristics. However, the V^2 term is still a big obstacle when we try to process a large vocabulary in the range of tens of thousands.

The second type of clustering algorithms (i.e. bottom-up merging algorithms) is represented by the mutual information (MI) clustering algorithm proposed by Brown et al. (1992). The process is basically as follows. Suppose we have a vocabulary of size V and a large text. Then we first assign each word in the vocabulary to its own distinct class. Starting with the V singleton classes, we repeatedly merge a pair of classes according to some information theoretic criterion until C classes remain, where C is a predefined number of classes. The criterion used is the average mutual information (AMI) between pairs of adjacent classes. Brown et al. showed that the partition of the vocabulary that maximizes the AMI also maximizes the likelihood of a bigram class model generating the text. The pair of classes whose merging reduces AMI the least is chosen to be merged. With certain optimization and approximation, the time complexity of this algorithm can be reduced to as low as $O(VC^2)$. This is linear in the size of the vocabulary. A more detailed description of this algorithm will be given in Section 3.3.1.

The MI clustering algorithm was also proposed initially to improve the quality of language modeling. However it has been shown, in a series of papers by the IBM group, that this algorithm is also useful for other purposes, including the construction of decision tree Part-Of-Speech tagging models (Black et al. 1992) and parsing models (Black et al. 1993, Magerman 1994) as well as a maximum entropy model for prepositional phrase attachment (Ratnaparkhi and Roukos 1994). However, no quantitative analysis of the effects of the quality of clusters on the performance of these models has yet been made.

The last type of clustering, a top-down method, includes work by McMahon and Smith (1996). They conducted a binary top-down form of word clustering and employed AMI as an optimization criterion. Their goal of constructing word clusters was also to improve statistical language models. Instead of constructing a single flat layer of word classes, however, they constructed hierarchical clusters of words and proposed a multilevel smoothed bigram model which used the hierarchical clusters. Their clustering algorithm is as follows.

Suppose that a training corpus and a vocabulary V are given. First, an initial binary tree representation t of the vocabulary is constructed by randomly assigning bit strings to words. The first bit of a word represents a position (left or right) of the word in the first level (depth 1) of the binary tree. Let s be the depth of the binary tree. Then there are 2^s classes at depth s . Now, starting with depth $s = 1$, do the following.

1. Calculate the AMI at depth s for the current tree t as follows:

$$M_s(t) = \sum_{c_i, c_j} Pr(c_i c_j) \log \frac{Pr(c_i c_j)}{Pr(c_i) Pr(c_j)},$$

where c_i and c_j are word classes at depth s .

2. Shuffle words among classes in depth s for a higher $M_s(t)$ value until no single move of a word leads to a higher $M_s(t)$ value.
3. Fix the partition at depth s ; increment s by 1.

If s reaches some predefined number then stop; otherwise go to step 1.

The cost of each iteration of this algorithm is $O(V^3)$ for vocabulary size V . For V on the order of tens of thousands, this is too expensive. Therefore, they processed only the most frequent 569 words in the vocabulary using this top-down method in their experiment.

All three types are driven by some objective function, in most cases by perplexity or average mutual information. The merit of the bottom-up merging algorithm 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, this type is prone to being trapped at a local optimum. The first type (shuffling-based) is

more robust to the local optimum 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 type only provides a means of partitioning the vocabulary and it doesn't provide a way of constructing a hierarchical clustering of words. The third type, a top-down method, can provide a way to construct hierarchical clusters, but its time complexity is too large for a large scale task. In this work we adopt the merging approach and propose a method of constructing hierarchical clusters. An attempt will also be made to combine the first two types of clustering.

Soft Clustering of Words

All of the three types of word clustering methods discussed above are what is called *hard* clustering in which any word either is or is not a member of a particular class. Most of the large scale word clustering methods reported in the literature are hard clustering. Moreover, in the above cases any two classes are either disjoint or else one completely contains the other. The latter case happens when the cluster is hierarchical. However, these types of clusters cannot completely capture the nature of ambiguity natural language exhibits. An obvious example is a homonym. The word *bank* can be in the same class as *treasury* or it can be in a totally different class containing words like *shore* or *dike*. It is not unusual for the same word to be in different classes depending on the context in which the word occurs. One way to cope with this is to construct a cluster with overlapping classes. Another way is to treat each homonym as a different entity. In this case each occurrence of a word in the text has to be disambiguated with respect to a word sense, which itself is a new issue and constitutes one research area (Schutze 1992, Dagan and Itai 1994, Yarowsky 1995, Pedersen and Bruce 1997).

A more general approach is to construct *soft* clustering in which each word is a member of multiple classes with some probability. Pioneering work in this direction was conducted by Pereira et al. (1993) and Lee (1997). The vocabulary to be clustered in their work is a set of nouns, and the raw knowledge for the clustering is a set S of pairs (x, y) , where y is a verb and x is the head noun of y 's direct object. They collected these pairs

using the combination of a statistical POS tagger (Church 1988) and a regular expression pattern matching method (Yarowsky 1992). The occurrence of each pair is assumed to be independent.

Their clustering method is divisive and hierarchical. The system starts with a single cluster and successively splits each cluster into two clusters. Each cluster is represented by the cluster centroid c which is placed at the weighted average over all data points x . For each head noun x , the probability distribution of x over the set \mathcal{Y} of the verbs is defined by $P(y|x)$ (or simply $P(\cdot|x)$), and the dissimilarity, or distance, of two nouns x_1, x_2 is measured by the KL Divergence of their distributions:

$$D(x_1||x_2) = \sum_{y \in \mathcal{Y}} P(y|x_1) \log \frac{P(y|x_1)}{P(y|x_2)}$$

Instead of measuring distances of all the pairs of nouns, they only use distances of nouns and cluster centroids for their modelling.

The search for the proper parameter settings is guided by two principles, minimum distortion and maximum entropy. The closed-form solution that extremizes the two criteria is given by the following equations:

$$P(c|x) = \frac{\exp(-\beta d(x, c))}{Z_x} \tag{3.1}$$

$$P(y|c) = \sum_x P(x|c) P_{MLE}(y|x), \tag{3.2}$$

where β is a free parameter, $d(x, c)$ is a notational shorthand for $D(P_{MLE}(\cdot|x)||P(\cdot|c))$, $P_{MLE}(y|x)$ is the empirical distribution of a noun x , and $Z_x = \sum_c \exp(-\beta d(x, c))$ is a normalization function. For a fixed value of β , the search for the proper parameter values is a two-step iteration:

- membership probabilities $P(c|x)$ are calculated by equation 3.1 with fixed centroid distributions $P(y|c)$,
- obtained values of $P(c|x)$ are plugged into equation 3.2 to update $P(y|c)$,

and this cycle is repeated until the parameters converge. The whole process is then repeated with a gradually increasing value of β until β reaches some predefined maximum

value β_{MAX} . At each incrementation of β value, classes (or class centroids) are split into two or more subclasses. Therefore the final number of classes is controlled by β_{MAX} . A hierarchical structure is constructed by tracing the history of each class split.

Saul and Pereira (1997) formalized a method of soft clustering of words as an Expectation-Maximization (EM) algorithm. They constructed a class-based bigram model in which the mapping from words to classes is probabilistic. The model predicts that word w_1 is followed by word w_2 with probability

$$Pr(w_2|w_1) = \sum_{c=1}^C Pr(w_2|c)Pr(c|w_1)$$

The hidden variable in this model is the class label c to which each word w_1 belongs with some probability $Pr(c|w_1)$. $Pr(w_2|c)$ denotes the probability that the word w_2 directly follows words in the given class c . EM algorithm is applied to estimate these parameters of the hidden variable models. It is guaranteed that the overall log-likelihood of the model generating the training text increases at each iteration of updates of the model parameters. Although both Brown et al. (1992) and Saul and Pereira (1997) decompose conditional bigram probabilities $Pr(w_2|w_1)$ using class-based parameters, the latter method has an advantage, besides the softness of the clustering, in that the log-likelihood is directly optimized in the process of parameter estimation. The utility of this soft clustering method demonstrated through experiments is that the model is intermediate both in size and in accuracy between unigram ($C = 1$) and bigram ($C = V$) models. It is also demonstrated that the model assigns non-zero probability to all the bigrams in the unseen test set, which is a useful feature when we construct language models.

Rooth et al. (1999) also proposed an EM-based soft clustering method. Unlike the method of Saul and Pereira, in which clusters are derived directly from a plain text, their method uses a sample of pairs of verbs and nouns which participate in grammatical relations of either verbs and their subjects or transitive verbs and their objects. These pairs are gathered by parsing sentences. The model treats verbs (v) and nouns (n) as conditioned on a hidden class $c \in C$. The joint probability of a pair (v, n) is decomposed using class-based parameters as follows:

$$Pr(v, n) = \sum_{c \in C} Pr(c, v, n) = \sum_{c \in C} Pr(c)Pr(v|c)Pr(n|c)$$

The conditioning of two words v and n on each other is made only through the classes c . Accordingly, the classes to be constructed are not classes of individual words, but classes of pairs of verbs and nouns. The evaluation of the models is conducted through a pseudo-disambiguation task which is similar to the task Pereira et al. (1993) and Lee (1997) have employed to demonstrate the usefulness of the obtained clusters. The task is to judge which of two verbs v and v' is more likely to take a given noun n as its argument. No comparison is given, however, with alternative approaches. The model is also applied to create a lexicon of several hundred verbs with subcategorization frames which are labeled with latent classes.

Although soft clustering is no doubt better suited for NLP tasks than hard clustering, several important issues have to be further explored. One issue is its heavy computational demand. In the case of Pereira et al. (1993) and Lee (1997), the time required to update the membership probabilities using equation 3.1 is $O(CV^2)$, and the time to update all the centroid distributions using equation 3.2 is also $O(CV^2)$. Therefore the total running time is $O(KCV^2)$, where K is the total number of iterations for all values of β . As a result, this method is prohibitively expensive for large vocabularies.¹ In case of Saul and Pereira, no analysis of time complexity is given, but the maximum number of classes reported is 32.

Another issue is the very utility of the softness for NLP tasks. Suppose a word w belongs to class C_i with probability p_i for $i = 1, 2, 3, \dots, n$. Suppose further, for simplicity, that all p 's except p_j and p_k are nearly equal to zero. Then two major *senses*² are depicted in the cluster. This may be useful by itself for lexicographical purposes. However, when the cluster is used for NLP tasks, the context in which w occurs may very well give more accurate information than the cluster's probabilities. For example, the word *bank* in the context of "*river bank*" has very little chance of being related to *treasury* even if the two clusters' probabilities clearly indicate that the financial usage of the word is much more common than the geographical one. How to integrate the a priori probabilities of soft clusters with the context information is still an open question.

Regarding the utility of soft clustering, Lee and Pereira (1999) suggested that the

¹Actually, the maximum size of vocabulary reported is 1000.

²The *sense* distinction referred to here does not necessarily correspond to that found in dictionaries.

advantage of soft clustering over hard clustering may be computational³ rather than in modeling effectiveness. In the evaluation of soft clusters in the pseudo-disambiguation task, they employed two heuristic ways of generalizing information on word distribution using clusters: one is to use the weighted average of contributions of all the clusters to which the word in question belongs, and the other is to simply use the estimate of the nearest cluster to the word. Their conclusion is that the weighted-average method never seems to outperform the nearest-cluster method, indicating that hard clustering is enough for this particular task. Further improvement of soft clustering technique seems to be required to take full advantage of the softness of clusters.

3.2 Word Clustering for NLP Tasks

As described in the previous section, various approaches have been proposed for clustering words, mostly for the improvement of language modeling. The majority of successful methods use, as optimization criterion for clustering, the log-likelihood of the model generating the clustering text, or equivalently, the cross-entropy of the model with respect to the empirical word distribution. This cross-entropy is often re-expressed as perplexity, and it is widely known that language models with lower perplexity generally produce better speech recognition results.

Some pieces of work that show practical use of clustering for NLP tasks include Charniak (1997) as well as those by the IBM group mentioned earlier. In his probabilistic context-free parsing model, Charniak used clusters of head words for conditioning probabilities of derivations instead of conditioning by individual heads. It is shown that using word classes for purposes of smoothing has a moderate effect of increasing parsing accuracy.

Compared to the efforts to improve language modeling, however, less attention has been paid to the application of clustering techniques to NLP tasks. In particular, very little work has been done to quantitatively assess the effects of clustering quality on the performance of NLP tasks. The aim of this dissertation is to investigate how the improvement of entropy-based clustering techniques improves performance of NLP tasks. In particular,

³Their argument is based on the fact that the problem of finding a partition that minimizes some optimization function is NP-complete.

it is shown that using cross-entropy measures to cluster words and compounds according to their mutual substitutability improves performance on NLP tasks, especially for rare events. In the following sections the basic framework of the word clustering methods investigated in this thesis work is presented.

3.3 MI Clustering and Hierarchical Word Clustering

In this section, we will first describe the MI clustering algorithm which serves as a base line upon which various kinds of improvements will be made. The MI clustering algorithm presented here is based on Brown et al. (1992), although an important formula is modified for the sake of mathematical soundness (Ushioda 1996a). Following that, a hierarchical word clustering algorithm will be presented.

3.3.1 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:

$$L(\pi) = -H + I \tag{3.3}$$

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 3.4:

$$I = \sum_{c_1, c_2} Pr(c_1 c_2) \log \frac{Pr(c_1 c_2)}{Pr(c_1)Pr(c_2)} \tag{3.4}$$

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 bottom-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 prede-

fixed number C . The time complexity of this basic algorithm is $O(V^5)$ when implemented straightforwardly, as can be seen below.

- A. There are in total $V - C = O(V)$ merging steps.
- B. After n merging steps, $V - n$ classes remain, and in the next merging step we have to investigate $\binom{V - n}{2} = O(V^2)$ trial merges, only one of which will be made effective in the later process.
- C. One trial merge at step n involves summations of $(V - n)^2 = O(V^2)$ terms for the calculation of AMI in equation 3.4.

Therefore the total time complexity is $O(V^5)$.

By eliminating redundant calculation, however, the time complexity can be reduced to $O(V^3)$ as described in some detail below. In short, the point is that part C can be done in constant time by:

1. computing only those terms in equation 3.4 whose values have changed by the previous merge ($O(V^2) \implies O(V)$).
2. storing the result of all the trial merges at the previous merging step ($O(V) \implies O(1)$).

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

$$I_k = \sum_{l,m} q_k(l, m) \quad (3.5)$$

$$q_k(l, m) = p_k(l, m) \log \frac{p_k(l, m)}{pl_k(l)pr_k(m)} \quad (3.6)$$

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

$$p_k(l, m) = Pr(C_k(l), C_k(m)), \quad (3.7)$$

and

$$pl_k(l) = \sum_m p_k(l, m), \quad pr_k(m) = \sum_l p_k(l, m). \quad (3.8)$$

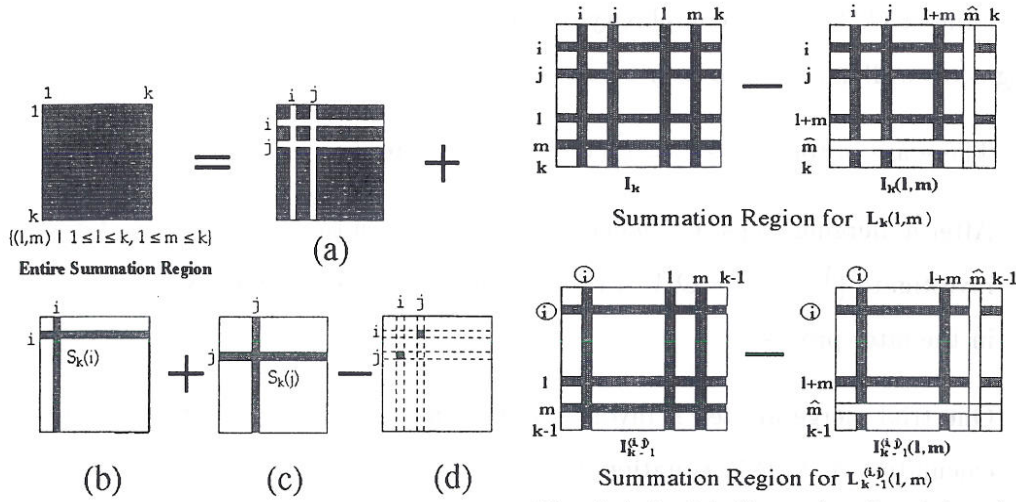


Figure 3.1: Summation Region (1)

The circle in \textcircled{i} indicates that class i here is different from class i for $L_k(l,m)$, that is: $C_{k-1}(i) = C_k(i+j)$

Figure 3.2: Summation Region (2)

In equation 3.5, q_k 's are summed over the entire $k \times k$ class bigram table in which cell (l, m) represents $q_k(l, m)$. Now suppose that we investigate a trial merge of $C_k(i)$ and $C_k(j)$ and compute the AMI reduction, $L_k(i, j) \equiv I_k - I_k(i, j)$, by this merge, where $I_k(i, j)$ is the AMI after the merge. As illustrated in figure 3.1, the summation region of equation 3.5 can be represented as a union of three parts, (a), (b), (c) minus (d). Out of these four parts, the summation over region (a) does not change its value by the merge of $C_k(i)$ and $C_k(j)$. Therefore, to calculate $L_k(i, j)$, the summation region can be reduced from a two dimensional region (a square region) to a one dimensional region (lines), hence, the complexity of part C can be reduced from $O(V^2)$ to $O(V)$. Using the notation $C_k(i+j)$ which represents a class created by merging $C_k(i)$ and $C_k(j)$, the AMI reduction can be given by equation 3.9:

$$L_k(i, j) = s_k(i) + s_k(j) - q_k(i, j) - q_k(j, i) - \left(\sum_{l \neq i, j} q_k(l, i+j) + \sum_{m \neq i, j} q_k(i+j, m) + q_k(i+j, i+j) \right) \quad (3.9)$$

where

$$s_k(i) = \sum_l q_k(l, i) + \sum_m q_k(i, m) - q_k(i, i) \quad (3.10)$$

After calculating L_k 's for all the pairs of classes, we choose the pair for which L_k is least, say, $C_k(i)$ and $C_k(j)$ with $i < j$, then we merge that pair and rename the new

merged class as $C_{k-1}(i)$, and go on to the next merging step with a new set of $k-1$ classes. Except for $C_k(i)$ and $C_k(j)$, all the classes are indexed the same way after the merge, that is, we rename $C_k(m)$ as $C_{k-1}(m)$ for $m \neq i, j$. If $j \neq k$, we rename $C_k(k)$ as $C_{k-1}(j)$. If $j = k$, $C_k(k)$ just disappears after the merge.

Further optimization is possible by storing all the values of L_k in the previous merging step. 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 $L_k(l, m)$ is that $L_{k-1}^{(i,j)}(l, m)$ is the AMI reduction by merging class l and class m after merging class i and class j , whereas $L_k(l, m)$ is the AMI reduction by merging class l and class m without merging class i and class j . Therefore, 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 $(C_k(i), C_k(j))$. To see it graphically, the summation regions of the class bigram table for $L_{k-1}^{(i,j)}(l, m)$ and $L_k(l, m)$ are illustrated in Figure 3.2. Because the summation over the region $\{(x, y) | x \neq i, j, l, m \text{ and } y \neq i, j, l, m\}$ does not change its value by the merge of class i and class j , or the merge of class l and class m , that region is omitted in the graph. Furthermore, as shown below, most regions in the graph cancel each other out when we calculate $L_{k-1}^{(i,j)}(l, m) - L_k(l, m)$, leaving only a number of point regions, hence the complexity of part C can be reduced to constant.

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

$$L_{k-1}^{(i,j)}(l, m) - L_k(l, m) = -(I_{k-1}^{(i,j)}(l, m) - I_k(l, m)) + (I_{k-1}^{(i,j)} - I_k). \quad (3.11)$$

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 possible cancellations have taken place. Then, we have

$$L_{k-1}^{(i,j)}(l, m) - L_k(l, m) = -(\hat{I}_{k-1}^{(i,j)}(l, m) - \hat{I}_k(l, m)) + (\hat{I}_{k-1}^{(i,j)} - \hat{I}_k), \quad (3.12)$$

where

$$\hat{I}_{k-1}^{(i,j)}(l, m) = q_{k-1}(l + m, i) + q_{k-1}(i, l + m) \quad (3.13)$$

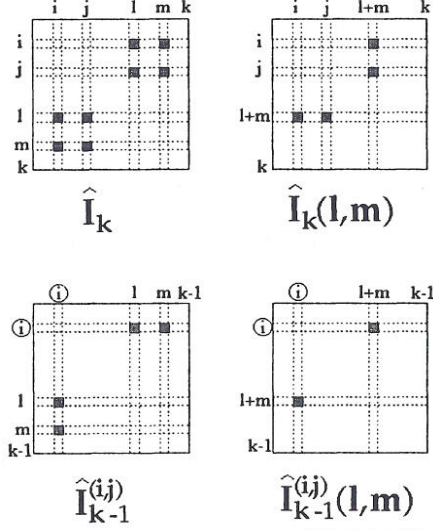


Figure 3.3: Summation Region (3)

$$\hat{I}_k(l, m) = q_k(l + m, i) + q_k(i, l + m) + q_k(l + m, j) + q_k(j, l + m) \quad (3.14)$$

$$\hat{I}_{k-1}^{(i,j)} = q_{k-1}(i, l) + q_{k-1}(i, m) + q_{k-1}(l, i) + q_{k-1}(m, i) \quad (3.15)$$

$$\begin{aligned} \hat{I}_k &= q_k(i, l) + q_k(i, m) + q_k(j, l) + q_k(j, m) + q_k(l, i) + q_k(l, j) \\ &\quad + q_k(m, i) + q_k(m, j) \end{aligned} \quad (3.16)$$

The summation region of the \hat{I} 's in equation 3.12 is illustrated in Figure 3.3. Brown et al. seem to have ignored the second term of the right hand side of equation 3.12 and used only the first term to calculate $L_{k-1}^{(i,j)}(l, m) - L_k(l, m)$.⁴ For the sake of mathematical completeness, we will use equation 3.12.

Even with the $O(V^3)$ algorithm, however, the calculation is not practical for a large vocabulary of order 10^4 or higher. Since part A must require $O(V)$ time, part B is the only part which can be modified. In part B we allowed all the possible pairs of classes to be considered for merging, but we can restrict the domain of possible merging pairs to investigate as follows. 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. So initially the $C + 1$ most frequent words are in the merging region. Then we:

⁴Actually, it is the first term of equation 3.12 times (-1) that appeared in their paper, but we believe that it is simply due to a misprint.

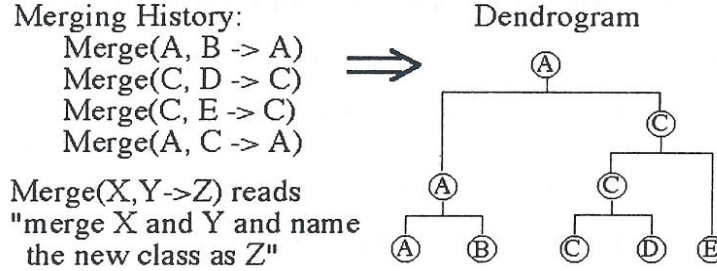


Figure 3.4: Dendrogram Construction

1. Combine the pair of classes whose merger results in the minimum AMI reduction.
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.

3.3.2 Hierarchical Word Clustering 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. A simple example with a five word vocabulary is shown in Figure 3.4. 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 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 to merge two higher frequency classes with substantial sizes.

A new approach we adopt incorporates the following steps:⁵

1. MI-clustering: Make C classes using the mutual information clustering algorithm with the merging region constraint mentioned in (cf. § 3.3.1).
2. Outer-clustering: Replace all words in the text with their class token⁶ and execute

⁵According to personal communication with John Lafferty, essentially the same algorithm for Hierarchical Word Clustering presented here was independently developed and used at IBM as early as 1991. It just wasn't published.

⁶In the actual implementation, we only have to work on the bigram table instead of the whole text.

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 $\{C(1), C(2), \dots, C(C)\}$ 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 $V_1 = \{\text{all the words in } C(i)\}$, $V_2 = \{C_1, C_2, \dots, C_{i-1}, C_{i+1}, C_C\}$, 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 continuing to merge with a merging region whose width is $|V_1|$ initially, decreasing by one with each merging step.

(b) Continue 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 comparatively balanced binary tree representation of words in which those words which are close in meaning or syntactic feature are close in position. Figure 3.5 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. In this example, the depth of the whole tree is 33, whereas the simple dendrogram method with the same text and vocabulary creates trees with depth 1000 or more. 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.

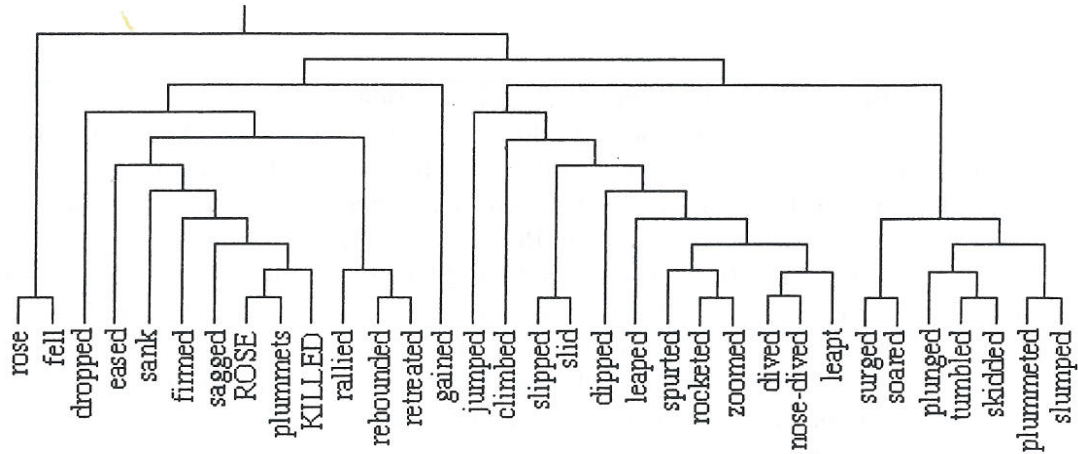


Figure 3.5: Sample Subtree for One Class

3.3.3 Example of Clusters

Some representative examples of clusters are given in this section. A 50 million-word text of Wall Street Journal articles is processed to create a set of 500 word classes by MI clustering. Five rounds of reshuffling (cf. § 3.4) are then applied. The size of the vocabulary is 70,000.

One distinctive feature of clusters created by the MI clustering method is that the size of classes varies considerably. The distribution of class sizes is given in Figure 3.6. A point at (X, Y) shows that there are Y classes in the cluster whose size is between $X - 9$ and X . The maximum size is as large as 6213. On the other hand, there are 124 classes whose size is one (singleton classes). The average class size is 140, and the standard deviation is 368.

Table 3.1 shows the three largest classes. Words are listed in descending order of frequency in the text, and each list is truncated at an arbitrary point. The number of elements in each class is, from top to down, 6213, 3077, and 1540.

The first class in the table is a class of family names, the second class is a class of company names. The third class is some mixture of family names and company names. Unlike categorization, clustering only divides elements into classes and we cannot control the nature of each class to be created. We can at best identify common characteristics of elements in a class after the classes are formed. ⁷

⁷There are, however, ways to influence the nature of each class by controlling the initial state of MI

Gorbachev Baker Miller Jackson Trump Morris Boesky Wilson Davis Greenspan Icahn Wright Milken Hall Meese Moore Taylor Cohen Robertson Walsh Yeltsin Murphy Anderson Volcker Gray Jacobs Pickens Edelman Murdoch Shultz Walker Hart Gates Levine Bennett Robinson Casey Keating Perot Reed Freeman Adams Iacocca Evans Buchanan Ryan Kelly Friedman Simmons Wolf Regan Siegel Weiss Brennan Levy Darman Cook Schwartz Seidman Kaufman Buffett Salinas McCarthy Cox Johnston Strauss Bilzerian Goldberg Sununu Pierce Li Hammer Abrams Powell Mitterrand Posner Coleman Garcia Griffin Gross Castro Rice Tisch Thornburgh Perelman Stern Shaw Klein Olson Webster Peterson Fitzwater Schneider Mandela Greenberg Tsongas Clarke Katz Peters Meyer Hoffman Mulheren Goldstein Breeden Deng Goodman Giuliani Levin Owen Stein Lorenzo Weinberger Ruder Yeutter Nakasone Henderson Farley Simpson Lucas McFarlane Skinner Roth Cole Chapman Sanders Dixon Rosenberg Porter Reilly Takeshita Stempel Rosen Hirsch Miyazawa Schmidt Duncan Gonzalez Poehl Kerkorian Bailey Koch Kerrey ...

Ford Salomon Chrysler Moody Texaco Citicorp Boeing CBS Warner Time Apple PaineWebber GE Sun Digital Toyota RJR Campeau Kodak Microsoft Intel Fidelity Honda Nomura Sony Exxon Coca-Cola UAL Nissan BankAmerica Prudential-Bache Irving Guinness Allied Prudential Sterling Lotus Compaq USX MCI MCA Lockheed USAir Occidental Northrop Gillette Robins Unisys Burlington LTV Philips NCR Chevron AMR Henley Xerox Mesa Mobil Amoco Macy BP Farmers ITT GTE B.A.T Toshiba Volkswagen Motorola NCNB Dome Pillsbury Allegheny Allegis Siemens Harcourt Macmillan Kraft WPP Daiwa Genentech Avon Revlon Southland Nova Mazda Tenneco Arco Hewlett-Packard PepsiCo Rockwell Hanson Nikko Cray Nestle Orion Integrated Polaroid Greyhound Monsanto Equitable TRW Unocal SmithKline Aetna Coors Nynex NEC Reebok Fiat Southmark Hitachi Zenith Dominion Wal-Mart Fujitsu Upjohn Interco Seagram Volvo McCaw Lucky Viacom Hyundai Wickes Jaguar Reliance Nintendo Resorts Sotheby Corona Firestone Grumman Gannett Coniston Beatrice Wedtech USG Bristol-Myers Allied-Signal Travelers Pfizer ...

Morgan Smith Goldman Johnson Brown Kidder King Price Harris Saddam Young Maxwell McDonnell Williams Saatchi McDonald Scott Simon Phillips Turner Allen Ross Lewis Jordan Clark Thompson Hughes Fox Campbell Green Montgomery Franklin Mitchell Bass Marshall Cooper Wang Stone Sullivan Dayton Thomson Hunt Clinton Parker Stevens Nelson Edwards Singer Murray Rogers Wood Hilton Hamilton Rose Brooks Lilly Steinberg Stewart Rothschild Corning Grant Foster Grace Fisher Bernstein Dillon Collins Baxter Schwab Andersen Morton Mason Jefferies Burns Gould Graham Rich Morrison Armstrong Bradley Taft Webb Marietta Bull Lane Christie Love Blair Stephens Olivetti Harper Kellogg Perry Marion Barnett Bryan Fleming Phelps Norton Gilbert Rockefeller Wagner Sharon McDermott Case Nielsen Marcus Spencer Wasserstein Forstmann Harrison Tyson Sherman Palmer Holland Fairfax Suzuki Putnam Kay Gabelli Barron Richardson Myers Matthews Forbes Dell Jefferson Moran Butler Gibson Scudder Keefe Solomon Needham Carson Whittle Todd Sharp Hampton Willis Rule Hunter Fish Snyder Hoffmann-La ...

Table 3.1: Large Classes

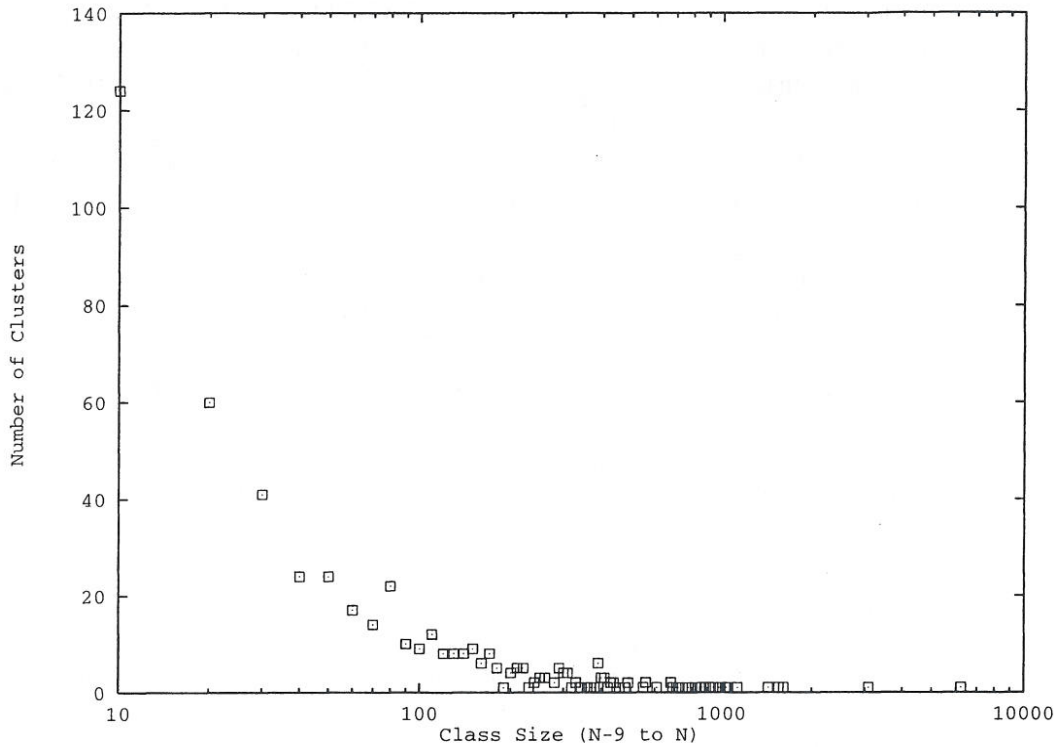


Figure 3.6: Distribution of Class Sizes

. , the of to a and in that for is \$ said on it by with at as from was be are have an will
would about they ...

Table 3.2: Examples of Singleton Classes

Some elements of the singleton classes are listed in Table 3.2. Most punctuation marks, prepositions, auxiliary verbs and determiners form singleton classes because their distributional characteristics are quite unique.

Table 3.3 shows examples of relatively coherent classes, and Table 3.4 shows some examples of bad classes.

The obtained cluster is further processed to create a hierarchical cluster using the method described in § 3.3.2. Figure 3.7 shows the distribution of bit length of word bits. The maximum length is 33 and the average length is 21.7. If the tree were completely balanced, the tree depth would have been 17.

clustering. For example, by using similar words as *seeds* in each initial class, we can expect to collect other similar words in the process of clustering. However, this method tends to create classes of a mixture of different subgroups unless the initial *similar* words are really similar in terms of the clustering criterion.

second third fourth fifth sixth seventh eighth 10th 11th ninth 14th rate. 12th 13th 15th
17th 16th 100th 50th 25th 30th 40th Caesarean 24th 70th 60th 27th 22nd 23rd 26th 200th
35th 75th aft 49th 38th 34th 46th 36th 28th 43rd 31st 51st 37th 29th 500th 45th 90th
33rd 80th 2,000-mile 41st 150th

two three five four six seven nine eight ten twelve two-and-a-half eleven twenty thirty
14-hour halcyon **Four fifty fifteen three-and-a-half forty better-off two-dozen sixty one-
and-a-half

still probably really certainly hardly obviously surely definitely undoubtedly scarcely
presently gonna doubtless unquestionably assuredly

major big leading key prominent dominant well-known prestigious high-profile pivotal
full-service staunch world-class state-chartered thorny high-flying pioneering reputed bud-
ding highflying reputable predominant top-tier die-hard Chicago-area price-depressing
Boston-area preeminent U.S.-flagged knotty barrier-free sub-par self-respecting similar-
sized top-10 world-famous reflagged wintry big-bank non-German trend-setting swanky
price-firming modest-sized big-stock once-proud non-lawyer hair-trigger second-story once-
thriving post-Challenger once-powerful late-summer heckuva fast-growth beaten-down

Japan Europe Canada Britain France China Mexico Israel Brazil Australia Switzerland
Italy Taiwan Singapore Spain India Poland Messrs Nicaragua Sweden Russia Argentina
Turkey Pakistan Lebanon Egypt Panama Cuba Hungary Baghdad Belgium Indonesia
Syria Chile Afghanistan Venezuela Peru Hawaii Norway Ireland Malaysia Tehran Thailand
Austria Yugoslavia Colombia Czechoslovakia Greece Finland Lithuania Libya J.P humans
Portugal Cambodia Denmark L.A Romania Angola Managua Nigeria Croatia Algeria Haiti
Bulgaria Honduras Sens Dealer Burma ...

significant great huge serious substantial sharp negative broad modest temporary wide
massive deep severe broader fundamental considerable rapid solid slight dramatic minor
tremendous steep sweeping sudden hefty bitter sizable partial constant mere comprehen-
sive strict stiff longstanding mild meaningful prolonged bold near-term genuine persistent
gradual precise nominal drastic vigorous chronic spectacular dismal lasting decent stunning
tangible sheer scant profound lingering handsome broad-based swift respectable periodic
continuous far-reaching meager concerted dire recessionary definite simultaneous ...

analysts traders dealers economists brokers experts critics participants specialists pro-
fessionals observers forecasters skeptics commentators pundits meteorologists NMTBA
wags demographers cynics timers Saxony market-watchers nutritionists catalogers con-
trarians audiophiles furriers psychotherapists numismatists seers Angelenos term-limits
Fed-watchers weathermen ophthalmologists

saying adding concerned showing talking calling worried asking convinced confident not-
ing indicating charging meaning suggesting demanding arguing skeptical claiming alleging
betting afraid assuming predicting explaining knowing recommending complaining hopeful
declaring insisting worrying signaling fearing asserting contending ensuring fearful stating
believing recalling realizing acknowledging concluding discovering speculating revealing
estimating confirming demonstrating conceding unsure implying proclaiming inconceiv-
able apprehensive sensing hinting boasting remembering pretending fretting unconvinced
inquiring noticing forgetting bragging imagining cautioning netting pausing bandied mut-
tering ...

Table 3.3: Examples of Good Classes

/ noncompetitively bldg = noncallable. TOTALS 6/88 Watched URGED FACES G.I ...
 (vs. Applications Offered e-In Than Gets SA-x Takes Agrees Become a- Romeo-b ...
 one none ands out-of-the-money
 only aged nary
 unit division subsidiary bottler knockoff cocoon Janes subsidiary divison co-trustee
 off aside amok blurs SDSB CDMA CPF
 each per 1/2-hour square-foot SUBTOTALS million-to- G.T million-vehicle milligram ...
 being z-Not contractually
 among afflicting implicating only. imploring engulfing amongst Amerindo one-price
 least par midnight noon half-price loggerheads gunpoint Kabi Nummi. daybreak all.

Table 3.4: Examples of Bad Classes

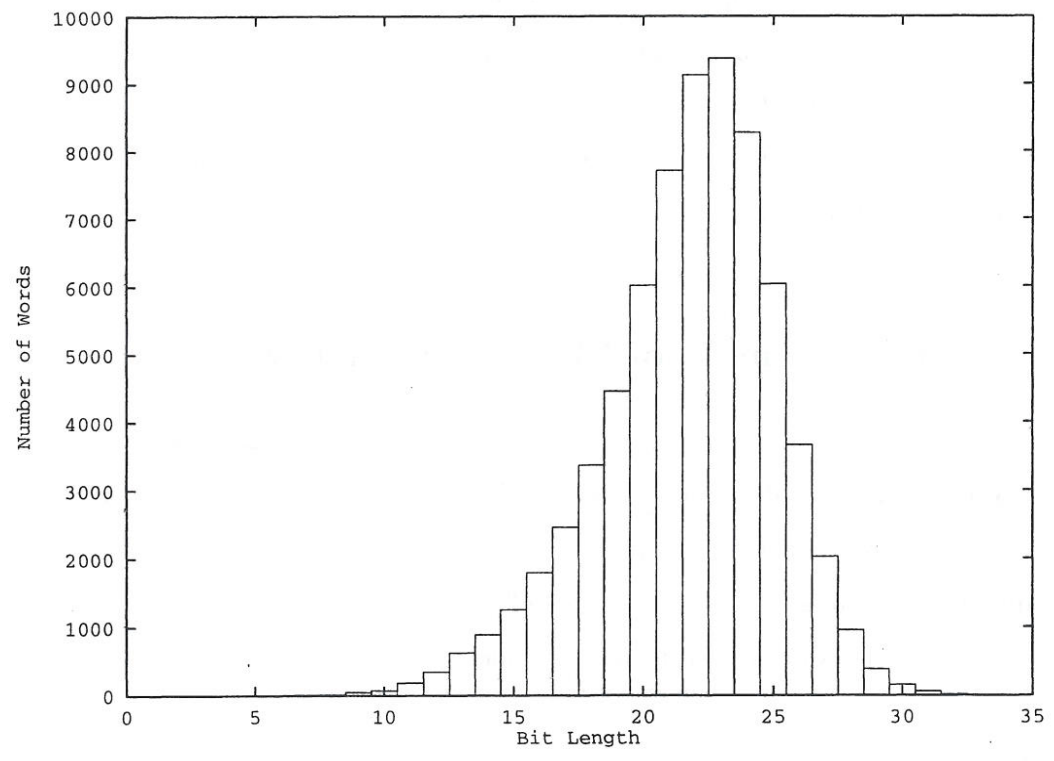


Figure 3.7: Distribution of Word Bits Length

Word	Word Bits
acknowledgment	00011001000111111110
acknowledgement	00011001000111111110
billion	011011010100
billon	011011010101
between	1000110110100
betwen	1000110110101
million	01101101000
milion	01101101001011100
millon	01101101001011101
milllion	0110110100101111
possibility	00011001000100100
possiblity	00011001000100101
uncollectible	001111100110001111010110
uncollectable	0011111001100011110101110

Table 3.5: Example of Misspelled Words

Some examples of subtrees are given in Figure 3.8. Note that two words in the figure, *spokeman* and *spokewoman*, are misspelled. Since distributional characteristics of a common misspelled word and of the correctly spelled word are quite similar, these words tend to come close to each other. Other such examples are given in Table 3.5. Many hyphenated words also behave quite similarly to non-hyphenated counterparts as exemplified in Table 3.6.

3.4 Improvement of Clustering Quality

Since the MI clustering algorithm is a greedy algorithm, the process is prone to being trapped at a local optimum. Indeed, a further refinement of clusters is possible after the completion of MI clustering, and we will discuss in this section methods to improve the quality of clusters, and give examples of improved clusters. More rigorous evaluation of cluster quality is presented in the next chapter.

Data Size

Perhaps the simplest method to improve the quality of clusters is to increase the amount

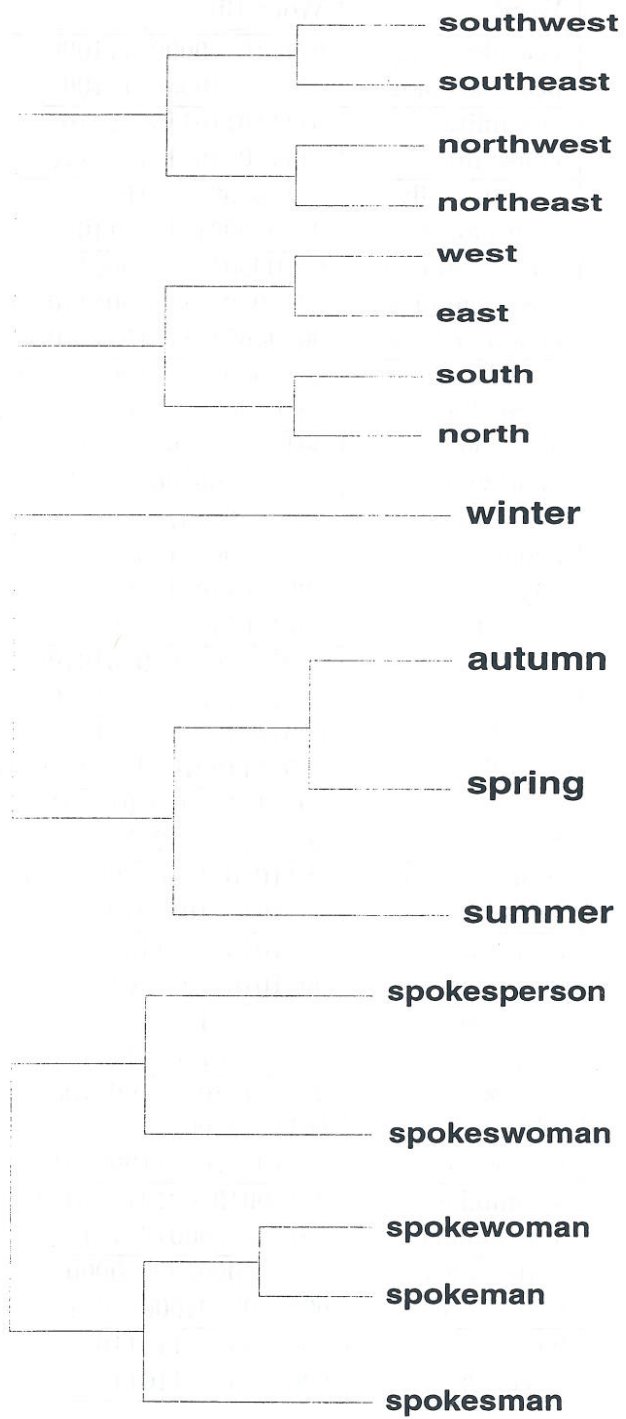


Figure 3.8: Example Subtrees

Word	Word Bits
peacekeeping	0111101100001111000
peace-keeping	0111101100001111001
nonmilitary	011110110110110110
non-military	0111101101101101110
semiannually	001000001110101100
semi-annually	001000001110101101
non-Communist	00100001111110011100
non-communist	001000011111100111010
noncommunist	001000011111100111011
Asia-Pacific	001000011111100111100
Asian-Pacific	00100001111110011110100
high-flying	001111100000111110110
highflying	001111100000111110111
loan-loss	001101001011000
loanloss	001101001011001
buy-outs	00011011001100
buyouts	000110110011010
run-up	00011110010101010100
runup	00011110010101010101
falloff	0001111001010101111010
fall-off	00011110010101011110110
preset	00111110001010101010110110
pre-set	00111110001010101010110111
supply-demand	0011010110100101111110
supply/demand	0011010110100101111111
anti-takeover	001101011011100
antitakeover	0011010110111010
Iran-Iraq	0011111011011111100
Iraq-Iran	0011111011011111101
midsized	0011010100111000100
midsize	00110101001110001010
mid-sized	00110101001110001011
lawmaking	0010001000010111101010
law-making	0010001000010111101011
write-down	000110000100010000
writedown	000110000100010001
buy-back	0001100111110110
buyback	0001100111110111

Table 3.6: Example of Hyphenated Words

of the clustering text. However, as far as the MI clustering method is concerned, we know of almost no quantitative analysis of the effects of the data size on the quality of clusters. From a practical point of view, it is important to know how much data is enough for a specific purpose or where the effect of increasing the data size begins to saturate.

Table 3.7 and Table 3.8 show how some of the classes change as the size of clustering text is increased.

Reshuffling

Other approaches to improving clustering quality are intended to compensate for the greediness of the MI clustering algorithm, which only guarantees local optimum. One such approach is to introduce a reshuffling process after the completion of MI clustering (Brown et al. 1992). The reshuffling process is as follows:

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 going from the most frequent word through the least frequent word.

This constitutes one *round* of reshuffling. In order to introduce the reshuffling process into hierarchical clustering, we only have to conduct the above reshuffling process just after step 1 (*MI-clustering*) of the hierarchical word clustering process (cf. § 3.3.2). Another way of looking at this approach is that it is a combination of a shuffling-based clustering method and a bottom-up merging method in which the result of the latter is used as a reliable initial set of classes for the former.

Table 3.9 show examples of how classes change after conducting five rounds of reshuffling.

Stepwise Clustering

Clustering Text Size: 5 million words

second third fourth whole bottom fifth sixth golden inner seventh eighth earliest longest
10th 11th feasibility 10-day ninth rocky onetime 12th Gallup 50-year-old 13th Serbian
marathon soft-spoken nominating year-old sticky Rainbow 33-year-old Bulgarian centrist
haunting three-hour violin hapless long-delayed polygraph breakaway 50th slippery nine-
member slender just-ended dilutive fast-paced Fight steamy fox Beat graveyard fateful
reclusive sophomore 30th pre-election mercurial Crocodile 90-minute domino four-member
watchful Zionist condensed roiling long-established Mick payment-in-kind 60th ...

Clustering Text Size: 10 million words

second third fourth fifth sixth seventh eighth 10th 11th foreseeable 10-day ninth 14th 12th
13th 15th faint half-day four-week 25th Yiddish 30th April-June full-length canine 24th
January-March crusty Janesville 27th PRC 23rd 26th pecking 35th living-room trembling
swan not-too-distant nitty-gritty medium-priced halo foreseeable 36th 1973-75 28th odds-on
titular dining-room 29th two-edged Jun. pituitary lowest-paid red-haired quiche opening-
day A-B stained-glass spindly AFC on-camera ...

Clustering Text Size: 20 million words

third second fourth fifth sixth seventh eighth mentally 10th 11th foreseeable ninth 14th
12th 13th 15th 17th 16th 11-20 half-day four-week 50th 1-10 25th 21-31 30th 40th right-
hand gentler proverbial 24th January-March 60th 27th July-September 22nd yearago 21-30
23rd 26th 35th mid-'80s not-too-distant 1-20 non-partisan upper-middle foreseeable 34th
46th 36th 28th odds-on 40-minute 31st 29th 90th 33rd languid fattest 1-ranked fifth-grade
butyl 150th three-run spindly Bedford-Stuyvesant AFC biannual CENSUS

Clustering Text Size: 50 million words

fourth second third fifth sixth seventh eighth 10th 11th ninth 14th 12th 13th 15th 16th
11-20 1-10 25th 21-31 30th April-June right-hand 24th 27th July-September 22nd 21-30
23rd 26th 35th 75th 1-20 01-10 49th 600-ship 34th 36th 28th 43rd 31st 29th 45th 4-5 33rd
5-6 girdle recession-plagued

Table 3.7: Example of Data Size Dependency (1)

Clustering Text Size: 5 million words

below around above behind beyond exceeding exceeds aboard strengthens underneath feast highlighting understate cornered outweighs alters overhauls notched decorate jeopardizes toasted usurped vented versed straddling hobbling unheeded crisscrossed mannequins approximating Molotov affixed blurs apprehend chucked spurns priming peppering eludes dramatizing thronged slitting

Clustering Text Size: 10 million words

above around below behind beyond exceeded exceeding exceeds misstated aboard equals beside enhances equaled underneath outpacing harassing insults misidentified intensifies outweighs bloodied commemorating exaggerates overhanging dilutes outstrips epitomizes smeared unheeded bores apprehend re-creating thumbing priming curtails higher. winging overestimates

Clustering Text Size: 20 million words

around above below behind beyond exceeding exceeds aboard equals alongside underneath outstripping outweighs exaggerates re-examining smeared approximating dissecting unhinged pronouncing amongst meteorology higher. askew outweighing overestimates neutralizes

Clustering Text Size: 50 million words

above around below behind beyond exceeding exceeds equals underneath outweighs exaggerates outstrips Chg. approximating undervalues outweighing

Table 3.8: Example of Data Size Dependency (2)

Before reshuffling														
fourth	second	third	fifth	sixth	seventh	eighth	10th	11th	ninth	14th	12th	13th	15th	16th
11-20	1-10	25th	21-31	30th	April-June	right-hand	24th	27th	July-September	22nd	21-30	23rd	26th	35th
75th	1-20	01-10	49th	600-ship	34th	36th	28th	43rd	31st	29th	45th	4-5	33rd	5-6
girdle recession-plagued														
After reshuffling														
second	third	fourth	fifth	sixth	seventh	eighth	10th	11th	ninth	14th	rate.	12th	13th	15th
17th	16th	100th	50th	25th	30th	40th	Caesarean	24th	70th	60th	27th	22nd	23rd	26th
200th	35th	75th	aft	49th	38th	34th	46th	36th	28th	43rd	31st	51st	37th	29th
500th	45th	90th	33rd	80th	2,000-mile	41st	150th							

Before reshuffling							
net	pretax	gross	after-tax	pre-tax	per-capita	non-interest	unearned
aftertax	spendable						
After reshuffling							
net	pretax	gross	after-tax	pre-tax	per-capita	non-interest	unearned
noninterest	aftertax	paid-in	paid-up	spendable			

Table 3.9: Reshuffling

Example of the effect of reshuffling. Clustering text size is 50 million words.

Another method of compensating for the greediness of the MI clustering is to undo *bad merging judgements* after the completion of clustering. Figure 3.9 shows the amount of AMI reduction for each merging step. A 50 million-word text of Wall Street Journal articles is used to create a set of 500 word classes. The size of the vocabulary is 70,000 and initial 70,000 singleton classes are merged until 500 classes remain. One natural way of identifying *bad* merging steps is to fit the plots in Figure 3.9 with some curve and judge that those plots above the fitted curve correspond to *bad* merging steps. Then for each class in the final clusters we can identify which words were put into the class by a *bad* judgement and eliminate these words from the class as *impurities*. In this way, we can *purify* each class. The eliminated words can then be re-classified starting with a set of *purified* classes.

A simple fit is made by averaging AMI reduction values of 1000 nearest points. Table 3.10 shows examples of classes before and after cutting off high AMI reduction points using this fit. Irrelevant words are more likely to be eliminated than relevant words.

Another merit of this method is that after the purification of the classes, we can tok-

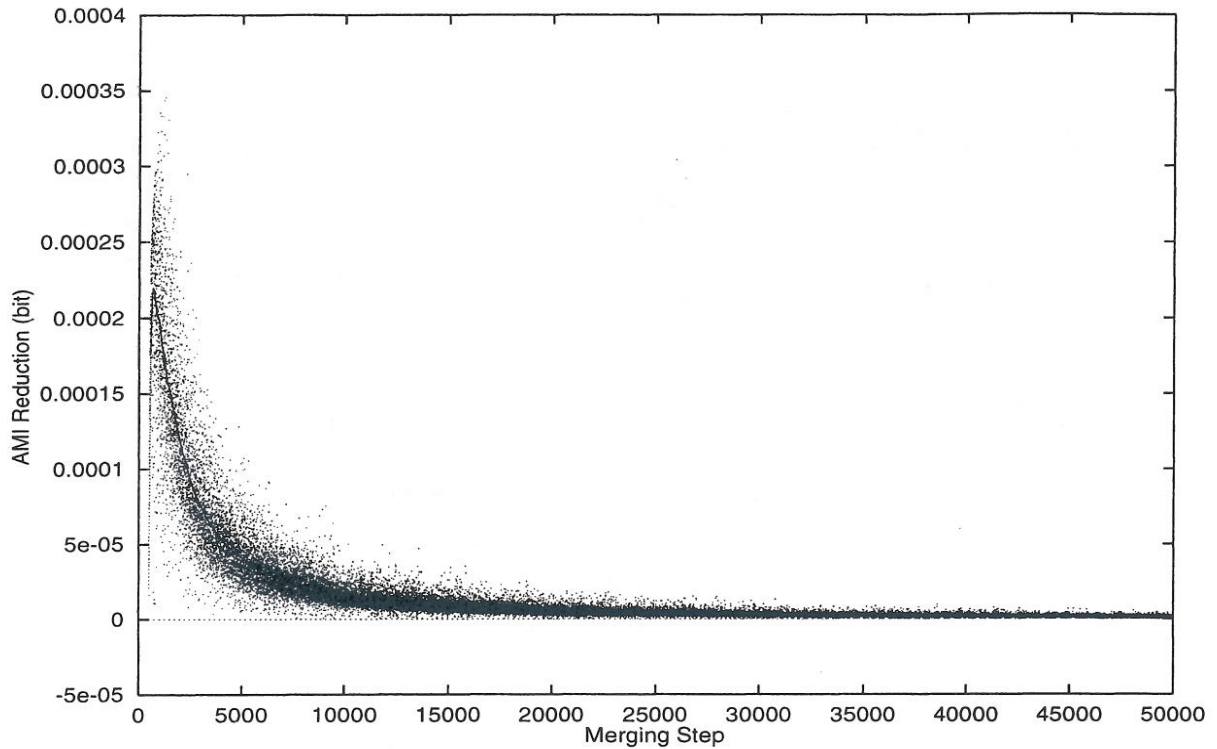


Figure 3.9: Reduction of Average Mutual Information

enize each class and replace all the words in the same class with a single token both in the text and in the vocabulary. As a consequence, the size of the vocabulary is reduced and hence the memory required to store the bigram table is reduced. This produces room for processing more text data with a larger vocabulary using the same memory. Additionally, the whole process can be repeated step by step.

Clustering with Multiple Texts (Cross-Validation)

Although the above method has several good features, it does not actually alleviate the very greediness of the algorithm. That is because we cannot completely eliminate the effect of having introduced *impurities* into a class even if they are removed at a later time, since some *impurities* may attract another impurity which may remain in the class till the end. A more dynamic way of ascertaining the goodness of each merging step is to consult

Before removal
unit division subsidiary arm affiliate segment radius Marketplace distributorship dispenser al. PHILIPS cocoon Seles VOLVO commissary subsidiary divison
After removal
unit division subsidiary segment Marketplace distributorship dispenser
Before removal
say suggest worry argue indicate predict contend insist complain warn acknowledge concede speculate assert allege fret reckon fretted grumble concur brag grouse stipulate decry bemoan theorize laud fantasize confide bicker fume Survive opine
After removal
say suggest worry argue indicate predict contend insist complain warn acknowledge concede speculate assert allege fret reckon grumble concur brag bemoan theorize
Before removal
report study review survey estimate forecast poll compound projection boycott chart tally encyclical Bendjedid
After removal
report study review survey forecast poll projection boycott tally encyclical
Before removal
years months weeks decades quarters Vegas generations centuries Raton Aires Cynwyd Lumpur Colinas thirds Vegas-based feng gigaflops eons Burkina millennia
After removal
years months weeks decades quarters generations centuries thirds
Before removal
yesterday Friday Monday Tuesday Wednesday Thursday Saturday Sunday Sundays semi-annually halts Fridays weekdays Thursdays semi-annually Showdown Pornography E.E Linjeflyg *Based Cited Marineland T.J K.C
After removal
yesterday Friday Monday Tuesday Wednesday Thursday Saturday Sunday Fridays

Table 3.10: Class Purification
Example of removing merging steps with high AMI reduction.

multiple texts on whether a pair of classes should be merged at each merging step. In the case of two texts, one example of the procedure is as follows. First, two sets of word bigram tables are created, a main table from one text and a sub-table from the other text. Then MI clustering is conducted with the main table, but when the merging judgement is not reliable enough, we consult the sub-table. The *reliability* of a merging judgement can be evaluated by comparing the AMI reduction of the merging with some threshold value that can be a function of the frequencies of the classes to be merged or some other factors.

The intuition behind this method is as follows. Suppose we have two texts, say a set of WSJ articles in 1988 and in 1989. Two words, especially if they are low frequency words, may just happen to appear in the same context often in one text, but never in the other text. In that case it is likely that those two words are not related and should not be put in the same class. Clustering with multiple texts is expected to enable this and other kinds of cross-validation.

A somewhat simpler variant to this cross-validation is to use two texts independently and combine the results. For example, if four words *A*, *B*, *C* and *D* are grouped together into one class when one text is used for clustering, and *A*, *B*, *C* and *E* are put into one class when the other text is used, then we can assume that *D* and *E* are not as relevant as the other three words are, or that their relevance is more context dependent. Then we can expect to create a less noisy or more coherent class by eliminating these less relevant words. This method, however, is based on the assumption that matching classes from two clusters is easy, so using two texts from very different sources is not practical.

Table 3.11 compares the results of this method with the results of regular MI clustering. A 50 million-word text of Wall Street Journal articles is split half and half, and the two 25 million-word texts are processed independently to create two cluster sets with 500 classes. A class from one set is considered to correspond to a class from another set if the total frequency, counted in the 50 million-word text, of all words common to both classes is more than half of the total frequency of each of the classes. Then the two corresponding classes are replaced with a new class consisting of only the words common to the two classes. Irrelevant words are much more likely to be eliminated by taking the intersection than relevant words, similar to the case of *Stepwise Clustering* discussed above.

Original
unit division subsidiary arm affiliate segment radius Marketplace distributorship dispenser al. PHILIPS cocoon Seles VOLVO commissary subsidiary divison
Cross-validation
unit division subsidiary glut subsidiary
Original
say suggest worry argue indicate predict contend insist complain warn acknowledge concede speculate assert allege fret reckon fretted grumble concur brag grouse stipulate decry bemoan theorize laud fantasize confide bicker fume Survive opine
Cross-validation
say suggest worry argue indicate predict contend insist complain warn acknowledge concede speculate assert allege fret reckon grumble concur grouse stipulate decry bemoan theorize fume opine
Original
report study review survey estimate forecast poll compound projection boycott chart tally encyclical Bendjedid
Cross-validation
report study survey forecast poll cap compound projection chart tally gerrymander
Original
years months weeks decades quarters Vegas generations centuries Raton Aires Cynwyd Lumpur Colinas thirds Vegas-based feng gigaflops eons Burkina millennia
Cross-validation
years months weeks decades quarters centuries seasons Cynwyd summers innings paragraphs thirds millennia
Original
yesterday Friday Monday Tuesday Wednesday Thursday Saturday Sunday Sundays semi-annually halts Fridays weekdays Thursdays semi-annually Showdown Pornography E.E Linjeflyg *Based Cited Marineland T.J K.C
Cross-validation
Monday Tuesday Wednesday Thursday Saturday Sunday semiannually Saturdays Mondays Fridays weekdays Thursdays

Table 3.11: Example of Crossvalidation

Original classes created using the 50 million-word text of Wall Street Journal articles is compared with *cross-validation* classes created by processing two sets of 25 million-word text independently and taking intersection of the two results.

Chapter 4

Evaluation of Clustering Quality

This chapter presents a quantitative evaluation of cluster quality using Part-Of-Speech tagging as a representative example of NLP tasks to which the clustering technique is directly applied. Improvement of cluster quality in terms of the clustering criterion (cross-entropy) is reflected in the improvement of the NLP task.

4.1 Introduction

In the field of corpus-based NLP, Part-Of-Speech (POS) tagging (Church 1988, Brill 1992, Cutting et al. 1992, Mikheev 1997) and parsing (deMarcken 1990, Jelinek et al. 1990, Lari and Young 1990, Magerman 1994, Collins 1996, Charniak 1997, Ratnaparkhi 1998) have attracted a significant amount of attention, partly because they serve as a basis for many other NLP tasks such as machine translation, information extraction and text summarization. In particular, POS tagging has been a central topic in the past several years because more and more training data is becoming available in many languages. In addition, many existing parsing systems use a POS tagger as a preprocessor to constrain the divergence of ambiguities (Charniak et al. 1994, Pereira and Schabes 1992). Many other techniques for structural analysis, such as verb frame extraction (Manning 1993, Ushioda et al. 1993, Briscoe and Carroll 1997), also use a POS tagger as a preprocessor. Therefore, we will choose POS tagging as a representative example of NLP tasks in which clustering quality is evaluated. In this chapter, we investigate utilization of clusters in two different types of POS tagging - a decision tree-based POS tagger and a class-based Markov tagger.

4.2 Decision-Tree Part-Of-Speech Tagger

MI clustering and hierarchical clustering experiments are performed using plain texts from six years of the Wall Street Journal Corpus to create clusters and hierarchical clusters (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. The number C of classes is set to 500, a realistic compromise considering that the larger C produces better clusters while the time complexity increases quadratically with the increase of C . The obtained hierarchical clusters are evaluated via the error rate of the ATR Decision-Tree Part-Of-Speech Tagger.

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 4.1 shows an example of an event with the current word *like*. The last pair in the event is a special item which shows the *answer*, the correct tag of the current word. The first two lines show questions about the 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 *linguistic questions*. 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 for the value of a specific feature on each event in the set, the set can be split into N subsets where N is the


```

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" >
. . . . .
< Inclass?(word(0), Class295), "yes" > < WordBits(Word(-1), 29), "1" >
. . . . .
< IsMember?(word(-2), Set("and", "or", "nor")), "no" > < IsPrefix?(Word(0), "anti"), "no" >
. . . . .
< Tag, "Prep-type5" >
}

```

(Basic Questions)

(WordBits Questions)

(Linguistic Questions)

Figure 4.1: Example of an Event

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

Table 4.1: Texts for Tagging Experiments

number of possible values for the feature. We can then calculate the conditional probability distribution of tags for each subset, conditioned on the feature value. After computing for each feature the entropy reduction incurred by splitting the set, we choose the feature which yields the largest 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. 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.

Texts from the Wall Street Journal corpus of the Penn Treebank Project (Marcus et al. 1994) and the ATR corpus are used 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 from many different domains, displaying language use in a very wide range of styles and settings. (Black et al. 1996).

Table 4.1 shows the sizes of texts used for the experiment. Figure 4.2 shows the tagging error rates plotted against various clustering text sizes. Of the three types of questions,

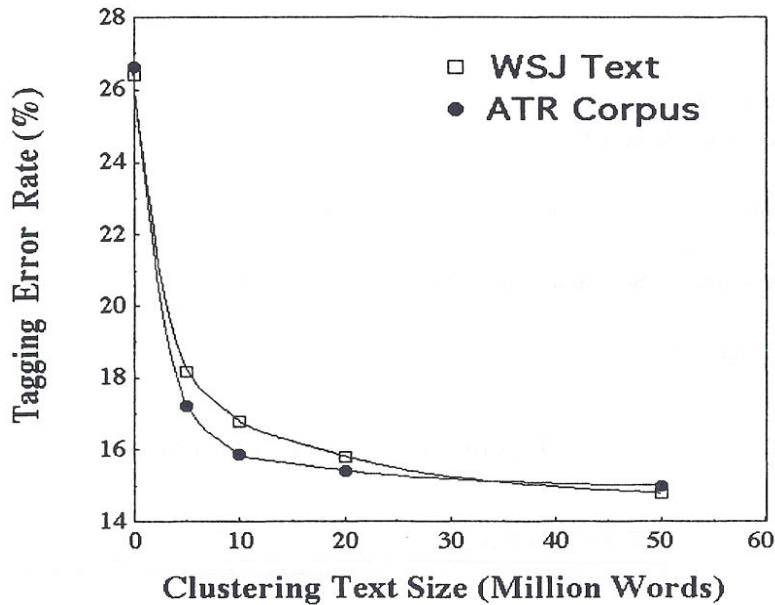


Figure 4.2: Tagging Error Rate

basic questions and word bits 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 word¹ and basic questions and word bits questions are used. The results are plotted at zero clustering text size. For both WSJ texts and the 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 in 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. So the obtained hierarchical clusters are considered to be portable across domains.

Figure 4.3 contrasts the tagging results using only word bits with the results using

¹Since 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 controlled 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.

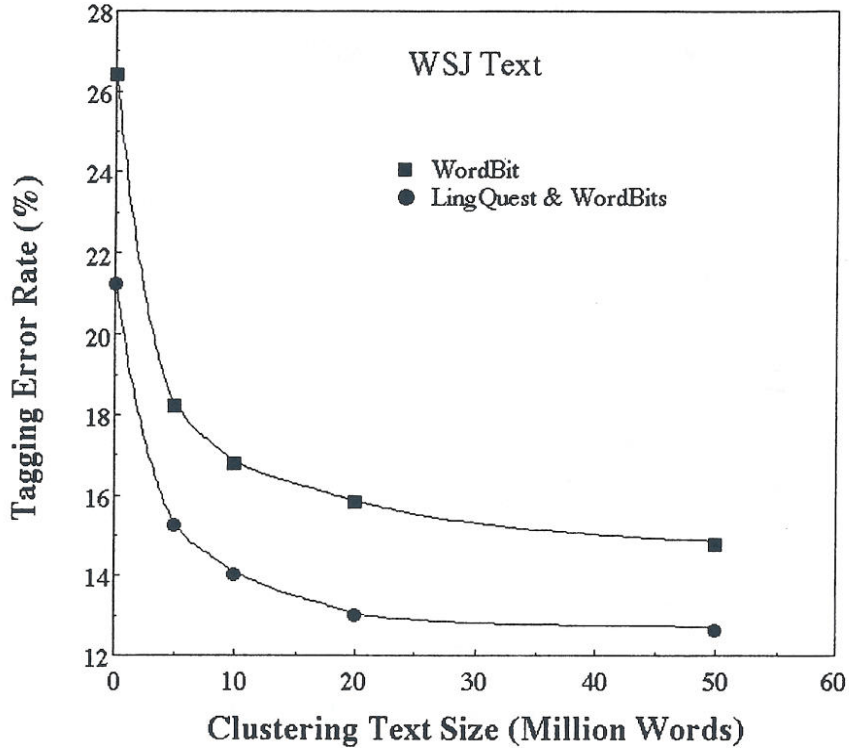


Figure 4.3: Comparison of WordBits with LingQuest & WordBits

both word bits and linguistic questions for the WSJ text. The zero clustering text size again corresponds 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.

Effect of Reshuffling

Figure 4.4 shows the tagging error rates with word bits obtained by zero, two and five rounds of reshuffling with a 23MW text. Tagging results presented in Figure 4.3 are also shown as a reference. Although the vocabulary used in this experiment is slightly different from the one used in the other experiments, we can clearly see the effect of reshuffling for

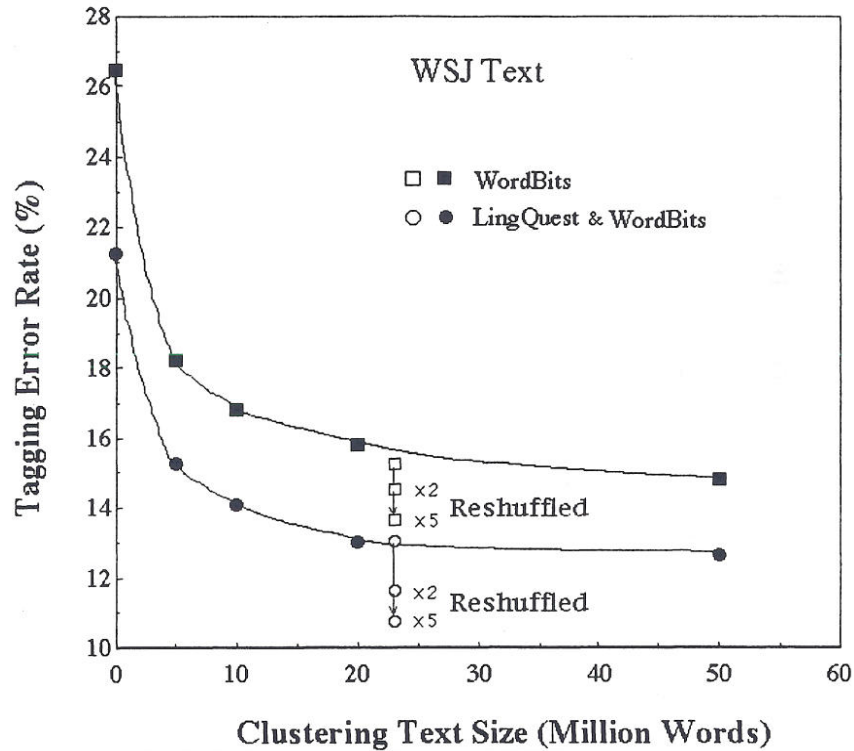


Figure 4.4: Effects of Reshuffling for Tagging

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.

4.3 HMM Tagger

4.3.1 Tagging Model

A decision-tree based NLP system is suited to take advantage of class information in terms of word bits. Many levels of class information enable the system to detect both broad and subtle word differences. However, word bits usually require unnecessarily many parameters for a system with no capability of automatically selecting meaningful features. One bit string representing a word incorporates much more information than necessary to distinguish the word from others in the vocabulary. A partition of the vocabulary, or a *flat* cluster, on the other hand, incorporates less information than necessary to distinguish the word from others, but requires many fewer parameters. This type of information is

particularly useful when there is no other information on the word.

A series of experiments are conducted to examine the usefulness of flat clusters for improving the accuracy of POS tagging of unknown words by an HMM tagger. It is well known that for a supervised POS tagging, tagging error rate for unknown words (words that never appeared in the training data) is much higher than that for known words. The aim of the experiments is to show that tagging accuracy for unknown words can be improved if the class of the unknown words is known and that the better cluster we use, the better result we obtain.

The base form of the HMM tagger is constructed following Ken Church's trigram-based POS tagger (Church 1988). In this model the joint probability of the tag sequence $T = t_1 t_2 \dots t_n$ occurring with a given word sequence $W = w_1 w_2 \dots w_n$ is given by:

$$Pr(T|W)Pr(W) = Pr(t_1)Pr(t_2|t_1) \prod_{i=3}^n Pr(t_i|t_{i-1}, t_{i-2}) \prod_{j=1}^n Pr(w_j|t_j)$$

In this formula, $Pr(t_i|t_{i-1}, t_{i-2})$ is a state transition probability which, in this model, refers to the probability that a word with tag t_i appears after seeing a tag sequence of $t_{i-1}t_{i-2}$. $Pr(w_j|t_j)$ is an emission probability which refers to the probability that a word with tag t_j turns out to be w_j . The tagger is purely probabilistic and contains no linguistic rules.

The tagger is trained and tested using manually tagged texts from the Wall Street Journal corpus of the Penn Treebank Project (Marcus et al. 1994). There are 25 directories of tagged WSJ corpus, and we used 21 directories for training and another 3 directories for testing. The training set contains about one million words and the test set contains 171 thousand words. The tagging accuracy of the basic HMM POS tagger for this data was 95.6 %, but the tagging accuracy for unknown words was as low as 62.7 %. The tagging accuracy of 95.6 % is comparable to other results in the literature (Black et al. 1992, Brill 1993). Because the training set is considerably large, only 2.48 % of the test set is unknown words (4245 words out of 171000). The low tagging accuracy for unknown words comes from the fact that for an unknown word the emission probability $Pr(w_j|t_j)$ cannot be estimated from the training corpus and thus an arbitrary small value is used instead. No lexical information is used in this case. Even if the word never appeared in the training

corpus, however, if some words of the same word class as the word in question appeared in the training corpus, we can use the emission probability of the class as an approximation. The following is the HMM tagging model which incorporates class information for tagging unknown words.

- Transition probability $Pr(t_i|t_{i-1}, t_{i-2})$ and emission probability $Pr(w_j|t_j)$ are trained from the training data.
- Class-based emission probability $Pr(class_j|t_j)$ is trained from the training data.
- Test sentences are tagged using the following model:

$$Pr(T|W)Pr(W) = Pr(t_1)Pr(t_2|t_1) \prod_{i=3}^n Pr(t_i|t_{i-1}, t_{i-2}) \prod_{j=1}^n Pr(V_j|t_j)$$

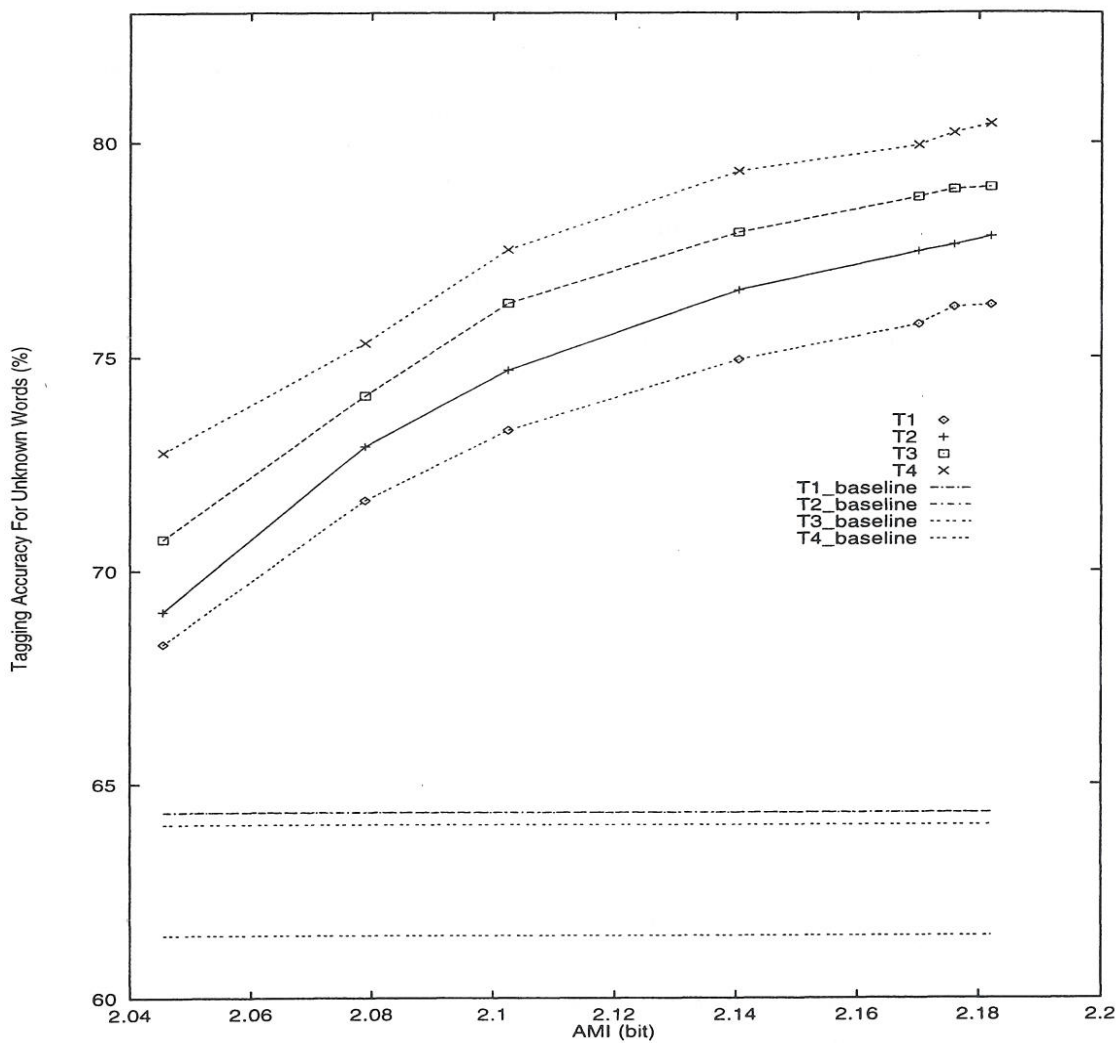
where $Pr(V_j|t_j)$ is a word emission probability when the word is a known word. Otherwise a class emission probability is used.

4.3.2 Experiments on the WSJ Corpus

Figure 4.5 shows the POS tagging accuracy for unknown words with different clusters and with varied training text size. The x-axis is the average mutual information of the entire WSJ POS-tagged corpus with respect to the clusters, and the y-axis is the POS tagging accuracy for unknown words.

Clusters used are, from left to right:

- CL1: obtained from 5 million words of WSJ corpus
- CL2: obtained from 10 million words of WSJ corpus
- CL3: obtained from 20 million words of WSJ corpus
- CL4: obtained from 50 million words of WSJ corpus
- CL5: obtained from 50 million words of WSJ corpus, and reshuffled once
- CL6: obtained from 50 million words of WSJ corpus, and reshuffled twice
- CL7: obtained from 50 million words of WSJ corpus, and reshuffled five times.



POS Training Set	T1	T2	T3	T4
Size (K Words)	1,070	758	498	245

Cluster	CL1	CL2	CL3	CL4	CL5	CL6	CL7
AMI (bits)	2.045	2.079	2.102	2.140	2.170	2.176	2.182

Figure 4.5: POS Tagging Accuracy for Unknown Words with Varied Training Sizes and Clusters

Baselines show POS tagging accuracy without using class information. Again, it can be seen that POS tagging accuracy for unknown words is considerably increased by using class information, and the better cluster (with respect to AMI measure) we use, the better result we obtain. Also, it is shown that the effect of using clusters is greater when the training data is more sparse. For example, in the case of T4 (245,000 word training data) with CL7, the accuracy is increased from 43.5 % to 77.8 % with class information. The tagging error rate became less than half (56.5 → 22.2: 39 %). It was found in this experiment that the total tagging accuracy, not only the tagging accuracy for unknown words, can be improved by this simple method, especially for smaller training sets. Figure 4.6 shows the overall tagging error rate with varied POS-tagged training text sizes.

As the training text size decreases from 1 million words to 200,000 words, the overall tagging error rate increases considerably. With class information (CL7), however, the change is much slower. For example, with a 200,000 word training text, the error rate is still restrained at 5 % when the cluster CL7 is used. It takes a training data more than twice as large to obtain the same error rate without using cluster information.

Figure 4.7 shows the POS training text size dependency of

1. tagging accuracy for unknown words
2. tagging accuracy for unknown words which belong to some class of CL7 (i.e. words which never occurred in the training data but whose class is known)
3. tagging accuracy for unknown words which belong to no class in CL7 (i.e. unknown words which have no class information)
4. the ratio of occurrences of unknown words which belong to some class of CL7 (i.e. out of all the occurrences of unknown words, how many of them have class information)

It is clear from this figure that the tagging accuracy for unknown words is higher for smaller training texts because more unknown words have class information. It can also be inferred from this figure that the upper bound of tagging accuracy for unknown words in this tagging model is about 83 %. Note that in this model class information is used only for unknown words.

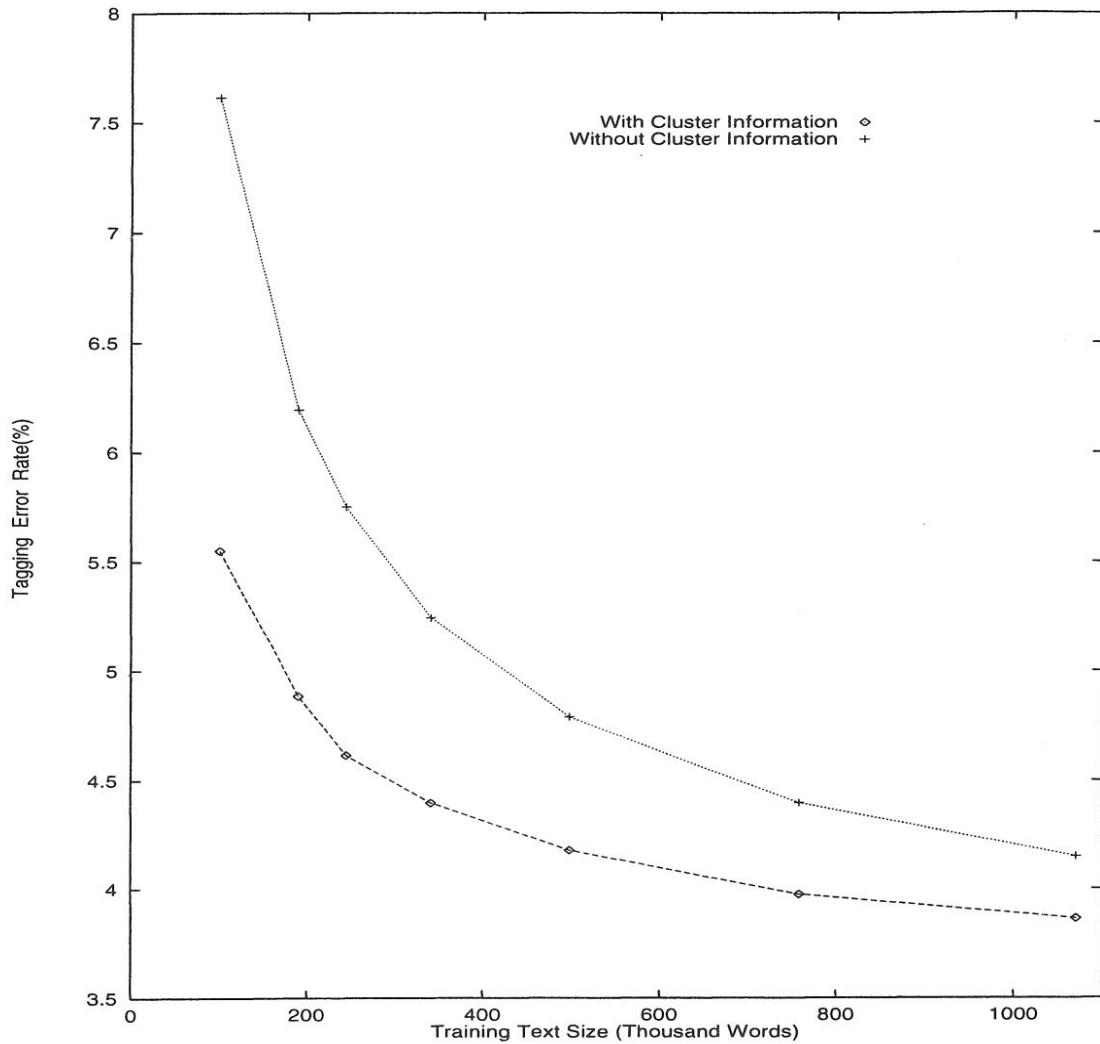


Figure 4.6: POS Tagging Accuracy for All Words with Varied POS-Tagged Training Text Sizes

4.4 Domain Dependence

The major utility of clusters lies in their generalization power. Predicting the distributional characteristics of a word through the characteristics of its class is one example of using the generalizability of clusters. Another important aspect of generalizability is cross-domain portability, that is, how useful the clusters created in one domain are for processing texts in another domain. In order to investigate the cross-domain portability of clusters, texts from Associated Press newswire articles were used as clustering texts and the obtained clusters were applied to the POS tagging task of WSJ articles.

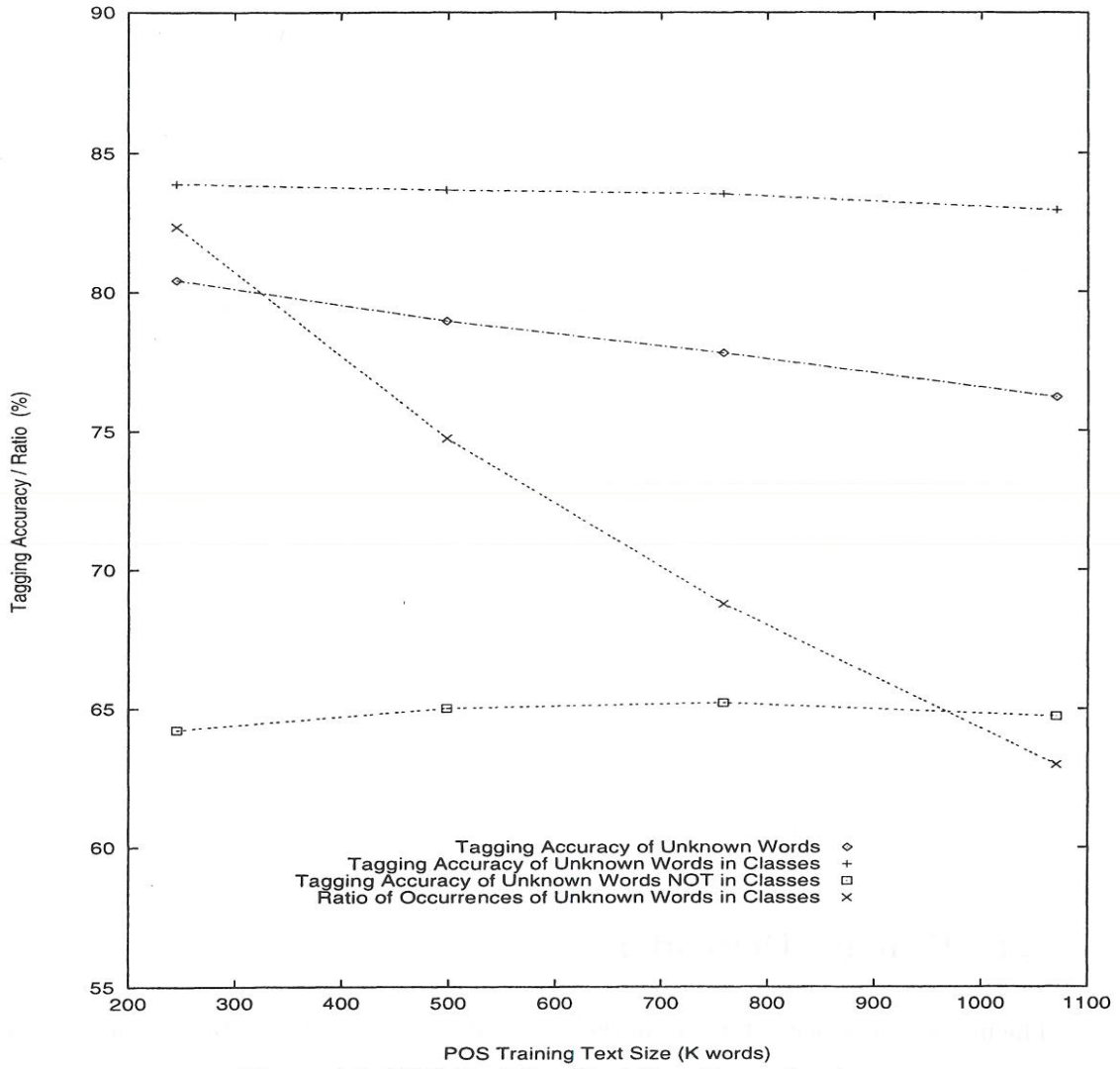


Figure 4.7: POS Training Text Size Dependencies

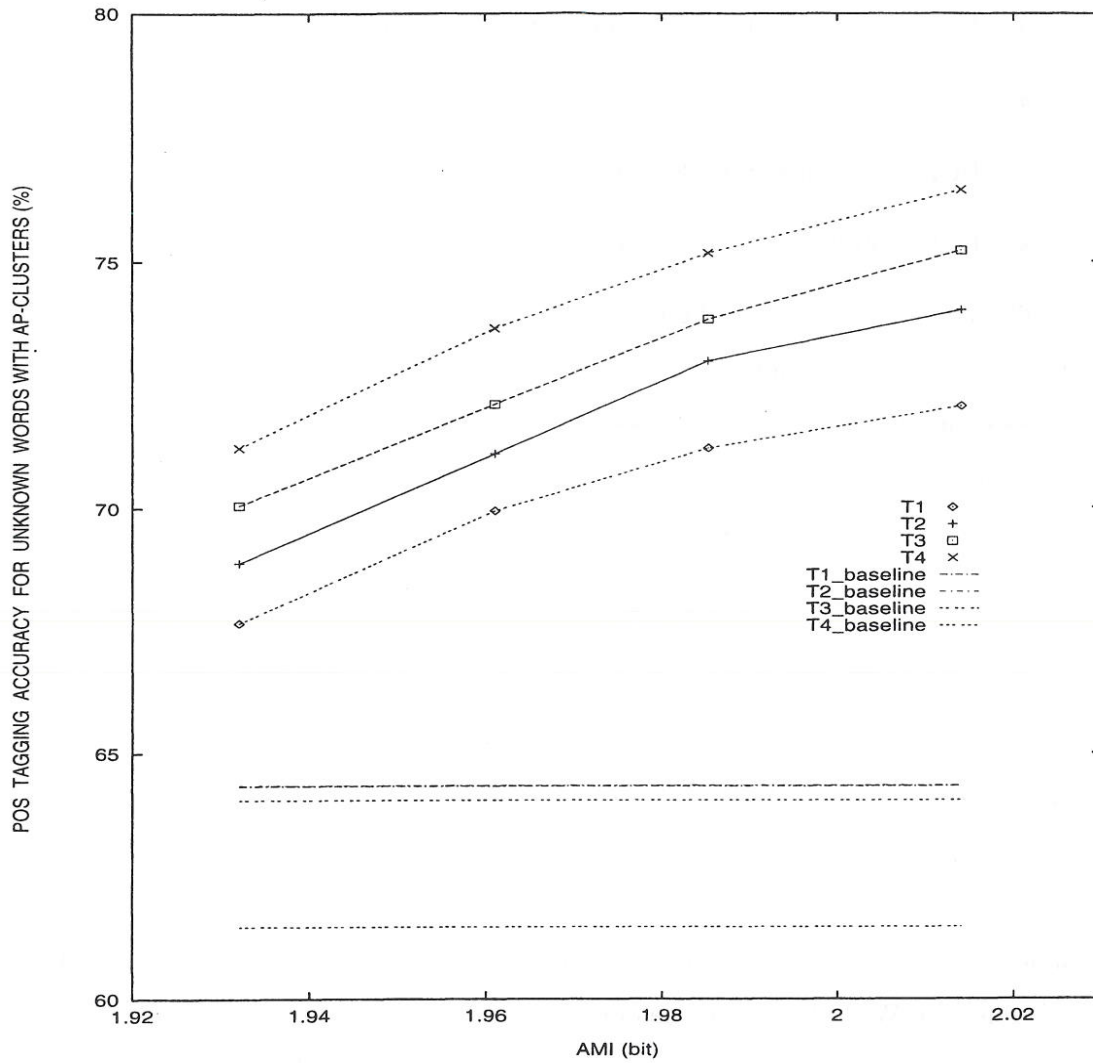
The following clusters are created using 39.4 million words of 1990 Associated Press newswire:

- APCL1: obtained from 5 million words of AP newswire
- APCL2: obtained from 10 million words of AP newswire
- APCL3: obtained from 20 million words of AP newswire
- APCL4: obtained from the entire 1990 AP newswire (39.4 million words)

The vocabulary clustered here is the set of the 70,000 most frequently occurring words in the entire 1990 AP newswire. Figure 4.8 shows the POS tagging accuracy for unknown words with different clusters and with varying training text size. As in Figure 4.5, baselines show POS tagging accuracy without using class information. Compared with Figure 4.5, the POS tagging accuracy for unknown words is 3 to 6 % lower for each training text, but still the improvement of the accuracy over the baseline is significant. This suggests that clusters created in one domain are useful in another domain if both are at least in the same category of news articles.

Another way of examining the difference of the effect of clusters across domains is to compare the AMI of texts from different domains with respect to clusters obtained in one of the domains. Figure 4.9 shows the AMI of 1990 AP newswire with respect to WSJ clusters plotted against the AMI of 50 million words of WSJ articles with respect to the same clusters. The relation between the two is almost linear and this also suggests the portability of WSJ clusters to the domain of AP newswire. The linearity of the plots also indicates that over-learning or over-fitting of clusters to the WSJ texts is not occurring in the range of this figure. One of the reasons may be that the number of the classes in the cluster, 500, is small enough to alleviate over-learning.

Another notable point in Figure 4.9 is that the effect of reshuffling is significant. The increase of AMI achieved by reshuffling 5 times using a 50MW clustering text is twice as large as the increase of AMI caused by increasing the text size from 20MW to 50 MW. However, as will be shown later (Figure 4.11), the effect of word-by-word reshuffling begins to saturate around 5 iterations. We do not know of any other procedures to further increase AMI.



POS Training Set	T1	T2	T3	T4
Size (K Words)	1,070	758	498	245

Cluster	APCL1	APCL2	APCL3	APCL4
AMI (bits)	1.932	1.961	1.985	2.014

Figure 4.8: POS Tagging Accuracy for Unknown Words with AP-Clusters

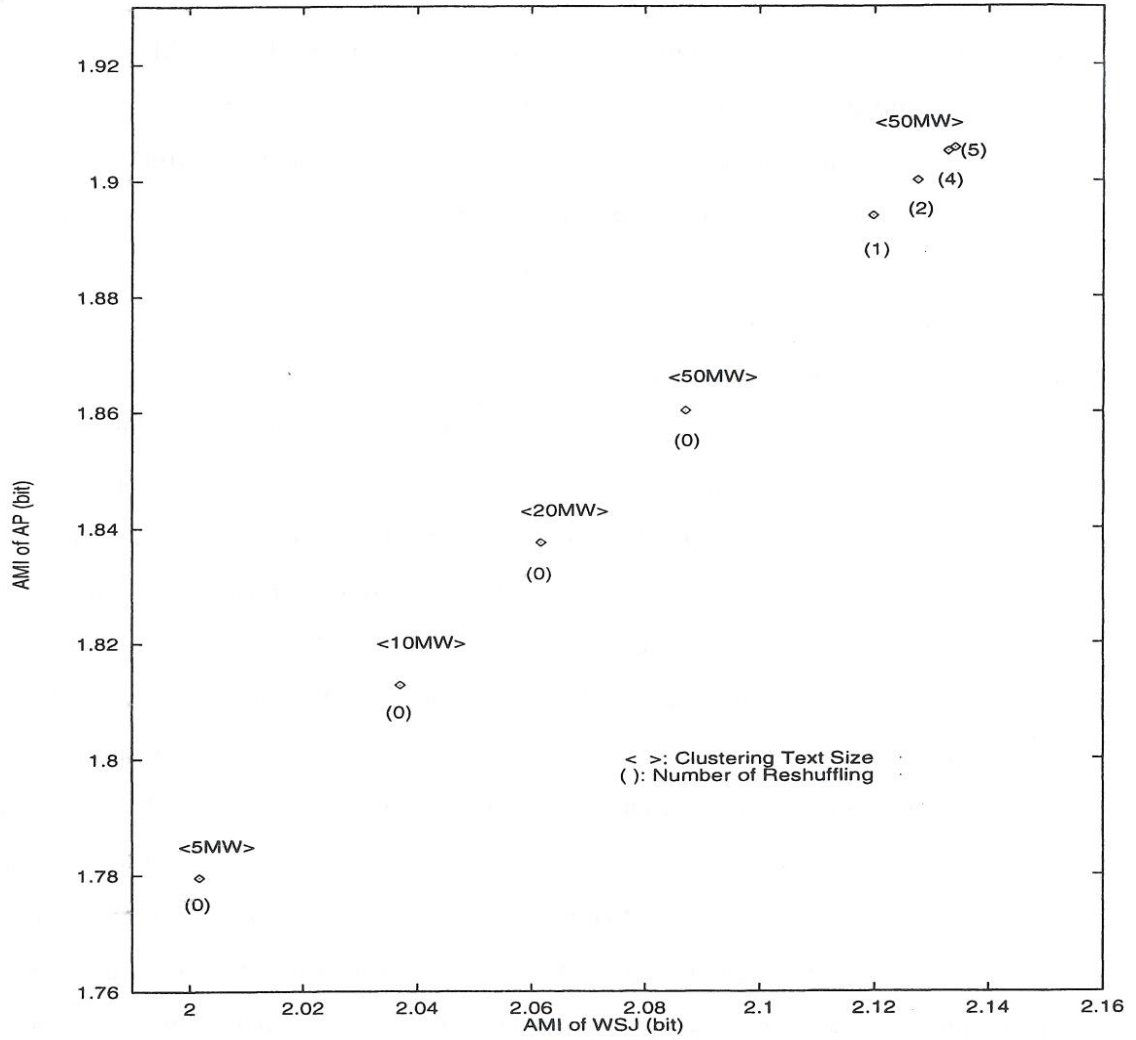


Figure 4.9: Domain Dependency

4.5 Split & Merge Clustering

We have so far seen that word-by-word reshuffling is a promising candidate as a way to improve cluster quality. We discuss here alternative ways to improve cluster quality and quantitatively compare them. One way would be to move two words at a time in the process of reshuffling. But this dramatically increases the computational burden. Another way would be to introduce simulated annealing. Boltzmann Machine-like annealing, however, is computationally intensive for a very large vocabulary. A new method we propose here is a kind of deterministic annealing, which does move multiple items at a time, but the group of items to be moved together is neither searched extensively nor generated in a random process as in simulated annealing.

The basic idea is based on the observation illustrated in Figure 4.10. We first create a hierarchical cluster using 50MW of WSJ text following the steps presented in § 3.3.1. The number C of classes in step 1 (MI-clustering) of § 3.3.1 is 500. When we construct dendrograms in step 2 (outer-clustering) and in step 3 (inner-clustering), we assign the loss α in AMI to the corresponding branch of the tree. For example, if merging some pair of classes (C_1, C_2) induces minimum MI loss among all possible pairs of classes, we merge C_1 and C_2 to create new class C_3 , and we assign the corresponding MI loss value to the branch (C_1, C_2) , or equivalently, to the node C_3 . In this way, we can construct a hierarchical cluster in the form of a binary tree whose branches have corresponding MI loss values (α). Out of such a hierarchical cluster, we can then construct clusters (partitions of the vocabulary) of an arbitrary granularity using a threshold value θ for α 's. Given a hierarchical cluster and a threshold value θ , we search, in a bottom-up manner, such nodes (say B nodes) whose α value is smaller than θ but whose parent nodes are greater than or equal to θ . Then the set of all such nodes (B 's) constitutes a partition of the vocabulary in such a way that all leaf nodes (i.e. individual words) which are descendants of node B are in the same class represented by node B .

Figure 4.10 shows the number of classes (B 's) created in this way plotted against the threshold θ in a log-log scale. The linearity in this figure can be interpreted in such a way that a linear function of $\log(\theta)$ represents the depth of the tree (hierarchical cluster). When θ is very small, we have a large number of very small classes, and all elements in a class

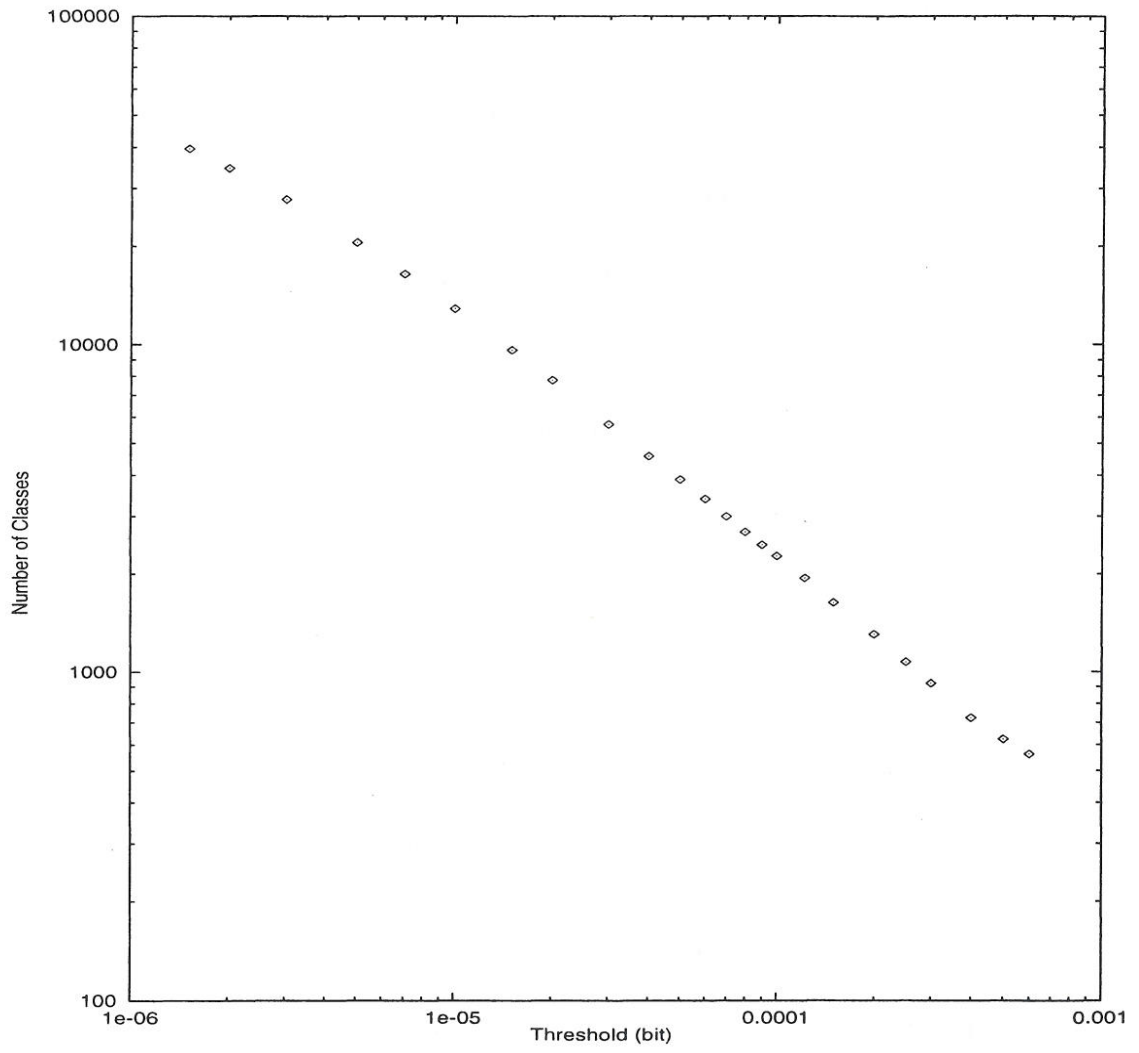


Figure 4.10: Threshold Value vs. Number of Classes

are considered to be quite correlated in terms of mutual substitutability. For example, when $\theta = 1.0e-5$ (bits), we have 12876 classes, and {after-tax, aftertax} constitutes one class and {pretax, pre-tax} constitutes another class. This motivates us to bind elements of those “microclusters” together and move “microclusters” instead of individual words while reshuffling. So the following is the new method we propose.

Given an initial cluster with C classes of words,

1. set $\theta = \theta_{begin}$ and iterate 2 through 8 while $\theta < \theta_{end}$
2. conduct hierarchical clustering using the current cluster with C classes
3. create a set of microclusters with θ as described above
4. conduct outer-clustering on the microclusters until C classes remain
5. reshuffle microclusters among C classes of microclusters until no movement increases AMI
6. for each class of microclusters, merge elements (words) of all the microclusters in the class and create one big class of words (then we have C classes of words)
7. conduct one round of word-based reshuffling on the result of 6
8. set $\theta = [\theta \text{ times (some constant } > 1)]$

We will call this method “Split&Merge Clustering” because we repeat the step of splitting C classes of words into microclusters followed by the step of merging microclusters back to C big classes. Steps 2 through 8 constitute one iteration.

4.5.1 POS Tagging of Unknown Words

Because the reshuffling step (step 5) is incorporated inside the routine, Split&Merge Clustering converges faster than word-by-word reshuffling. As an example of this method, we conducted Split&Merge Clustering with three iterations. The values of θ were 4.0×10^{-5} , 1.0×10^{-4} , and 2.5×10^{-4} for each iteration and the numbers of corresponding microclusters were 4504, 2195, and 1060, respectively. Figure 4.11 shows the value of AMI after each iteration for Split&Merge Clustering and for word-based reshuffling. Note that

Split&Merge Clustering can achieve AMI values which are well above the upper-limit of word-based reshuffling. As far as Mutual Information-based clustering is concerned, this new method performs better than any conventional method, with the possible exception of the computationally intensive simulated annealing method. It is also expected that the Split&Merge Clustering method can be used with global objective functions other than MI as long as construction of hierarchical clusters is possible using these functions.

Figure 4.12 shows the AMI for the same clusters as in Figure 4.11 for 1990 AP newswire. Again, Split&Merge Clustering achieves higher AMI values with no sign of over-fitting.

Figure 4.13 shows POS tagging accuracy for unknown words using the same clusters as in Figure 4.12. This figure illustrates the practical applicability of the clusters created by Split&Merge Clustering.

4.5.2 Number of Classes

One of the fundamental problems with MLE-based models is that the more parameters the model includes, the better it can fit the training data. This is the over-training problem. To prevent over-learning, some external constraints on the number of parameters might be incorporated. In the case of MI clustering, we can obtain higher AMI with larger number of classes, so some constraints on the number C of classes might be effectively introduced. However, the number C of classes is a parameter which must be predefined before starting the clustering process, so we have to repeat the whole clustering process for each number of classes. Moreover, there is no guarantee that the optimal number of classes in terms of the clustering optimization criterion with a particular type of constraint is also optimal in terms of the NLP tasks on which the class-based approach is applied.

A better approach, which we propose here, is to create clusters of varying sizes at the computational cost of one full clustering step, without repeating MI clustering for each C , and to choose desirable numbers of classes for a specific NLP task. Actually, this part of the new method has already been presented in Section 4.5. Given an initial hierarchical cluster, we just create sets of classes following step 3 in Section 4.5 with varying values of θ . Each value of θ corresponds to a different number C of classes.

Using the hierarchical cluster created in the experiments in Section 3.3.3 as an initial

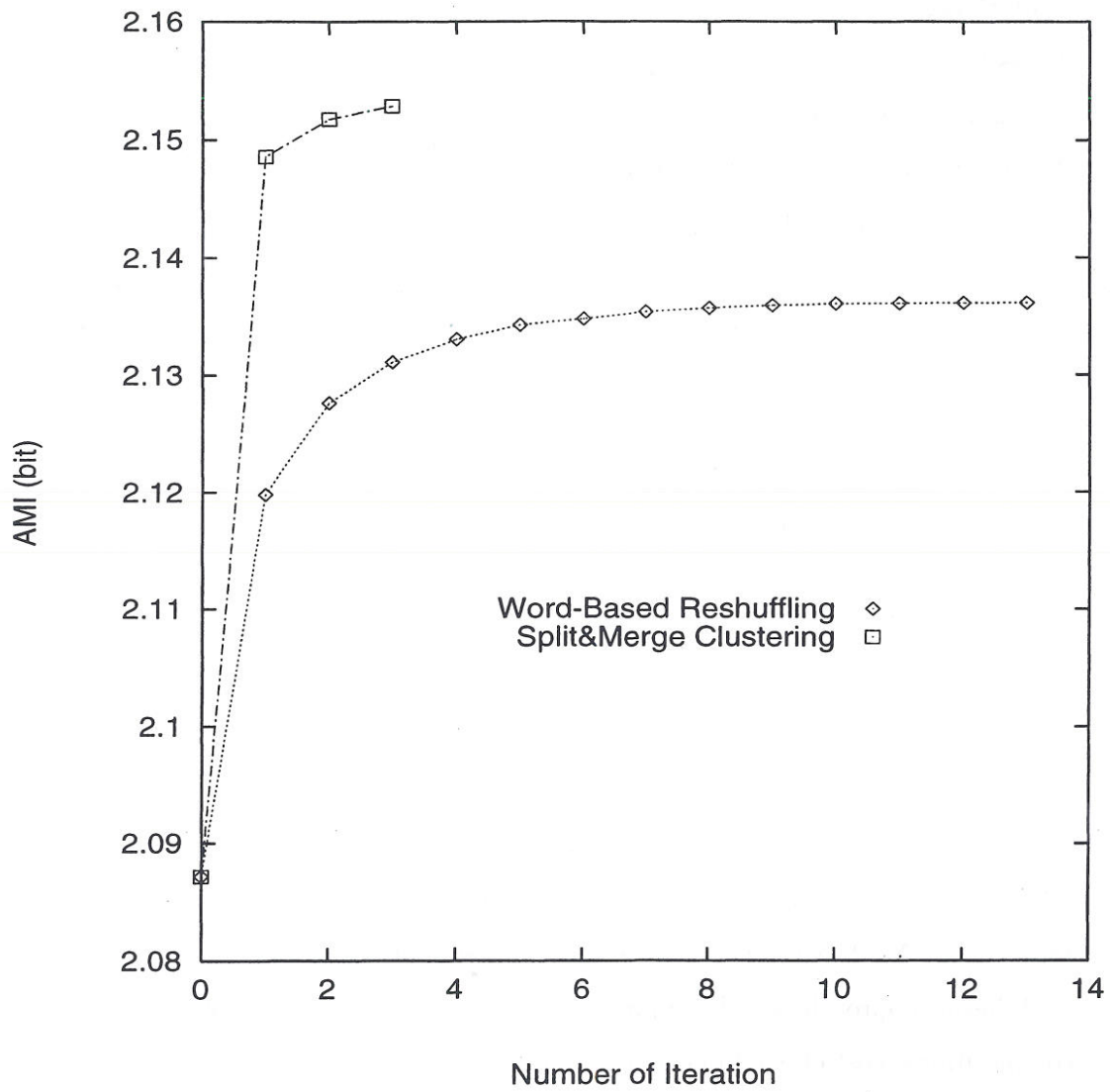


Figure 4.11: Example of Split&Merge Clustering Performance

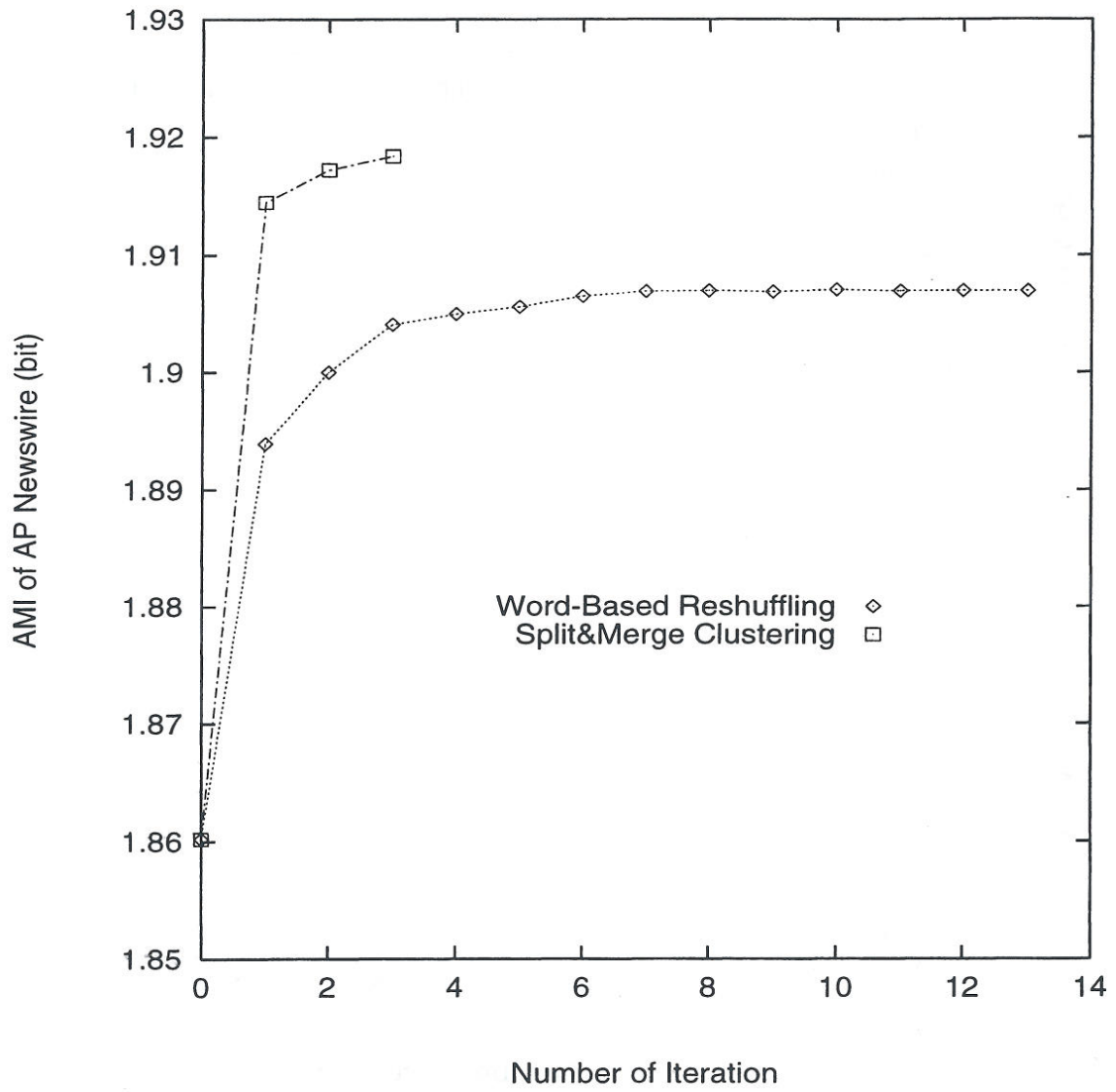


Figure 4.12: AMI of AP Newswire with WSJ Clusters

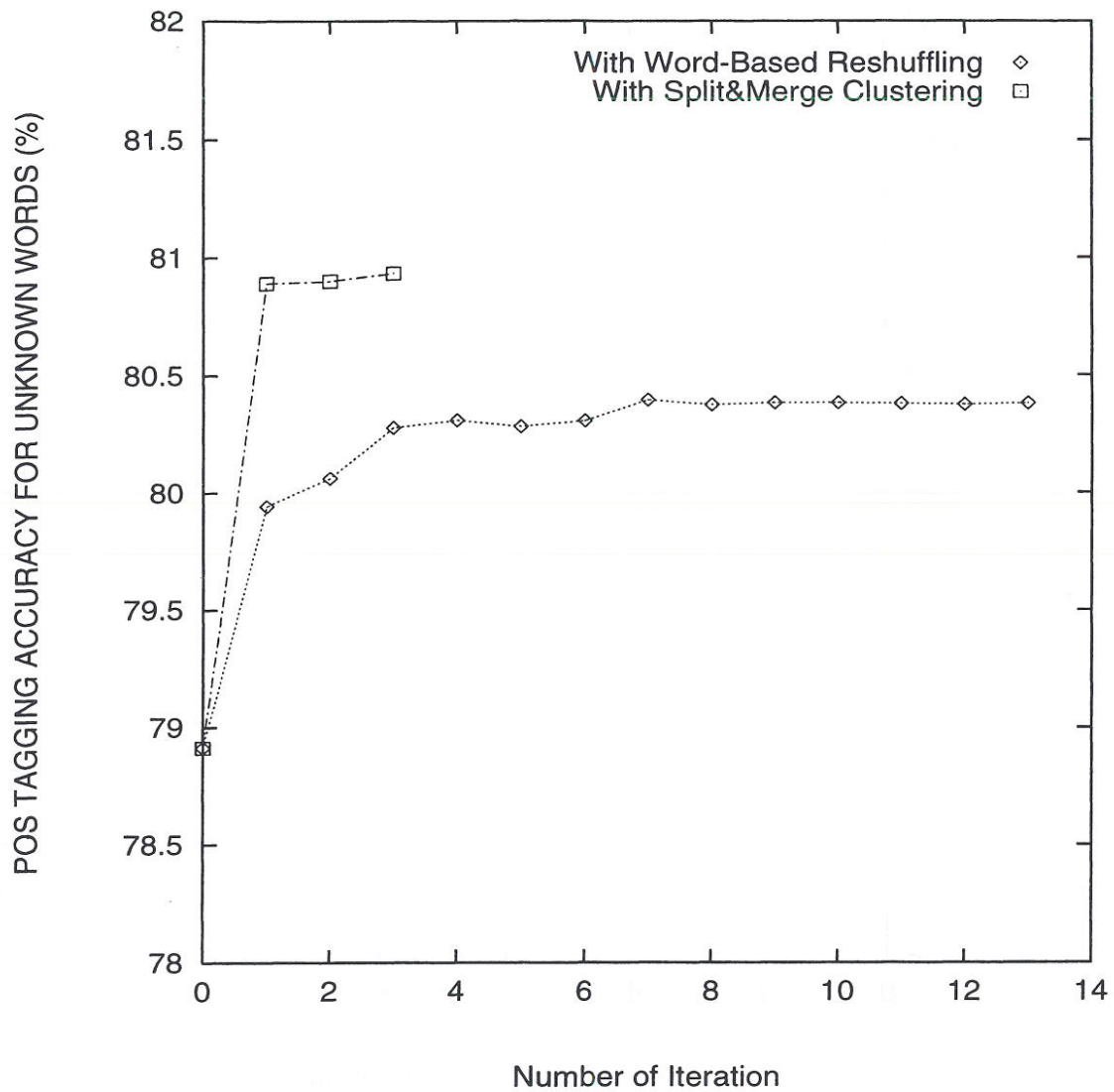


Figure 4.13: POS Tagging Performance for Unknown Words

cluster, clusters of varying number of classes from one to 2000 are created and class-based POS tagging accuracy for unknown words is evaluated for each class (Figure 4.14).

For the evaluation, five sets of text, each containing 4000 sentences (around 96,500 words), are extracted from the POS-tagged WSJ corpus, and each set is used as training data. For each training set remaining four sets are separately used as a test set, and the average accuracy of the 20 combinations of training and test sets are plotted in the figure. The graph shows that the tagging accuracy saturates at around 1000 classes. To compare the result with regular MI clustering, a separate set of clusters with varying numbers of classes is created by the MI clustering method (step 1 of Section 3.3.2). Due to the quadratic growth of computation time and memory with the increase of the number of classes, MI clustering with class numbers of more than 1500 were not attainable. Although there is still a slight increase in tagging accuracy between 1000 and 1500 classes for MI clustering, Split&Merge Clustering at 1000 classes performs best in the figure. Two important points are made clear in this section. First, Split&Merge Clustering provides a simple solution to the problem of determining the number of classes, and secondly, higher performance of an NLP task can be achieved with clusters created by Split&Merge Clustering than clusters created by MI clustering.

4.6 Comparison with Previous Work on Part-Of-Speech Tagging

A considerable amount of work has been presented in the literature on POS tagging, but we will restrict discussion in this section to reports on POS tagging of the WSJ corpus among which direct quantitative comparison is possible.

Weischedel et al. (1993) proposed an HMM-based POS tagging model in which information on word spelling is directly incorporated as model parameters for disambiguating unknown words. In this model the joint probability of the tag sequence $T = t_1 t_2 \dots t_n$ occurring with a given word sequence $W = w_1 w_2 \dots w_n$ is given by:

$$Pr(T|W)Pr(W) = Pr(t_1)Pr(t_2|t_1) \prod_{i=3}^n Pr(t_i|t_{i-1}, t_{i-2}) \prod_{j=1}^n Pr(w_j|t_j)$$

For tagging unknown words, the emission probabilities of a word given a tag are assumed

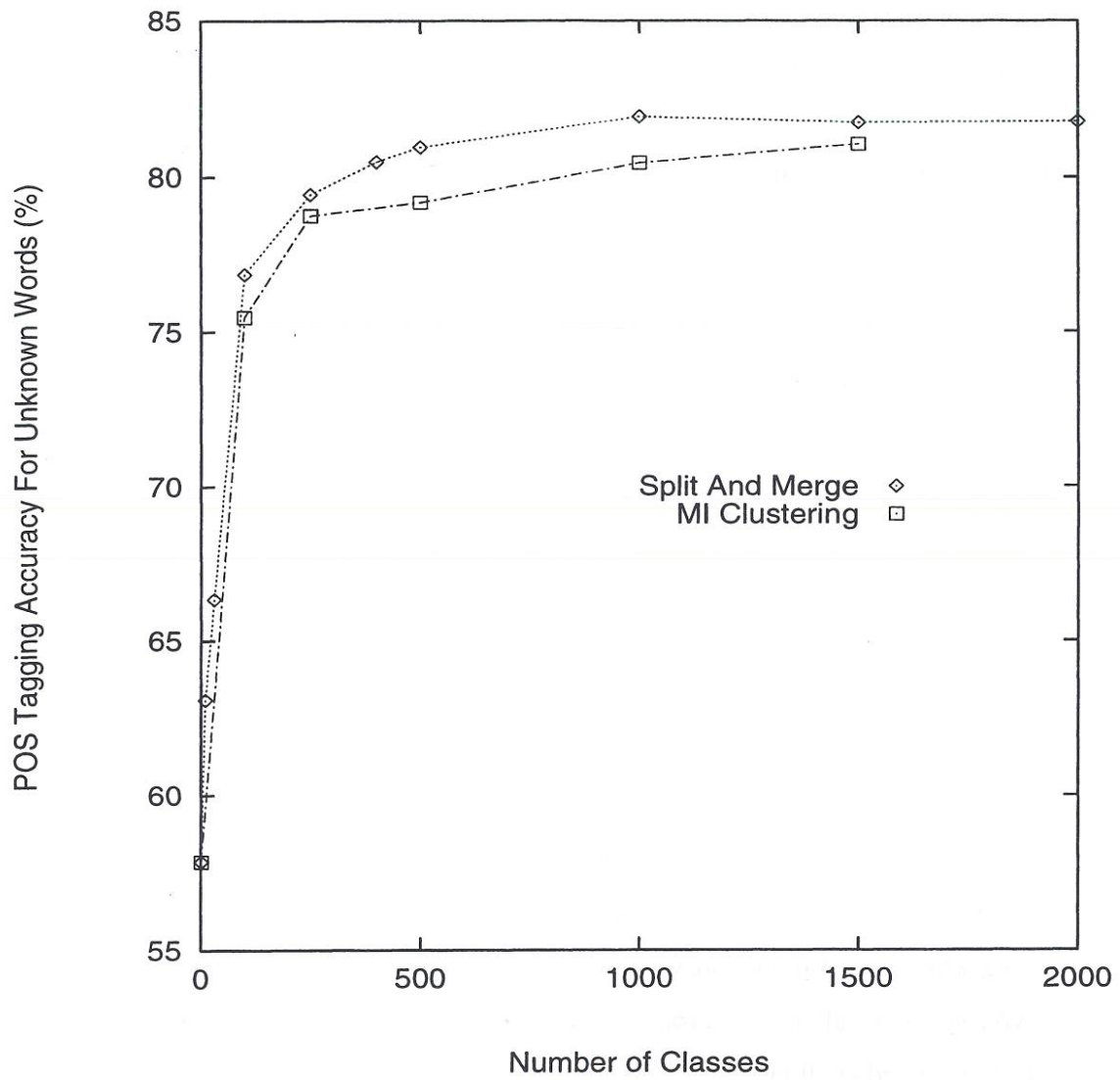


Figure 4.14: POS Tagging Accuracy with Varying Cluster Sizes

to be proportional to the product of the probabilities of specific morphological forms:

$$Pr(w_i|t_i) = Pr(\text{unknown-word}|t_i)Pr(\text{capital-feature}|t_i)Pr(\text{endings, hyphen}|t_i)$$

In this model, the only lexical information used for tagging unknown words is morphological features of words.

Brill (1993) proposed a transformation-based POS tagging method. The method is rule-based, although it incorporates numerical values to score and order candidate rules to be fired. There are two types of rules (or transformations): lexical transformations, which are used to learn the most likely tag for each word, and context-triggered transformations which are used to improve the tagging performance by adding contextual information. Both types of transformations are automatically learned from the training corpus by instantiating transformation templates which are compiled manually. Out of the 16 lexical transformation templates Brill used, 14 refer to spelling features like suffixes and prefixes.

Brill (1993) comparatively evaluates the performances of a transformation-based POS tagger (Brill tagger) and a probabilistic tagger on small training corpora. The first 1,000 sentences (about 23 thousand words) of the POS-tagged WSJ corpus were used for the lexical training of the Brill tagger and the second 1,000 sentences were used for the training of contextual transformation rules. The last 2,787 sentences (about 65 thousand words) of the WSJ corpus were used as a test corpus. To compare the performance of the Brill tagger with that of a probabilistic tagger, Brill re-implemented Weischedel's HMM tagger (HMM-MORPH). (a) and (b) in Table 4.2 are cited from Brill's report on the comparison of the Brill tagger with Weischedel's HMM tagger. In order to compare these results with the performances of other approaches, we prepared the same training and test data from the WSJ corpus as described in Brill's report. To check the consistency, the Brill tagger was run on the prepared data ((c) in Table 4.2) and it was confirmed that our experimental setting is the same as in Brill's report. Note that we used 2,000 sentences (1,000 sentences for training lexical rules and 1,000 sentences for training contextual rules) in total to train the Brill tagger. Since (a) in Table 4.2 (HMM-MORPH) shows the result with only the training text of 1,000 sentences, however, we split the first training text of 1,000 sentences into two 500-sentence texts and reran the Brill tagger with the first 500-sentence text for

lexical training and the second 500-sentence text for contextual training. (d) in Table 4.2 shows the result. The Brill tagger still outperforms HMM-MORPH for tagging unknown words. In all of the following experiments in this section, training and testing are carried out on the same corpora as were used above, that is, the text of 1,000 sentences, called Corpus-A, for training and the last 2,787 sentences of the POS-tagged WSJ corpus for testing.

Ratnaparkhi (1998) applied the Maximum Entropy (ME) framework to the task of POS tagging. The ME framework provides a way of obtaining the most unbiased probability distribution of a variable among a set of probability distributions that satisfy certain constraints on probability distributions. In Ratnaparkhi's ME model for POS tagging, the probability distribution whose entropy is maximized is the conditional probability of a tag given the context of the current word, and the constraints are expressed in the form of feature expectations whose values are given as observed expectations of the features in the training data. A feature can refer to any information that might help to predict a tag, such as spelling of the current word or tags of the surrounding words. For tagging rare words including unknown words, special features are used that encode spelling characteristics of the words such as prefix, suffix, (upper) case, and hyphenation.

Ratnaparkhi's ME POS tagger is run on the same corpora as used above ((e) in Table 4.2).

(f) and (g) in Table 4.2 show the performance of the HMM-based tagger described in Section 4.3.1 without and with class information for tagging unknown words, respectively. Although ME performs slightly better than HMM-CLASS for unknown words, HMM-CLASS performs best in total word accuracy.

Combining Distributional and Morphological Features

One distinctive feature of HMM-CLASS is that it does not use morphological (spelling) features of words at all. In all the other approaches discussed above, morphological features of words play a central role for tagging unknown words. A natural question is whether we

	Unknown Word Accuracy	Total Word Accuracy
(a) HMM-MORPH (cited)	<i>71.7</i>	<i>91.0</i>
(b) Transformation-Based (cited)	<i>81.2</i>	<i>92.7</i>
(c) Transformation-Based (2000 sentences for training)	81.18	92.90
(d) Transformation-Based (1000 sentences for training)	79.52	90.82
(e) Maxmum Entropy	80.54	91.67
(f) HMM	49.74	86.55
(g) HMM-CLASS	79.01	91.98
(h) HMM-CLASS + DTree (without context)	82.78	92.67
(i) HMM-CLASS + DTree (with context)	83.48	92.80
(j) HMM-CLASS + ME (without context)	84.03	92.90
(k) HMM-CLASS + ME (with context)	85.54	93.17
(l) HMM + DTree (without context)	74.94	91.12
(m) HMM + DTree (with context)	73.41	90.84

Table 4.2: Comparison of POS Tagging Accuracies

can combine class information and morphological information to obtain higher accuracy. One way of combining the two is to incorporate class features and morphological features directly into the process of searching the best tag sequences. Another way, which is chosen here, is as follows. We split the original training corpus (Corpus-A) into 90 % training data and 10 % held-out data and conduct HMM-CLASS tagging with 10-fold cross-validation. The parameters tuned by cross-validation are those of a learner which learns error patterns of tagging unknown words using HMM-CLASS. Morphological features of unknown words, as well as the tagging results of the base tagger (HMM-CLASS), are used as features of the learner. We tested two types of learner. One is a decision tree learner for which the commercially available C4.5 (Quinlan 1993) is used. Another learner is based on a ME model that we implemented along the lines described in Rosenfeld (1994). For each of the learners, we trained and tested the learner with two types of error patterns. One is simple error patterns which contain a set of morphological features of an unknown word to be tagged and the predicted tag by the base tagger (HMM-Class) along with the correct tag for training. The other error patterns additionally contain context information, that is, predicted tags of words immediately before and after the unknown word to be tagged. (h)-(k) in Table 4.2 show the result. HMM-CLASS+ME+context performs best and its accuracy for unknown words is higher than HMM-CLASS by 6.5 percentage points. The error rate for unknown words of HMM-CLASS+ME+context is 25.7 % smaller than that of ME and 29.4 % smaller than that of the Brill tagger. This clearly shows that a considerable benefit is obtained by combining distributional characteristics of words (expressed in the form of clusters) and morphological features of words. As a controlled experiment, 10-fold cross-validation is also conducted with HMM tagging with no class information (HMM+DTree). (l) and (m) show the results of decision tree learners. HMM+DTree performs slightly better than HMM-MORPH, but worse than HMM-Class and far worse than HMM-Class+DTree. This result also shows that these models work better together.

4.7 Conclusion

This chapter has demonstrated that word clusters created with cross-entropy as an objective function can improve performance of POS tagging tasks, especially for rare events such

as tagging unknown words or tagging with small training data. It also has demonstrated that improvement of clusters in terms of cross-entropy (or equivalently AMI in this case) is reflected in the improvement of the NLP task. Purely distributional characteristics of words expressed in the form of clusters has been shown to be as useful as morphological features of words in POS tagging tasks, and moreover, the two apparently orthogonal types of features can be combined to produce considerably better performance than any single type of features. The combination of clusters and spelling features of words has been shown to have an error rate of tagging unknown words which is at least 25 % lower than that of other previous approaches examined here, including an HMM tagger with only morphological information for unknown words, a transformation-based POS tagger, and a Maximum-Entropy model. Future work includes application to other NLP tasks and direct incorporation of the class information into various NLP engines.

Chapter 5

From Word Clustering to Compound Clustering

5.1 Introduction

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-classes¹ instead of a series of words, we can constrain the explosion of the number of compounds. One approach to this is to bundle quite similar compounds in a small subclass and treat them as a single compound. Suppose that we have a set of word classes and that some word class, say WC397, contains almost exclusively first names, and another class, say WC381, contains almost exclusively family names. Then the chain of classes “WC397_WC381” 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 will describe one approach to this goal which is completely automatic and very fast.

5.2 Compound Clustering Method

One approach for compound clustering consists of the following three steps:

¹We use the term *word-class* for a class of words to make a clear distinction from a *compound-class*.

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 the “stickiness” of two classes, and identify which class pair 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

$$MI(C1, C2) = \log \frac{Pr(c_1 c_2)}{Pr(c_1)Pr(c_2)} \geq \theta \quad (5.1)$$

for some threshold θ .

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. ²

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 will 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 will call

²The 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 can be represented by a series of (word, word-class) pairs and no conversion actually needs to be carried out.

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.

5.3 Improvement of Compound Clusters

The compound clustering method described above uses a simple linear concatenation of classes for constructing longer compound patterns. An advantage of this method is that it is fast; it takes only linear time with respect to the text size and no iteration is needed.

A more elaborate method is to iterate the process of binary concatenation as follows. Each step of binary concatenation involves either one instance of concatenating two items or one round of the binary concatenation process through all possible pairs of items. After each step of binary concatenation, the identified pairs are added to the token vocabulary as new entries, and mutual information for the new token vocabulary is recalculated. Note here that a word class is a minimum unit of items, and the initial token vocabulary is simply a set of word classes. Although this iterative process is far more time consuming than the above method, this class-based approach is expected to significantly reduce the computational burden, and possibly produce a better result than the above method.

5.4 Evaluation of Compound Clusters

It is not a simple task to quantitatively evaluate the quality of obtained compound clusters for NLP tasks. Perhaps the best way is to integrate the obtained compound clusters into an existing NLP system like a parser or an example-based machine translation system and evaluate the performance of the system. Since no such system is available, however, this kind of evaluation is out of the scope of this work. Another way of evaluating the obtained compound clusters is to evaluate the coherence of each compound cluster. In a preliminary experiment which will be described in the next section, it was found that

some of the compound classes are semantically coherent. Such compound classes include classes of proper names such as organization names and person/place names. For those classes, direct precision measurements can be made. Also as in the case of word clusters, a linguistically intuitive evaluation will be given by illustrating lists of obtained compound classes with varied clustering conditions.

5.4.1 Compound Clustering Experiment

Plain texts from two years (1987 and 1988) of the Wall Street Journal Corpus were used to create compound clusters. The compound clustering method described in Section 5.2 was used. The total volume amounts to about 40 million words 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 θ in equation 5.1. We used $\theta=3$ following “a very rough rule of thumb” used for word-based mutual information in (Church and Hanks, 1990).

Out of the 40MW text, 7,621 distinct class chains and 287,656 distinct multiword compounds are extracted. To construct a new vocabulary, we selected the words and tokens that appeared more than four times in the reduced text. The size of the new vocabulary is 60,589 and it contains 4,685 class chain tokens. Some of the compound-classes that were obtained are shown in Table 5.1 through Table 5.5. Figures on the left of each cell represent frequencies of the compound in the entire corpus and the compounds are listed in descending order of frequency in each class, and the lists are truncated at an arbitrary point.

Table 5.1 shows most frequent compounds in Compound-class-171 and Table 5.2 is an excerpt from the least frequent part of the same compound class. Compound-class-171 consists of names with titles many of which are politicians’ names. Note that compounds that appeared only once in the entire corpus of 40MW can be quite reliably and efficiently extracted. An interesting example is that *Interior_Minister_Friedrich_Zimmermann* and *Interior_Minister_Friedrich_Zimmerman* should be the same person and both are extracted even though they appeared only once. Compound-class-179 (Table 5.3) contains multiword company names. Compound-class-221 (Table 5.4) consists of multiword compound nouns

4390 President_Reagan	2956 Mr._Reagan
2742 Mr._Bush	2137 Mr._Dukakis
960 Judge_Bork	884 Ronald_Reagan
846 George_Bush	645 Michael_Dukakis
483 Treasury_Secretary_James_Baker	477 Mr._Holmes
465 Vice_President_George_Bush	456 Gov._Dukakis
453 Gen._Noriega	450 Mrs._Thatcher
416 someone_who	388 Mrs._Aquino
366 Mr._Roh	358 Gen._Secord
343 Mr._Lawson	342 Adm._Poindexter
324 anyone_who	323 Mr._Dole
296 Lt._Col._North	288 Jimmy_Carter
270 Sen._Dole	265 Mr._Mulroney
261 Mr._Quayle	260 Sen._Bentsen
249 Mr._Chirac	241 Mr._Gephardt
237 Mr._Marcos	232 Vice_President_Bush
228 Sen._Quayle	228 Mr._Carter
224 Mr._Chun	223 Prime_Minister_Margaret_Thatcher

Table 5.1: Compound Class 171 (high frequency part)

from several specific semantic domains including money, surgery and natural environment, but most of the frequent compounds are money-related terms. Compound-class-256 (Table 5.5) is worth special attention because although single words and multiword compounds are mixed almost evenly in the high-frequency part, most of the single words are abbreviations of organizations, mostly public organizations, and the multiword compounds also almost exclusively represent public organizations. *FTC* and *Federal.Trade.Commission* also represent the same entity. 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

1 Democratic_Sen._Harry_Reid	1 Democratic_Sen._George_McGovern
1 Democratic_Sen._Edmund_Muskie	1 Democratic_Sen._Carl_Levin
1 Democratic_Sen._Alan_Dixon	1 Democratic_Rep._William_Meyer
1 Democratic_Rep._William_Gray	1 Democratic_Rep._Peter_Kostmayer
1 Democratic_Rep._Mike_Lowry	1 Democratic_Rep._Mel_Levine
1 Democratic_Rep._John_Cavanaugh	1 Democratic_Rep._James_McClure
1 Democratic_Rep._George_Miller	1 Democratic_Rep._Gary_Ackerman
1 Democratic_Rep._Doug_Barnard	1 Democratic_Rep._David_Bonior
1 Democratic_Rep._Bob_Carr	1 Democratic_Mayor_Raymond_Flynn
1 Democratic_Leader_Jim_Wright	1 Democratic_Governor_Jim_Blanchard
1 Democratic_Gov._Robert_Kerrey	1 Democratic_Gov._Richard_Lamm
1 Democratic_Gov._Madeleine_Kunin	1 Democratic_Gov._Joseph_Brennan
1 Democratic_Gov._Jim_Blanchard	1 Democratic_Gov._Charles_Robb
1 Democratic_Gov._Bob_Kerrey	1 Democratic_Congressman_David_Nagle
1 Democratic_Chancellor_Helmut_Schmidt	1 Democratic_Chairman_Robert_Strauss
1 Democratic_Chairman_Richard_Wiener	1 Confederate_Gen._Sidney_Johnston
1 Vice_President_Warren_Lasko	1 Vice_President_Sylvia_Brenner
1 Vice_President_Diana_Aldridge	1 Vice_President_Carlos_Morales
1 Vice_President_Bill_Reidy	1 Vice_Chairmen_Randall_Tobias
1 Vice_Chairman_Martin_Shugrue	1 Vice_Chairman_Bill_Ryan
1 Vice_Adm._Dudley_Carlson	1 State_Sen._Barry_Keene
1 State_Rep._Bud_Gardner	1 Rear_Adm._Stuart_Platt
1 Prime_Minister_Paul_Schlueter	1 Prime_Minister_Karoly_Grosz
1 Prime_Minister_Fidel_Castro	1 Prime_Minister_Felipe_Gonzales
1 Managing_Editor_Henry_Muller	1 Maj._Gen._Schuyler_Bissell
1 Labor_Secretary_Ray_Donovan	1 Justice_Secretary_Hector_Rivera
1 Interior_Minister_Friedrich_Zimmermann	1 Interior_Minister_Friedrich_Zimmerman
1 Interior_Minister_Cesar_Gaviria	1 Interior_Minister_Abel_Salinas
1 Foreign_Minister_Alexander_Bessmertnykh	1 Finance_Minister_Ludwig_Erhard
1 Finance_Minister_Leslie_Navarro	1 Economics_Minister_Carlos_Solchaga
1 Deputy_Mayor_Luis_Salazar	1 Defense_Secretary_Paul_Thayer
1 Defense_Secretary_Frank_Gaffney	1 Commerce_Minister_Alejandro_Martinez
1 Associate_Director_Craig_Simmons	1 Assistant_Administrator_Rita_Lavelle

Table 5.2: Compound Class 171 (low frequency part)

1373 General_Motors_Corp.	1211 Drexel_Burnham_Lambert_Inc.
1153 Ford_Motor_Co.	1000 International_Business_Machines_Corp.
965 General_Electric_Co.	923 Shearson_Lehman_Brothers_Inc.
858 Chrysler_Corp.	811 First_Boston_Corp.
739 Merrill_Lynch_&_Co.	691 Morgan_Stanley_&_Co.
671 Shearson_Lehman_Hutton_Inc.	640 News_Corp.
610 American_Telephone_&_Telegraph_Co.	577 PaineWebber_Inc.
509 Prudential-Bache_Securities_Inc.	441 Texaco_Inc.
391 McDonnell_Douglas_Corp.	391 Dean_Witter_Reynolds_Inc.
342 Time_Inc.	336 AMR_Corp.
334 CBS_Inc.	332 American_Express_Co.
296 Campeau_Corp.	289 BankAmerica_Corp.
288 Du_Pont_Co.	268 Allegis_Corp.
254 General_Dynamics_Corp.	253 Digital_Equipment_Corp.
252 Kohlberg_Kravis_Roberts_&_Co.	248 Exxon_Corp.
1 Chase_Federal_Savings_&_Loan_Association	1 USAA_Federal_Savings_Bank
1 Liberty_Federal_Savings_Bank	1 Fayez_Sarofim_&_Co.
1 Dean_Witter_&_Co.	1 Bear_Stearns_&_Co.
1 Shearson_Lehman_Hutton_Ltd.	
1 Westdeutsche_Landesbank_Girozentrale_International_S.A.	
1 Merrill_Lynch_Capital_Markets_Ltd.	1 Merrill_Lynch_Capital_Markets_Group.
1 Luz_International_Ltd.	1 Bear_Stearns_Asset_Management_Inc.
1 Thorndike_Deland_Associates_Inc.	1 International_Services_Management_Ltd.
1 International_Management_Consultants_Inc.	1 International_Explorer_Services_Ltd.
1 International_Development_Systems_Inc.	1 Mezzanine_Fund_L.P.
1 Merrimack_Bancorp_Inc.	1 Finish_Line_Inc.
1 Eagle_Bancorp_Inc.	1 ELDERS_IXL_Ltd.
1 ECI_Telecom_Ltd.	1 Cunard_Line_Ltd.
1 Carena_Bancorp_Inc.	1 Britannia_Airways_Ltd.
1 American_Cyanamid_Inc.	1 Alpha_Service_S.A.
1 Aloha_Airlines_Inc.	1 Abington_Bancorp_Inc.

Table 5.3: Compound Class 179

2984 common_stock	2035 preferred_stock
1528 cash_flow	909 bank_debt
849 long-term_debt	733 foreign_debt
708 subordinated_debt	707 senior_debt
587 balance_sheet	207 short-term_debt
198 balance_sheets	194 cost_overruns
187 corporate_debt	185 debt_load
183 convertible_preferred_stock	152 international_debt
131 debt_outstanding	120 Class_B_stock
113 debt_ratings	110 cumulative_preferred_stock
106 corporate_IOUs	102 current_delivery
93 preference_stock	89 ozone_layer
85 buffer_stock	79 unsecured_debt
76 convertible_preferred	75 external_debt
74 debt_offering	73 current_contract
70 blood_clots	62 Class_B_common
61 cumulative_convertible_preferred_stock	58 corporate_governance
1 working-capital_debt	1 unregistered_debt
1 unfunded_debt	1 speculative-grade_debt
1 single-A-rated_debt	1 shorter-term_debt
1 serial_debenture	1 revolving_debt
1 non-interest-bearing_debt	1 lower-yielding_debt
1 interest-paying_debt	1 interest-bearing_debt
1 interest-bearing_debenture	1 high-yielding_debt
1 high-yield_debenture	1 high-rate_debt
1 government-backed_debt	1 general-obligation_debt
1 equity-linked_debt	1 double-A-rated_debt
1 dollar-denominated_IOUs	1 asset-backed_debt
1 FMS_debt	1 ECU-denominated_debt
1 senior_cumulative_convertible_preferred_stock	
1 redeemable_cumulative_junior_preferred_stock	
1 convertible_cumulative_exchangeable_preferred_stock	
1 redeemable_convertible_preferred_stock	
1 non-cumulative_convertible_preferred_stock	1 junior_convertible_preferred_stock
1 exchangeable_pay-in-kind_preferred_stock	1 exchangeable_cumulative_preferred_stock
1 convertible_subordinated_preferred_stock	1 convertible_senior_preferred_stock
1 convertible_exchangeable_preference_stock	1 convertible_cumulative_preferred_stock

Table 5.4: Compound Class 221

6475 Fed	6276 SEC
3415 Reagan_administration	2530 IRS
2351 Pentagon	2270 Justice_Department
2252 Navy	1736 Commerce_Department
1634 FDA	1433 Army
1384 FCC	1329 FDIC
1271 Federal_Reserve_Board	1262 State_Department
1125 Bundesbank	1062 EPA
1026 FAA	1001 IMF
992 Labor_Department	885 Agriculture_Department
835 FBI	747 NASD
687 Defense_Department	667 Federal_Home_Loan_Bank_Board
661 British_government	652 NRC
647 Finance_Ministry	640 Japanese_government
622 FTC	603 UAW
497 Kremlin	489 PRI
482 Transportation_Department	475 PLO
460 Federal_Trade_Commission	458 CFTC
1 South_Korean_central_bank	1 East_Coast_money-center_bank
1 West_German_embassy	1 West_German_Army
1 South_Korean_navy	1 South_Korean_army
1 South_African_embassy	1 Roman_Catholic_diocese
1 North_Korean_government	1 Senate_Steering_Committee
1 Senate_Permanent_Subcommittee	1 Senate_Judiciary_Subcommittee
1 Senate_Investigations_Subcommittee	1 Senate_Health_Committee
1 Senate_Campaign_Committee	1 Senate_Banking_Subcommittee
1 Senate_Banking_Committees	1 Senate_Appropriations_Committees
1 Senate_Agricultural_Committees	1 House_Trade_Subcommittee
1 House_Intelligence_Committees	1 House_Banking_Committees
1 House_Agricultural_Committee	1 Homeless_Task_Force
1 Flood_Control_Authority	1 Fair_Trade_Committee
1 Fair_Housing_Amendments	1 Bay_Transit_Authority
1 Senate_Finance_Committees	1 Senate_Commerce_Committees
1 House_Finance_Subcommittee	1 House_Agriculture_Subcommittee

Table 5.5: Compound Class 256

text, we randomly chose 3,000 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 is occurrence-based. Only one time was a compound irrelevant to company names extracted (a person name). For a controlled experiment which will be described shortly, all the incorrect compounds were corrected by hand and a standard file was created which contained all the correct 2,999 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 in compound classes, we can carry out a real time identification of the compound-classes without actually storing the compounds in the lexicon.

As a controlled experiment within the above experiment, we conducted word-based compound extraction and compared the result with the above result. Instead of calculating the mutual information of adjacent classes, the mutual information of adjacent words was calculated for all the bigrams in the text. Then using various MI threshold values, words were *chained* in a similar way as described in Section 5.2, and compounds were identified. We then determined how many of the occurrences of company names in the standard file were identified in the word-based compound extraction experiment. The MI threshold values were varied 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 80 %.

The main reason the class-based approach is superior to 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 the class chain “WC397_WC381” discussed in Section 5. Table 5.6

word-MI	bi-count	word-1	word-2	class-MI	cl-bi-count	class-1	class-2
24.2	1	Detlev	Rohwedder	5.7	52240	397	381
23.2	4	Nomi	Ghez	5.7	52240	397	381
22.9	2	Mikael	Salovaara	5.7	52240	397	381
21.0	4	Patricio	Aylwin	5.7	52240	397	381
19.7	8	Karlheinz	Kaske	5.7	52240	397	381
18.6	25	Hans-Dietrich	Genscher	5.7	52240	397	381
17.7	18	Clyde	Prestowitz	5.7	52240	397	381
15.4	2	Jules	Kroll	5.7	52240	397	381
12.6	3	Joel	Segal	5.7	52240	397	381
10.7	1	Elizabeth	McNamara	5.7	52240	397	381
9.2	6	Richard	Epstein	5.7	52240	397	381
7.0	2	Carl	Olson	5.7	52240	397	381
6.7	6	Michael	Cook	5.7	52240	397	381
5.9	1	Brian	Jacobs	5.7	52240	397	381
4.5	3	Michael	Moore	5.7	52240	397	381
3.8	1	Gary	Miller	5.7	52240	397	381
2.2	1	William	Walsh	5.7	52240	397	381
2.1	1	David	Morris	5.7	52240	397	381
1.9	1	Michael	Wright	5.7	52240	397	381
1.6	1	Thomas	Baker	5.7	52240	397	381
0.3	1	John	Jackson	5.7	52240	397	381

Table 5.6: Examples of Compounds for Names

shows some examples of compounds matching the pattern "WC397_WC381" 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 the same for all the compounds because we collected compounds with the same class chain. Although all 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 "Detlev" and "Rohwedder" were very rare names in WSJ news articles in 1987 and 1988, the MI becomes very high even though "Detlev Rohwedder" 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 52240. When we examine frequencies of all the compounds in the text that match "WC397_WC381", it turns out that more than 80 % of the compounds appear less than five times in the text. This shows how the data sparseness problem is critical for compound extraction and how the class-based approach can alleviate this problem.

Compound clustering has potential application in many areas. As mentioned in Introduction, one obvious candidate application is example-based machine translation in which similarity or substitutability among multiword compounds are more important than those among individual words. Compound clustering can also alleviate weakness of hard clustering; the problem of homonym, or more generally, sensitivity to the context. Although *bank* of "river bank" and that of "investment bank" are treated as the same entity in hard clustering of words, successful compound clustering methods should put these two compounds into totally different compound classes since distributional characteristics of the two compounds are expected to be quite different. Related to the problem of context sensitivity, compound clusters are also expected to provide information sources of new dimensionality to probabilistic parsing methods. In the comparative analysis of re-

cent successful probabilistic parsing methods, Charniak (1997) has shown that collecting statistics based on word classes of head words of constituents can improve the performance of the parser. Since distributions of a compound which contain a given head word provides more specific and reliable information on the context in which the compound occurs than the distributions of the head word itself, compound clusters are expected to be useful for probabilistic parser. Compound clusters could also be useful for broad-coverage terminology compilation and lexicography, information retrieval and language modeling for speech recognition.

Chapter 6

Conclusion

The research in this thesis addresses the problem of automatically constructing word clusters from plain texts and applying them to corpus-based natural language processing tasks to alleviate the data sparseness problem. The major contribution of this thesis work is the experimental demonstration of our claim that the improvement of clusters in terms of cross-entropy is reflected in the improvement of NLP tasks, especially for rare events. Various methods for improving and extending the conventional MI clustering algorithm were studied and experimentally evaluated. It was shown that word clusters created with these methods can improve performance of POS tagging tasks, especially for rare events such as tagging unknown words or tagging with small training data, and that clusters with higher AMI improve tagging accuracy more. We have also shown how word clusters can be used to construct compound clusters in an efficient and reliable way. It was demonstrated that for the purpose of compound extraction, class-based mutual information is much more reliable than word-based one.

Because rare events are very common in large scale NLP tasks, this thesis work should be interesting to researchers in many NLP fields including corpus linguistics, broad-coverage terminology compilation and lexicography, text categorization and summarization, machine translation, as well as information retrieval and language modeling for speech recognition.

This research can be extended to many different directions. The clustering criterion used in this work is based on bigram language modelling, and the model can be at least theoretically naturally extended to trigram language modelling. How to constrain the ex-

plosion of computation time is the practical problem, but a hybrid of both bigram-based and trigram-based components might be a realistic compromise. Another direction for future research is to directly integrate the cluster information to various NLP engines. NLP systems which use context information as features, such as decision tree-based systems or maximum entropy-based systems, can readily incorporate word class and compound class information. Compound class information should also be useful for example-based machine translation systems. It would also be interesting to consider how to utilize automatic clustering to improve the efficiency of terminology compilation. Because of the dramatic development of internet, it has now become a major resource for terminology compilation. Accordingly, automatic or semi-automatic way of compiling new words and new compounds are seriously desired and automatic clustering might play an important role in this problem.

References

- Bezdek, J.C. (1981) *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York.
- Black, E., Jelinek, F., Lafferty, J., Mercer, R. and Roukos, S. (1992) "Decision Tree Models Applied to the Labeling of Text with Parts-of-Speech." *Proceedings of the DARPA Workshop on Speech and Natural Language*, Morgan Kaufmann.
- Black, E., Jelinek, F., Lafferty, J., Magerman, D., Mercer, R. and Roukos, S. (1993) "Towards History-based Grammars: Using Richer Models for Probabilistic Parsing." *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*.
- 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*.
- Brill, E. (1992) "A simple rule-based part of speech tagger." *Proceedings of the DARPA Speech and Natural Language Workshop*, Morgan Kaufmann.
- Brill, E. and Marcus, M. (1992) "Automatically Acquiring Phrase Structure Using Distributional Analysis." *Darpa Workshop on Speech and Natural Language*, Harriman, N.Y.
- Brill, E. (1993) *A Corpus-Based Approach To Language Learning*. Doctoral dissertation, University of Pennsylvania.

- Briscoe, T. and Carroll, J. (1997) "Automatic extraction of subcategorization from corpora." *Proceedings of the 5th Conference on Applied Natural Language Processing*
- 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.
- Charniak, E. (1997) "Statistical parsing with a context-free grammar and word statistics." *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, AAAI Press/MIT Press, Menlo Park.
- Charniak, E., Carroll, G., Adcock, J., Cassandra, A., Gotoh, Y., Katz, J., Littman, M., and McCann, J. (1994) "Taggers for Parsers." Technical Report CS-94-06, Department of Computer Science, Brown University.
- Church, K.W. (1988) "A stochastic parts program and noun phrase parser for unrestricted text." *Proceeding of the Second Conference on Applied Natural Language Processing*.
- Church, K. and Hanks, P. (1990) "Word Association Norms, Mutual Information, And Lexicography." *Computational Linguistics*, Vol. 16, No 1, pp. 22-29.
- Collins, M. (1996) "A New Statistical Parser Based on Bigram Lexical Dependencies." *Proceedings of the 34th Annual Meeting of the ACL*, pp. 184-191.
- Cutting, D., Kupiec, J., Pedersen, J., and Sibun, P. (1992) "A Practical Part-Of-Speech Tagger." *Proceedings of the Third Applied Natural Language Processing Conference*.
- deMarcken, C.G. (1990) "Parsing the LOB corpus." *Proceedings of the 28th Annual Meeting of the ACL*, pp. 243-251.
- Duda, R.O. and Hart, P.E. (1973) *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York.
- Jardino, M. and Adda, G. (1991) "Automatic word classification using simulated annealing", *ICASSP 91*.

- Jelinek, F., Lafferty, J.D., and Mercer, R.L. (1990) *Basic Method of Probabilistic Context Free Grammars*. Technical Report RC 16374 (72684), IBM, Yorktown Heights, NY 10598.
- Kneser, R. and Ney, H. (1993) "Improved Clustering Techniques for Class-Based Statistical Language Modelling." *Proceedings of European Conference on Speech Communication and Technology*.
- Lafferty, J. and Mercer, R. (1993) "Automatic Word Classification Using Features of Spellings." *Proceedings of the 9th Annual Conference of the University of Waterloo Center for the New OED and Text Research*, Oxford Univ. Press.
- Lari, K. and Young, S.J. (1990) "The estimation of stochastic context-free grammars using the Inside-Outside algorithm." *Computer Speech and Language*, 4, pp. 35–56.
- Lee, L. (1997) *Similarity-Based Approaches to Natural Language Processing*. Doctoral dissertation, Harvard University.
- Lee, L. and Pereira, F. (1999) "Distributional Similarity Models: Clustering vs. Nearest Neighbors." *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.
- Li, H. and Abe, N. (1996) "Clustering Words with the MDL Principle." *Proceedings of the 16th International Conference on Computational Linguistics*.
- MacNaughton-Smith, P., Williams, W.T., Dale, M.B. and Mockett, L.G. (1964) "Dissimilarity Analysis." *Nature*, Vol. 202, pp. 1034–1035.
- MacQueen J. (1967) "Some methods for classification and analysis of multivariate observations." *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol 1, Statistics, pp. 281–297. Berkeley: University of California Press.
- Manning, C. (1993) "Automatic acquisition of a large subcategorization dictionary from corpora." *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*.

- Magerman, D. (1994) *Natural Language Parsing as Statistical Pattern Recognition*. Doctoral dissertation, Stanford University.
- Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1994) "Building a large annotated corpus of English: the Penn Treebank." *Computational Linguistics*, Vol. 19, No 2, pp. 313–330.
- Martin, M., Liermann, J. and Ney, H. (1995) "Algorithms for Bigram and Trigram Word Clustering." *Proceedings of European Conference on Speech Communication and Technology*.
- McMahon, J. and Smith, F. (1996) "Improving Language Model Performance with Automatically Generated Word Hierarchies." *Computational Linguistics*, Vol. 22, No 2, pp. 217–247.
- Mikheev, A. (1997) "Automatic Rule Induction for Unknown-Word Guessing." *Computational Linguistics*, Vol. 23, No 3, pp. 405–423.
- Pedersen, T. and Bruce, R. (1997) "Distinguishing Word Senses in Untagged Text." *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, Providence, RI.
- Pereira, F. and Schabes, Y. (1992) "Inside-Outside Reestimation From Partially Bracketted Corpora." *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*.
- Pereira, F., Tishby, N., and Lee, L. (1993) "Distributional Clustering of English Words." *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*.
- Quinlan, J. (1993) *C4.5: Programs for Machine Learning*. Morgan Kaufmann.
- Ratnaparkhi, A. and Roukos, A. (1994) "Maximum Entropy Model for Prepositional Phrase Attachment." *Proceedings of the ARPA Workshop on Human Language Technology*.

- Ratnaparkhi, A. (1998) *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Doctoral dissertation, University of Pennsylvania.
- Rooth, M., Riezler, S., Prescher, D., Carroll, G. and Beil, F. (1999) "Inducing a Semantically Annotated Lexicon via EM-Based Clustering." *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.
- Rosenfeld, R. (1994) *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. Doctoral dissertation, Carnegie Mellon University.
- Saul, L. and Pereira, F. (1997) "Aggregate and mixed-order Markov models for statistical language processing." *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, Providence, RI.
- Schutze, H. (1992) "Dimensions of Meaning." *Proceedings of Supercomputing '92*.
- Smadja, D. (1993) "Retrieving Collocations From Text: Xtract." *Computational Linguistics*, Vol. 19, No 1, pp. 143-177.
- Sneath, P. (1957) "The application of computers to taxonomy." *Journal of General Microbiology*, Vol. 17, pp. 201-226.
- Sokal, R.R. and Michener, C.D. (1958) "A statistical method for evaluating systematic relationships." *University of Kansas Scientific Bulletin*, Vol. 38, pp. 1409-1438.
- 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., Waibel, A., Gibson, T. and Evans, D. (1993) "Frequency Estimation of Verb Subcategorization Frames Based on Syntactic and Multidimensional Statistical Analysis." *Proceedings of the International Workshop on Parsing Technology '93*, pp. 309-317.

- Ushioda, A. (1996a) "Hierarchical Clustering of Words." *Proceedings of The 16th International Conference on Computational Linguistics*, pp. 1159–1162.
- Ushioda, A. (1996b) "Hierarchical Clustering of Words and Application to NLP Tasks." *Proceedings of the 4th Workshop on Very Large Corpora*, pp. 28–41.
- Yarowsky, D. (1992) "CONC: Tools for text corpora." Technical Memorandum 11222-921222-29, AT&T Bell Laboratories.
- Yarowsky, D. (1995) "Unsupervised Word Sense Disambiguation Rivaling Supervised Methods." *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pp. 189–196.
- Weischedel, R., Meteer, M., Schwartz, R., Ramshaw, L., Palmucci, J. (1993) "Coping with Ambiguity and Unknown Words through Probabilistic Models." *Computational Linguistics*, Vol. 19, No 2, pp. 359–382.
- Ward, J. (1963) "Hierarchical grouping to optimize an objective function." *Journal of the American Statistical Association*, Vol. 58, pp. 236–244.
- Wilks, S. (1962) *Mathematical Statistics*. John Wiley and Sons, New York.