Computational Linguistics and Chinese Language Processing
Vol. 1, No. 1, pp. 100-156, 1996.

# An Overview of Corpus-Based Statistics-Oriented (CBSO) Techniques for Natural Language Processing

**Keh-Yih Su*, Tung-Hui Chiang** and Jing-Shin Chang***

*Department of Electrical Engineering
National Tsing-Hua University
Hsinchu, Taiwan 300, R.O.C.

**Advanced Technology Center,
Computer & communication Research Laboratories,
Industrial Technology Research Institute
Chutung, Hsinchu, Taiwan 310, R.O.C.

email: kysu@bdc.com.tw*, thchiang@e0sun3.ccl.itri.org.tw**, shin@hermes.ee.nthu.edu.tw*

**keywords:** corpus, CBSO, knowledge acquisition, class-based language modeling, natural language processing

## ABSTRACT

A Corpus-Based Statistics-Oriented (CBSO) methodology, which is an attempt to avoid the drawbacks of traditional rule-based approaches and purely statistical approaches, is introduced in this paper. Rule-based approaches, with rules induced by human experts, had been the dominant paradigm in the natural language processing community. Such approaches, however, suffer from serious difficulties in knowledge acquisition in terms of cost and consistency. Therefore, it is very difficult for such systems to be scaled-up. Statistical methods, with the capability of automatically acquiring knowledge from corpora, are becoming more and more popular, in part, to amend the shortcomings of rule-based approaches. However, most simple statistical models, which adopt almost nothing from existing linguistic knowledge, often result in a large parameter space and, thus, require an unaffordably large training corpus for even well-justified linguistic phenomena.

The corpus-based statistics-oriented (CBSO) approach is a compromise between the two extremes of the spectrum for knowledge acquisition. CBSO approach emphasizes use of well-justified linguistic knowledge in developing the underlying language model and application of statistical optimization techniques on top of high level constructs, such as annotated syntax trees, rather than on surface strings, so that only a training corpus of reasonable size is needed for training and long distance dependency between constituents could be handled.

In this paper, corpus-based statistics-oriented techniques are reviewed. General techniques applicable to CBSO approaches are introduced. In particular, we shall address the following important issues so that general guidelines for developing a particular NLP system will be available to NLP researchers:

(1) general tasks in developing an NLP system;

(2) why CBSO is the preferred choice among different strategies;

(3) how to achieve good performance systematically using a CBSO approach, and

(4) frequently used CBSO techniques.

Several examples characterizing the CBSO approach are also reviewed in this paper.

## 1. Introduction

In general, the development of a natural language processing (NLP) system must handle the following problems.

(1) Knowledge representation: how to organize and describe linguistic knowledge in a linguistically meaningful and computationally feasible manner. For example, one may describe linguistic knowledge in terms of grammar rules and restriction rules on grammatical constructs. On the other hand, the same knowledge could be described using a set of features and the associated probabilities in statistical approaches.

(2) Knowledge control: how to apply linguistic knowledge for effective processing. For instance, a context-free grammar may be carried out by an LR parser. The disambiguation process may be carried out by an expert system by consulting a set of disambiguation rules. Or a statistics-oriented system may adopt a statistical language model by using likelihood measures for choosing the most likely analysis among all.

(3) Knowledge integration: how to use the various knowledge sources effectively. A system may resolve ambiguity by using both syntactic and semantic constraints. It may also adopt a rule-based system for parsing; however, probability is used for choosing preferences instead of the rule-based system. A system adopting different paradigms (e.g., rule-based and statistical approaches) at the same time is called a "hybrid system" by some researchers [Su 92a].

(4) Knowledge acquisition: how to systematically and cost-effectively acquire the required knowledge and maintain consistency of the knowledge base, so that there is no confliction among the rules which may degrade system performance. Some systems may rely on human experts to induce linguistic rules based on linguistics theory or the materials they have observed. Statistical approaches, on the other hand, would automatically acquire the knowledge, which is the probability values in this case, *via* estimation processes, from a corpus.

In general, real text contains much greater numbers of ambiguities and illformed sentences than people realize at the first glance. For instance, the sentence "The farmer’s wife sold the cow because she needed money" usually seems to people to have no ambiguity. It normally refers to, unambiguously, the fact that "she" is "the farmer’s wife" [King 95]. Similar sentences, such as "The farmer’s wife sold the cow because she wasn’t giving enough milk," however would be interpreted in quite different ways by people without any difficulty. However, both sentences present an ambiguity problem to an NLP system. The knowledge required for resolving such ambiguity and ill-formedness is usually non-deterministic, huge in amount, messy and fine-grained in nature although most people are not aware of these facts. Therefore, it is costly and time-consuming to acquire such knowledge by hand. As a result, usually, knowledge acquisition is the major engineering bottleneck for designing a large NLP system.

Due to its important role in NLP, various knowledge acquisition strategies had been exploited in the literature. Roughly, the methodologies can be characterized into the following major categories, namely (1) rule-based approaches [Hutchins 86], (2) purely statistical approaches [Brown 90], (3) corpus-based statistics-oriented (CBSO) approaches [Chen 91, Su 92a], (4) symbolic learning approaches developed in the AI field [Michalski 83, 86], and (5) connectionist approaches (i.e., neural network approaches [Cottrell 89, Schalkoff 92]). Each category may have its own specific method for knowledge representation. Since the first three categories are frequently used in the NLP community, and the last two approaches have not shown much success in real NLP applications so far, this paper will discuss and compare the first three methodologies with particular emphasis on the CBSO approaches.

A rule-based approach usually has the following characteristics. (1) The linguistic knowledge is usually expressed in terms of high level constructs such as parts of speech, phrases, syntactic trees and feature structures, which are described in most traditional linguistics textbooks, and the knowledge is expressed in the form of syntactic or semantic constraints over such constructs. (2) Most rule-based systems have a strict sense of well-formedness; therefore, the rules are applied deterministically to reject ill-formed constructs. (3) Most rules are based on existing linguistic theories that are linguistically interesting. When the required knowledge does not appear in the literature, ad hoc heuristic rules may be used. (4) Such rules are normally induced by linguists based on their expertise. For instance, heuristics such as "a determiner cannot be followed by a verb" may be used in filtering out inappropriate part of speech tags in a rule based POS tagging system.

In contrast to a rule based system, a purely statistical approach has the following characteristics. (1) Its knowledge is expressed in terms of the likelihood of certain events. Most of the time, the language generation process is simply modeled as a simple stochastic decoding process, such as a Markov chain [Brown 90]. Each event is associated with the occurrence of a particular word string, such as a word N-gram, not a high level construct. There is, essentially, no syntactic or semantic constraints on the words. Normally, the only constraint on the words is the dependency implied by the conditional probabilities among adjacent words. (2) There is not a strict sense of well-formedness. Therefore, all possible analyses will be assigned probabilities for searching the most probable analysis. Hence, a large computation time is usually required. (3) The probabilities are usually estimated automatically to maximize the likelihood of generating the observations in a training corpus. Therefore, the knowledge acquisition task is simply an estimation process.

For instance, a purely statistical approach for translating an English sentence into a French sentence might be interested in asking which French words are most likely to be the translation of a particular English word, and what the most likely word order of the translated words is [Brown 90]. The most likely translation is then determined based on such probabilities for all possible target words and word orders.

The major advantage of a rule-based system is that existing linguistic knowledge can be incorporated into the system directly in a compact and comprehensive way. However, it is hard to scale up such a system for several reasons. First, the required knowledge is huge and messy; the cost for acquiring and maintaining the huge set of rules is extremely high. Second, it is not easy for such a system to resolve problems of ambiguity and ill-formedness, which usually need nondeterministic knowledge and require objective preference metrics to quantitatively qualify all possible analyses. Third, the portability of such a system is poor when porting from one domain to another or from one language to a different language. Finally, although new included rules may improve the system performance for certain cases, they may not improve, and may very likely even degrade, performance in other cases. In other words, a local improvement made by modifying the rules for bad cases does not guarantee a global improvement for all cases; the result, then, is a 'seesaw phenomena,' in which a few mistakes are corrected at the cost of producing even more new mistakes. Such phenomena have been observed in many cases in the history of NLP development. The induced effects for other unseen cases as a result of adding new rules, are, therefore, usually unpredictable. As a result, enlarging the rule base of the system does not guarantee that a monotonic increase in system performance will result. Besides, from a practical point of view, it is very difficult to maintain the rules, especially when the system must be maintained by many persons across a long time span. Therefore, scaling-up such a system often degrades the system performance and renders it cost-ineffective because it is very difficult to further improve the system performance in later stages.

Purely statistical models are preferred with respect to a rule-based approach in several respects. (1) Non-deterministic behavior and uncertainty can be objectively qualified by objective probabilistic metrics. (2) The system parameters can be consistently maintained easily even when the system is scaled up because the knowledge acquisition task, namely the estimation process, can easily be repeated over enlarged training materials when new instances of interest are included. (3) Automatic or semi-automatic training of the system parameters is possible using well developed optimization techniques. Hence, the burden of knowledge acquisition can be shifted from human experts to machines. (4) The methodologies can be easily adapted to other languages and domains since the estimation process is highly independent of any particular language and application domain. Furthermore, since no high level constructs are used in a purely statistical model, no preprocessing, such as syntax analysis, is required to produce such constructs; therefore, the optimization process of a purely statistical model can be implemented easily.

However, the parameter space of a purely statistical model is usually extremely large, and searching for the optimal solution is time consuming in many cases because the language model is not established on top of high level constructs. For instance, a practical NLP system should be able to handle $10^5$ words for real applications. Using a purely statistical approach whose language model is a simple word trigram model would requires $10^{15}\left(=10^5 \times 10^5 \times 10^5\right)$ parameters to describe the complete probabilistic knowledge. According to the general rule, the number of samples required to reliably estimate a set of parameter values is about 5 to 10 times the parameter size. Therefore, a text corpus having about $10^{16}$ words is, theoretically, required to estimate the parameter values reliably in this case. The estimated parameter values might not be reliable if a large corpus is not available. (Although, practically many tri-gram combinations never occur, and a smaller corpus should be sufficient. However, the magnitude of the number of parameters is still huge.) For this reason, a system which attempted to extend the scope of knowledge by including more words in its n-gram model would be unaffordable in most applications. Ambiguity resolution, which must take long distance dependency into consideration, sometimes also would be infeasible using such purely statistical approaches. Therefore, a compromise must be made to take advantage of both rule-based approaches and purely statistical approaches. To avoid the drawbacks of both rule-based and purely statistical approaches, CBSO approaches were proposed in [Chen 91, Su 92a]. Basically, these approaches impose stochastic models on classes, not just on words as some n-gram models do, as will be described in the following sections.

In fact, words can be clustered into different equivalent classes of certain particular types for processing purposes, and many such equivalent classes are well identified in linguistics. For instance, words having the same parts of speech exhibit similar syntactic behavior. Therefore, we can use classes, instead of words, in developing statistical language models. A statistical language model based on part of speech trigram, for example, requires only about $10^6$ parameters to describe the complete statistical knowledge if we are using a tag set of 100 parts of speech. The size of the training corpus would be about $10^7$ in this case, which is affordable in the current environment.

A statistical language model can even be developed based on syntactic categories, i.e., nonterminal nodes of syntax trees or can even be based on semantic categories when higher level analysis is necessary for applications. By developing statistical language models based on high level constructs at the syntactic or semantic level, most of the above-mentioned disadvantages of a purely statistical method can be relieved. For instance, the size of the training corpus can be greatly reduced. The linguistic knowledge need not be learned from low level constructs. Long distance dependency can also be modeled since we are able to impose constraints among adjacent nonterminal nodes (such as NP and VP), whose head words may be far apart in the surface strings. All such advantages suggest use of the CBSO approaches proposed in the following sections.

## 2. What are CBSO Approaches and Why CBSO Approaches

The CBSO approaches are hybrid approaches which take advantage of both rule-based and purely statistical approaches. A CBSO approach builds statistical language models on top of high level constructs such as annotated syntax trees. For example, a CBSO model for machine translation was proposed in [Su 90, 92a, 95] based on high level constructs such as parse trees and normal forms; the translation problem is modeled as an optimization problem which selects the best translation that maximizes the following translation score:

$$\begin{aligned} & P\left(T_i|S_i\right) \\ & \cong \sum_{I_i}\left\{\left[P\left(T_i|PT_t(i)\right)\times P\left(PT_t(i)|NF1_t(i)\right)\times P\left(NF1_t(i)|NF2_t(i)\right)\right] \right. && \ldots(1) \\ & \quad \times \left[P\left(NF2_t(i)|NF2_s(i)\right)\right] && \ldots(2) \\ & \quad \left. \times \left[P\left(NF2_s(i)|NF1_s(i)\right)\times P\left(NF1_s(i)|PT_s(i)\right)\times P\left(PT_s(i)|S_i\right)\right]\right\}, && \ldots(3) \end{aligned}$$

where $\left(S_i,\ T_i\right)$ is the i-th source-target translation pair, $\left(PT_s,\ PT_t\right)$ are the parse trees for the source-target sentences, $NF1_s$ and $NF2_s$ represent syntactically and semantically normalized parse trees, called normal forms, of the source sentence, $NF1_t$, $NF2_t$ are the normal forms of the target sentence, and the summation of the probabilities is taken over all such intermediate representations, $I_i$. The three equations (1), (2) and (3) by themselves define the generation, transfer and analysis models of a transfer-based MT system in a CBSO manner; they can be further simplified for implementation. (Some of the details will be given in later sections.)

Such an approach usually has the following characteristics. (1) It uses high level constructs long adopted in conventional linguistics, instead of surface strings, to model the stochastic behavior of the languages, so that the number of parameters in the language model can be greatly reduced. (2) It uses a parameterized statistical approach to resolve ambiguities and ill-formedness, so that the language processing task can be objectively optimized and the required knowledge; i.e., the parameter values, can be acquired automatically and consistently. (3) It is usually more robust compared with the purely statistical approaches since statistical optimization is applied on high level constructs, whose statistical properties are more likely to be generalized to unseen data better than surface strings.

For instance, in a CBSO approach for parts of speech tagging, we are interested in knowing the likelihood of a part of speech following a determiner and an adjective, and we use such knowledge to justify the most likely part of speech for a word which appears immediately following a word from the determiner category and the adjective category. Unlike a purely statistical model, which needs to estimate all the $10^{15}$ probabilities of occurrences of word pairs in a system having $10^5$ words in its vocabulary, a CBSO taggeronly requires $10^6$ parameters, which correspond to all possible occurrences of part of speech pairs, to model the system if the words are classified into 100 parts of speech.

In a more complicated system, such as a machine translation system, the CBSO approach will be even more demanding considering the huge amount of analysis, transfer and generation knowledge required to optimize the translation score, $P\left(T_i|S_i\right)$, for all possible source-target sentence pairs $\left(S_i,T_i\right)$. In a purely statistical model, the huge number of possible alignments will make this impractical for a large system. On the other hand, by introducing intermediate linguistics constructs, such as nodes in parse trees and normal forms, which represent particular syntactic or semantic equivalent classes, as shown in the above CBSO translation model [Su 95], the translation task can be greatly simplified by using a parameter space of affordable size. Using such a formulation, statistical optimization techniques can be applied to get the best translation objectively without resorting to rules or a large parameter space. Such a formulation thus combines the advantages of both the rule-based systems and purely statistical methods.

In summary, in comparison with rule-based approaches, the CBSO approaches can handle non-deterministic situations more objectively by adopting probabilistic measures estimated from the corpus. Knowledge acquisition for such a system is also less expensive and much faster than that for a rule-based system since the acquisition process is simply a parameter estimation process. The knowledge base, namely the probability values, can also be maintained more consistently as every parameter is estimated by jointly considering all the data in the corpus, which is a big plus compared with a rule-based system, especially when its knowledge is maintained by different persons across a long time span.

In comparison with purely statistical approaches, the CBSO approaches make use of well justified linguistics constraints and constructs in structural and semantic levels. Therefore, an unaffordably large corpus is not required in a CBSO approach to develop the underlying language model. Under such circumstances, the parameter values can usually be estimated more reliably than can those for a purely statistical model. Furthermore, models based on high level constructs have greater generalization capability for unobserved text. Therefore, a CBSO approach is usually more robust than are other approaches.

Furthermore, since the language models for CBSO approaches are based on high level constructs, the dependency of such constructs, instead of the dependency among surface strings, can be easily modeled for statistical optimization. Therefore, long distance dependency, which can not be handled in a purely statistical mode, can be handled easily in CBSO approaches. Finally, the searching space for optimal solution finding is significantly reduced in CBSO approaches. Therefore, CBSO approaches are preferred over purely statistical approaches.

To make a CBSO approach feasible, a corpus of moderate size, cheap computation power and well developed statistical techniques for reliable estimation of the parameter values are essential. Fortunately, large online electronic texts and dictionaries are becoming more and more easily accessible due to the rapid growth of the number of companies adopting online processing and publication in recent years. Microelectronic technology has also seen computation power, including processor speed and memory size, increase with time exponentially at very low cost. Furthermore, statistical techniques for parameter re-estimation, smoothing, backing-off, and robust estimation, which will be described in later sections, have been well developed and widely used in the statistics and speech communities in the last twenty years and further make CBSO approaches most appropriate for developing large scale NLP applications. Finally, the knowledge required to attack problems of ambiguity and ill-formedness, which are the two main problems in natural language processing, is largely inductive, not deductive. Statistical methods are especially appropriate for such problems. Therefore, we believed that CBSO approaches will be the most promising design methodologies for handling NLP tasks in the future.

## 3. Techniques Frequently Used in CBSO Approaches

In a typical CBSO NLP system, several important issues, including *feature selection*, *language modeling*, *corpus annotation*, *parameter estimation*, *smoothing* and *learning*, must be taken into consideration.

For instance, in a part-of-speech tagging task, we may use the current word ($w_i$) and the assigned tags of two preceding words ($c_{i-1}$, $c_i$) as the feature vector to decide the tag of this word. We then use the language model:

$$\hat{c}_i = \underset{c_i}{\text{argmax}}\ P\left(c_i | c_{i-1}, c_{i-2}\right) \times P\left(w_i | c_i\right)$$

to select the tag for $w_i$, where $P\left(c_i | c_{i-1}, c_{i-2}\right)$ is the probability that the current tag is $c_i$ given that the preceding tags are $\left(c_{i-1}, c_{i-2}\right)$, and $P\left(w_i | c_i\right)$ is the probability that the current word will be $w_i$ given that the current tag is $c_i$. The **argmax** operator returns the argument $c_i$ which makes the product of the above two probabilities maximal among all possible $c_i$. The probabilities, of the form $P\left(c_i | c_{i-1}, c_{i-2}\right)$ and $P\left(w_i | c_i\right)$, referred to as the *parameters* of the model, are all we need to make decisions in the part-of-speech tagging task. To estimate the values of the parameters, we might first annotate the corpus with correct parts of speech for the words in the corpus and then use MLE to estimate the parameters. Since the training corpus may not be large enough, many parameter values may be assigned zero probability according to the maximum likelihood criterion. Such assignments, in general, will result in poor performance for unseen input data. Therefore, smoothing techniques, which assign non-zero probabilities to unseen events, are important when the parameter space is large and the training data are sparse. Further details will be given in the following sections.

### 3.1 Designing Issues of an NLP System

In general, two kinds of features, namely statistical features and linguistic features (such as parts of speech and word senses) have been commonly used in various research works. Statistical features, such as mutual information and entropy, usually carry only statistical senses and carry few traditional linguistic notions. A few such features will be introduced in the following sections. Linguistic features, such as parts of speech, on the other hand, are usually used to designate certain properties of the linguistic constructs under consideration.

Good (statistical or linguistic) features should be able to provide discriminative information for the task. However, discriminative features are usually not easy to determine. Therefore, techniques for selecting the most discriminative features will be introduced in the following sections.

Given a set of features, the NLP system must make proper decisions based on a *language model*. The main purpose of language modeling is to choose a desired result from different alternatives for various kinds of linguistic problems, such as assigning the best POS to a word or assigning the best syntactic structure to a sentence. Therefore, the *language modeling* problem can usually be considered as a *classifier design* task, in which a set of features are given and decision rules are established to optimize certain performance criteria, such as minimum recognition error or minimum cost. We therefore introduce two commonly used classifiers,the maximum likelihood classifier and Bayesian classifier.

Note that some of the feature selection mechanisms are designed to select a set of features that maximizes the system performance. Therefore, feature selection and classifier design might be integrated as a single step. Examples, such as CART (Classification and Regression Tree), which integrate the two tasks will be introduced as well.

Third, before applying a language model to resolve problems, the values of system parameters must be specified in advance through some *parameter estimation* process; such values of the parameters represent our knowledge of the language; therefore, the knowledge acquisition task in a CBSO approach can be regarded as an estimation process.

In general, the values of the parameters are estimated from an annotated corpus (called the training set) to meet some estimation criteria. This kind of training is usually called *supervised training* if the information for estimating the parameter values, such as the parts of speech of the words, is annotated by linguists in advance. However, corpus annotation is usually labor-intensive (and, hence, expensive). Another way to obtain the parameter values without making use of labeled corpora is called *unsupervised training*. Two typical unsupervised training methods, the EM algorithm and the Viterbi training algorithm, will be discussed in this paper.

The estimation process may introduce estimation errors due to insufficient training data. Although estimation errors can be reduced by increasing the size of the training corpus, the cost is generally too high to prepare a large annotated corpus. Therefore, several *parameter smoothing* procedures, capable of reducing the estimation error by smoothing unreliably estimated parameter values, are introduced.

Furthermore, the parameters, in general, are estimated using a maximum likelihood estimator (MLE) which assigns values to the parameters in such a way as to maximize the likelihood of the training corpus. The estimated values which maximize the likelihood of the training sentences, however, do not necessarily minimize the error rate (or other performance criteria) of the system when such values are used to resolve the NLP task. Therefore, such values may not provide the best *discrimination power* as far as the system performance for the *training set* is concerned. To compensate for the modeling errors and estimation errors, the adaptive learning process is usually required to adjust the parameter values to minimize the training set error rate.

Even though such values are properly adjusted so that the training set error rate performance is minimized, they may not be the best parameter values when used to resolve problems of ambiguity and ill-formedness in NLP tasks for *testing set* sentences that are never seen in training sentences. In most cases, the performance using such values, which optimizes the training set performance, is over-optimistic; the performance for unseen testing data, in general, will be degraded using such parameter values. To enhance the robustness of the system, robust estimators must be used so that the estimated values will still provide sufficient discrimination power when used on unseen testing data. For the above reasons, *robust adaptive learning* techniques capable of reducing the modeling error and statistical error as well as enhancing the discrimination capability and robustness will be reviewed. All the issues, including feature selection, language modeling, parameter estimation, parameter smoothing and robust oriented adaptive learning, will be addressed in the following sections.

## 3.2 Language Modeling as a Classifier Design Process

Given a set of features, the central problem is to design some decision rules which can maximize system performance. Most natural language tasks, especially for ambiguity resolution, can be formulated as pattern classification problems in which the classes are defined from the linguistic point of view. The compound noun detection model proposed in [Su 94a] and the unknown word detection model in [Lin 93] are two such applications. In fact, POS tagging, probabilistic parsing, word sense disambiguation, and many other interesting problems, can also be models as classification problems in which the classes are the set of possible parts of speech, parse trees or word senses. Therefore, we can regard language modeling as a classifier design process. In general, the choice of the classifier depends on the criterion to be optimized. Two widely used classifiers, the Bayesian classifier and maximum likelihood selection, are introduced here.

### Bayesian Classifier

If the cost (or penalty) associated with each type of misclassification is known and the *a posteriori probability*, $P(c_j \mid x)$, (i.e., the probability that the input feature vector *x* belongs to class $c_j$) is known, then it is possible to design a classifier which minimizes the risk (such as the error rate) of making a classification decision. This could be done by assigning $x$ to the class $c_i$ which has the minimal risk, $R_i$ , defined as follows for classifying an input token into class $c_i$ :

$$R_i = \sum_{j=1}^{K} l(c_i | c_j) P(c_j | x).$$

In the above equation, $l(c_i | c_j)$ is the loss one may incure if a token in $c_j$ is misclassified as $c_i$ , and $P(c_j | x)$ is the chance that the real class of $x$ is $c_j$ . Briefly, if one assigns $x$ to class $c_i$ , then the possible cost, $R_i$ , one may incure for doing such judgement is the the weighting sum of the loss $l(c_i | c_j)$ of all the different kinds of possible misclassifications, where the probability of 'possible misclassifications' is $P(c_j | x)$. To minimize the cost of making a wrong decision, the best classifier, in terms of a given cost function, will therefore assign $x$ to class $c_j$ if

$$j = \underset{\mathrm{i}}{\operatorname{argmin}} \; R_i,$$

where the **argmin** operator returns the argument (i) which has the minimum risk. Such a classifier is called the Bayesian classifier for the classification problem.

The simplest version of the Bayesian classifier is the *minimum error rate* classifier, which has the following special *zero-one* loss (or cost) function:

$$l(c_i | x \in c_j) = \begin{cases} 0, & c_i = c_j \\ 1, & c_i \neq c_j \end{cases} \quad i,j = 1, \ldots, K.$$

This means that the loss function assigns no penalty to a correct classification and assigns a unity loss to any type of error. Such a classifier will insure minimum probability of classification errors because the risk associated with class $c_i$ is now simplified as

$$\begin{aligned} R_i &= \sum_{j=1}^{K} l(c_i | c_j) P(c_j | x) \\ &= \sum_{j \neq i} P(c_j | x) \\ &= 1 - P(c_i | x). \end{aligned}$$

Minimizing the risk thus reduces to finding the class $c_j$ which has minimum error probability $1 - P(c_i | x)$ or maximum *a posteriori* probability $P(c_i | x)$. The minimum error rate classifier is normally used when the penalty of all types of errors are equally weighted.

In a part-of-speech (POS) tagging task, the minimum error rate classifier can be derived by regarding the input words as the input vector, and the classes are all possible POS sequences associated with the input words. We can, therefore, expect the minimum probability of tagging errors if we choose the POS sequence which has the maximum *a posteriori* probability, that is,

$$\hat{c}_i = \underset{c_j}{\operatorname{argmax}} P\left(c_j \mid x\right).$$

### Maximum Likelihood Selection

If all the classes are uniformly distributed, i.e., $P\left(c_j\right) = K^{-1}$ for $j$ = 1, ..., $K$, where K is the number of classes, then the class with maximal *a posteriori* probability $P\left(c_j \mid x\right)$ is also the one having maximal likelihood $P\left(x \mid c_j\right)$ since

$$\underset{c_j}{\operatorname{argmax}} P\left(c_j \mid x\right) = \underset{c_j}{\operatorname{argmax}} \frac{P\left(x \mid c_j\right) P\left(c_j\right)}{P(x)} = \underset{c_j}{\operatorname{argmax}} P\left(x \mid c_j\right).$$

The probabilities $P\left(c_j\right)$ and $P(x)$ do not affect the selection of the class because they are constants for different classes in this case.

With the ML selection scheme, the decision is made to select class $c_i$ if the conditional likelihood value of the input feature vector is maximum, i.e.,

$$\hat{c}_i = \underset{c_j}{\operatorname{argmax}} P\left(x \mid c_j\right).$$

The maximum likelihood decision rule is used when the class probabilities are not available.

## 3.3 Statistical Features

The decisions made by the classifiers are based on a set of features. The features can be statistical measures, which can be acquired easily from a training corpus. Some features, on the other hand, are linguistic features such as parts of speech. In the following sections, some frequently used statistical features are introduced. Automatic methods for generating linguistic features will be introduced in later sections.

### · Likelihood and Likelihood Ratio

*Likelihood* is a measure of how likely it is that an event will happen under a specific condition. Formally, the conditional probability of $P\left(\boldsymbol{x} \mid \omega_i\right)$ is called the *likelihood* of an event $\boldsymbol{x}$ being generated from the *i*-th class (or model) $\omega_i$. The likelihood values of a feature vector are normally used in the maximum likelihood classifiers for classification; the feature vector $\boldsymbol{x}$ is categorized into class $\omega_m$ if

$$P\left(\boldsymbol{x} \mid \omega_m\right) > P\left(\boldsymbol{x} \mid \omega_i\right) \text{ for all } \omega_i \neq \omega_m .$$

Another measure, known as the *likelihood ratio* of the feature vector $\boldsymbol{x}$ in the two classes $\omega_1$ and

$\omega_2$, is defined as follows [Duda 73]:

$$\gamma = \frac{P(x|\omega_1)}{P(x|\omega_2)}.$$

It is frequently used to determine the source from which the feature vector is generated for two-class classification problems.

To avoid mathematical overflow and to save computation time, the logarithmic version of the likelihood ratio, known as the *log-likelihood ratio*, defined below, is often used instead [Duda 73]:

$$\log\gamma = \log P(x|\omega_1) - \log P(x|\omega_2).$$

In such applications, the feature vector will be labelled as class-1 ($\omega_1$) if $\log\gamma \geq 0$; otherwise, it will be classified as class-2 ($\omega_2$). For example, [Su 94a] uses the log-likelihood ratio to determine whether a word bigram (or trigram) belongs to the compound class or the non-compound class in a compound noun extraction task where the normalized frequency, mutual information and parts of speech are used in the feature vector.

· **Mutual Information**

The *mutual information* of *x* and *y*, $I(x;y)$ [Blahut 87], is the log-likelihood ratio of the joint probability of events *x* and *y* over the probability that these two events will happen independently. In other words,

$$I(x;y) \equiv \log_2\left\{\frac{P(x,y)}{P(x)\cdot P(y)}\right\}.$$

It provides a measure of the degree of dependence between two events. Intuitively, $I(x;y) >> 0$ when *x* and *y* are highly associated, $I(x;y) \approx 0$ when *x* and *y* are independent, and $I(x;y) << 0$ if *x* and *y* are complementarily distributed. Therefore, it can be used to measure the degree of "word association" between two words. For instance, the mutual information between "strong" and "tea" is greater than the mutual information between "powerful" and "tea" [Church 88]. Therefore, "strong tea" is more likely to be a lexicon entry than is "powerful tea."

· **Entropy and Perplexity**

The *entropy* of a random variable X represents the average uncertainty of X [Blahut 87]. For example, the expected number of word candidates, in bits, (i.e., the expected number of binary decisions) at a decision point (or state) *k* can be estimated by the entropy, *H*, associated with the distribution $P(w_i|k)$ [Lee 88]:

$$H(W|k) = -\sum_{i=1}^{V}\left\{P(w_i|k)\cdot\log_2 P(w_i|k)\right\},$$

where $P\left(w_i|k\right)$ is the probability of candidate word $w_i$ at state k, and *W* denotes a sequence of *V* words $w_1,w_2,\ldots,w_V$, which can be accepted at state k. Alternatively, the number of binary decisions can be expressed in a real number with the *perplexity* (*Q*) at the decision point [Lee 88] as:

$$Q(W|j) = 2^{H(W|k)}.$$

In natural language applications, the entropy or perplexity metrics can be used to indicate how random the neighbors W of a particular word k are. For instance, [Tung 94] used the entropy measure to determine whether an n-word chunk belongs to one lexicon unit; if the left and right neighbors of the chunk are randomly distributed, which indicates that there are possible natural break points between the chunk and its neighbors, then the chunk is likely to be a lexicon unit; otherwise, the chunk tends to appear simultaneously with its neighbors, and hence is less likely to be a lexicon unit by itself.

· **Dice**

The dice metric is commonly used in information retrieval tasks [Salton 93] to identify closely related binary relations. It has been used for identifying bilingual collocation translation [Smadja 96]. The dice metric for a pair of words x, y is defined as follows [Smadja 96]:

$$D_2(x,y) = \frac{P(x=1,y=1)}{\frac{1}{2}[P(x=1)+P(y=1)]},$$

where x=1 and y=1 correspond to the events where x appears in the first place and y appears in the second place, respectively. It is another indication of word co-occurrence which is similar to the mutual information metric. It has been suggested in some researches (e.g., [Smajda 96]) that this feature is a better indication of word co-occurrence than is (specific) mutual information in some cases since it discards less informative events corresponding to the x=0 and y=0 (0-0 match) cases in estimating word co-occurrence. For instance, if x represents a term in the source language and y is a possible translation of x in the target language, than it is possible to evaluate the dice metric between such a translation pair and to tell whether y is the preferred translation of x.

## 3.4 Parameter Estimation and Smoothing

### 3.4.1 Parameter Estimation

The operation of the classifiers depends on the values assigned to the parameters of the system. Therefore, the parameter values encode all statistical knowledge of a CBSO model. The parameter estimation process thus corresponds to the knowledge acquisition process of such a model. In the parameter estimation process, the parameters are regarded as variables whose values are to be determined based on the observation of a training set. Various estimation criteria can be adopted so that the estimated values objectively optimize the stochastic behaviors of the training data. The most commonly used criterion for parameter estimation is the *maximum likelihood estimation* (MLE) criterion [Papoulis 84]; the "best" estimated parameter values are the set of values which maximizes the likelihood of obtaining the (given) training set.

By definition, given the training data $X = \{x_1, x_2, \ldots, x_n\}$, the objective of the MLE estimator is to find the parameter set $\Lambda$ that maximizes the likelihood function $P(X|\Lambda)$. If the $x_i$'s are assumed to be independent, then the likelihood function can be rewritten as follows:

$$P(X|\Lambda) = \prod_{i=1}^{n} p\left(x_i|\Lambda\right).$$

To find the parameter values which maximize the likelihood function, we can take partial derivatives on the likelihood function with respect to all the parameters and set them to zeros. The solutions of the equations then correspond to the maximum likelihood estimate of the parameters. In other words, the maximum likelihood estimator can be acquired by resolving the solutions of

$$\nabla_\Lambda f(P(X|\Lambda)) = 0,$$

where $\nabla_\Lambda f(\cdot)$ is the gradient of the function $f(\cdot)$ with respect to the parameter set $\Lambda$, and $f(\cdot)$ can be any monotonically increasing or decreasing function of $P(X|\Lambda)$. One convenient function for $f(\cdot)$ is the natural logarithm function, which makes the above constraint on $\Lambda$ take the following equivalent form:

$$\sum_{i=1}^{n} \nabla_\Lambda \log\left\{ p\left(x_i|\Lambda\right)\right\} = 0.$$

For example, the ML estimate for the mean $\mu$ of a Gaussian probability density function (pdf) with a known variance is the sample mean, i.e., $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$; and the MLE of the probability of an event is the relative frequency of the event occurring in the sample space, i.e., $P_{ML}(e) = \frac{r}{N}$ for the event *e* occuring *r* times out of a total of *N* trials.

However, the MLE is quite unreliable when the training data is insufficient. For example, in a case with sparse training data, the events that do not appear in the training set will be assigned a zero probability by the ML estimator. This is inappropriate for most applications. Therefore, effective smoothing of the parameters of the null events should be adopted to reduce the degree of estimation error.

A variety of parameter smoothing methods have been proposed in the literature. Two of the most frequently adopted methods, Good-Turing's formula [Good 53] and the back-off procedure [Katz 87], will be discussed in the following sections.

### 3.4.2 Good-Turing's Formula [Good 53]

Let $N$ be the number of training tokens and $n_r$ be the number of events that occur exactly $r$ times. Then, the following equation holds: $N = \sum_r r \cdot n_r$. The maximum likelihood estimate $P_{ML}$ for the probability of an event $e$ occurring $r$ times is known to be $P_{ML}(e) = \frac{r}{N}$. The estimate based on Turing's formula [Good 53] is given by $P_{GT}(e) = \frac{r^*}{N}$, where $r^* = (r+1)\frac{n_{r+1}}{n_r}$.

The total probability estimate, using Turing's formula, for all the events that actually occur in the sample space is equal to

$$\sum_{e:C(e)>0} P_{GT}(e) = 1 - \frac{n_1}{N},$$

where $C(e)$ stands for the frequency count of the event $e$ in the training data. This, in turn, leads to the following equation:

$$\sum_{e:C(e)=0} P_{GT}(e) = \frac{n_1}{N}.$$

According to Turing's formula, the probability mass $\frac{n_1}{N}$ is then equally distributed over the events that never occur in the sample. In simple language models, such as the n-gram models, the number of parameters can be calculated easily. Therefore, the number of events that never occur, i.e., $n_0$, can be set to the number of possible parameters minus the number of events that have already appeared in the training corpus. However, depending on the real nature of the language, not all possible parameters should really be given nonzero probabilities. Therefore, another way to estimate $n_0$ is to extrapolate it from $n_r$ for all $r > 1$.

## Back-off Procedure [Katz 87]

In 1987, Katz proposed a *back-off* procedure to estimate m-gram parameters, i.e., the conditional probabilities of words given the (m-1) preceding words [Katz 87]. This procedure is summarized as follows:

$$P_{BF}(w_m | w_1^{m-1}) = \begin{cases} P_{GT}(w_m | w_1^{m-1}), & if\ C(w_1^m) > 0, \\ \alpha(w_2^{m-1}) \cdot P_{BF}(w_m | w_2^{m-1}), & if\ C(w_1^m) = 0 \,\&\, C(w_2^m) > 0, \\ P_{BF}(w_m | w_2^{m-1}), & if\ \sum_{w_m} C(w_1^m) = 0, \end{cases}$$

where $P_{GT}(\cdot)$ and $P_{BF}(\cdot)$ are the probabilities estimated with the Good-Tuning formula and Back-off procedure, respectively, $C(\cdot)$ is the frequency count for a word string and

$$\alpha\left(w_1^{m-1}\right) = \frac{1 - \sum_{w_m : C\left(w_1^m\right) > 0} P_{BF}\left(w_m \middle| w_1^{m-1}\right)}{1 - \sum_{w_m : C\left(w_1^m\right) > 0} P_{BF}\left(w_m \middle| w_2^{m-1}\right)}$$

is a normalization factor that makes

$$\sum_{w_m : C\left(w_1^m\right) > 0} P_{BF}\left(w_m \middle| w_1^m\right) + \sum_{w_m : C\left(w_1^m\right) = 0} P_{BF}\left(w_m \middle| w_1^m\right) = 1.$$

Compared with Good-Turing's formula, the probability for an m-gram that does not occur in the training set is backed-off to refer to its corresponding (m-1)-gram probability. Taking the part-of-speech tagging as an example, we suppose that three events $(\text{p,n,n})$, $(\text{p,art,n})$, $(\text{n,v,n})$ in the trigram model are not found in the samples, where "art", "n", "p", "v" stand for lexical tags of "article", "noun", "preposition", and verb", respectively. We further suppose that $\frac{n_1}{N} = 0.03$ by Good-Turing's formula. Since $n_0 = 3$, the probabilities of these three events, $P(\text{n}|\text{n,p})$, $P(\text{n}|\text{art,p})$, $P(\text{n}|\text{v,n})$, are assigned equally to be 0.01 by Good-Turing's smoothing method, while these probabilities, using the Back-Off method, are distributed according to $P(\text{n}|\text{n})$,$P(\text{n}|\text{art})$, and $P(\text{n}|\text{v})$, respectively. For instance, supposing $P(\text{n}|\text{n}) = 0.1$, $P(\text{n}|\text{art}) = 0.2$, and $P(\text{n}|\text{v}) = 0.3$, the estimated probabilities smoothed using the Back-Off method are

$$\begin{aligned}
P(\text{n}|\text{n,p}) &= 0.03 \times \frac{P(\text{n}|\text{n})}{P(\text{n}|\text{n}) + P(\text{n}|\text{art}) + P(\text{n}|\text{v})} = 0.005, \\
P(\text{n}|\text{art,p}) &= 0.03 \times \frac{P(\text{n}|\text{art})}{P(\text{n}|\text{n}) + P(\text{n}|\text{art}) + P(\text{n}|\text{v})} = 0.01, \\
P(\text{n}|\text{v,n}) &= 0.03 \times \frac{P(\text{n}|\text{v})}{P(\text{n}|\text{n}) + P(\text{n}|\text{art}) + P(\text{n}|\text{v})} = 0.015.
\end{aligned}$$

### 3.5 Automatic Feature Selection

The purpose of feature selection is to choose a subset of discriminative features for the language processing task. By selecting the features properly, the dimension of the feature space can be reduced withoutsignificantly degrading system performance. In some cases, the system performance can even be improved since noisy features are discarded and the estimation error of the parameters is reduced due to a smaller number of parameters as a result of the smaller feature dimension. In general, selecting features by hand is costly, and often the features selected in an intuitive way cannot optimize the system performance. Hence, a procedure for automatically selecting features is highly desirable.

#### 3.5.1 Sequential Forward Selection (SFS) [Devijver 82]

SFS is a simple bottom-up searching procedure which finds the best feature sequence sequentially

[Devijver 82]. The same technique can also be used to find the best rule order for a set of rules. The selection method adds one new feature to the feature set in each iteration. Initially, there is no feature in the feature set. At each iteration, a new feature is selected from the remaining features not in the feature set so that the newly formed feature set yields the maximum value according to a criterion function. The SFS algorithm can be outlined as follows:

1. Suppose that $\Phi=\{\lambda_1,\lambda_2,\ldots,\lambda_D\}$ is the feature set containing D features, and that k features have been selected to form the feature set $\Lambda_k$. Initially, k=0 and $\Lambda_0=\{\varphi\}$ is empty.
2. Find the feature $\lambda_m$ from the available set, $\Phi$-$\Lambda_k$, so that

$$\lambda_m = \underset{\lambda_i}{\operatorname{argmax}}\ \Gamma\left(X;\ \Lambda_k \cup \{\lambda_i\}\right),\ \left(\lambda_m \notin \Lambda_k\right),$$

   where $\Gamma(\cdot)$ is a pre-defined characteristic function.
3. Add the feature $\lambda_m$ to form the new feature set $\Lambda_{k+1} = \Lambda_k \cup \lambda_m$ .
4. Go to step 2 until the desired criterion is satisfied.

For instance, in the grammar checker application [Liu 93], 127 pattern rules are used to detect ungrammatical errors. At the first iteration, each of the 127 rules is applied independently to detect errors. The rule which maximizes a pre-defined score (corresponding to the number of detected errors minus the number of false alarms) is selected to be in the rule set $\Lambda$ , and the other 126 rules are left in the original rule set $\Phi$ . At the second iteration, each rule in set $\Phi$ (which now consists of 126 rules) is combined with all the rules in $\Lambda$ (which contains only one rule in this case); the score of each combination is examined. Again, the rule with the highest score in combination with rule set $\Lambda$ is added to rule set $\Lambda$ . This procedure is repeated until a pre-defined number of rules in $\Phi$ are selected or when the score begins to decrease as new rules are incorporated.

It has been found that the score is not always increased by adding rules to $\Lambda$ , which means that there is redundancy and conflict among rules or that there are rules which introduce many false alarms. This trend is shown in the following figure. For instance, region I of the figure shows that the overall score is increased initially when complementary rules are included. However, as more and more rules are included, redundancy or conflict among rules might prevent the score from increasing, as shown in flattened region II of the figure, and the score even decreases when more rules are applied, as shown in region III of the figure. Removal of redundant and conflicting rules is, generally, not an easy job for non-experts, or even for the linguistic experts. Nevertheless, rules of this kind can be detected easily by using the SFS algorithm.

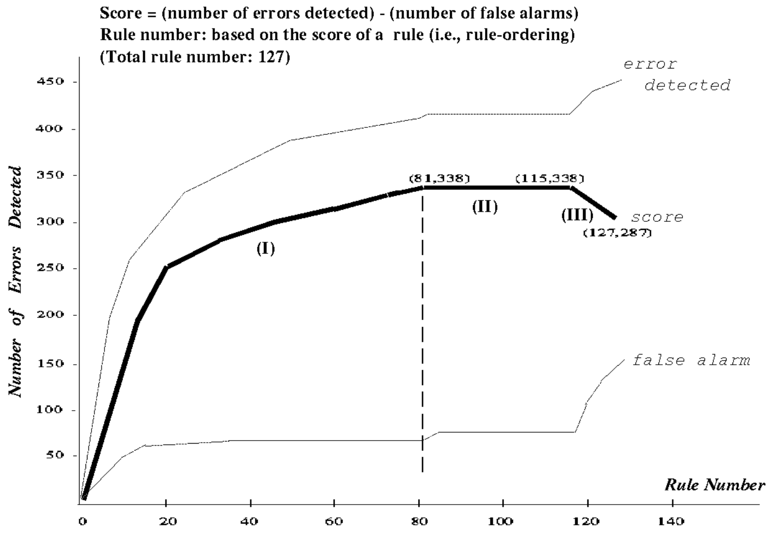


**Figure 1** Number of Rules *vs* Overall Score in SFS [Liu 93]

With the SFS algorithm, the statistical dependency between different features or rules is considered because this algorithm selects successive features or rules with reference to the current rule or feature set. However, SFS is a suboptimal search method because it does not jointly take all the feature combinations into consideration. Readers who are interested in other more advanced and complicated procedures for feature selection are referred to [Devijver 82].

### 3.5.2 Classification and Regression Trees (CART) [Breiman 84]

CART is basically a binary decision tree constructed by repeatedly splitting the tree nodes. The data of a node are split according to the most significant feature which minimizes a criterion function, usually referred to as an impurity measure. The tree grows until all the terminal nodes are either pure or contain the tokens which cannot be differentiated into different classes using the current available feature set; the former case means that the tokens associated with the terminal nodes are all correctly classified with the set of features along the branches of the classification tree; and the later case means that the data cannot be classified into correct classes using the currently available feature set. In the later case, the class associated with the node is determined by the majority-vote policy.

Taking the part-of-speech tagging model in [Lin 96] as an example, the features listed in the following are considered as potentially useful features for choosing the part-of-speech of a word:

- the left-2, left-1, right-1, and right-2 parts-of-speech of the current word;
- the left-1 and right-1 words of the current word;
- the distance (number of words) from the beginning of the sentence to the current word;
- the distance from the current word to the end of the sentence;
- the distance to the nearest right-hand side noun from the current word;
- the distance from the nearest left-hand side verb to the current word;
- the distance to the nearest right-hand side verb from the current word.

The impurity measure used to split the tree nodes is usually defined as *i(t)=M(t)+E(t)*, where *M(t)* is the number of misclassified tokens in node *t*, and *E(t)* is the entropy of node *t*.

Once the initial classification tree is constructed, the tree is usually pruned to an optimal classification tree, which minimizes the number of errors of the validation data in a cross-validation set. This pruning step can often prevent the classification tree from being over-tuned by the training data.

The pruned classification tree for the word "out" is shown in the following figure [Lin 95]. In this example, only four questions are asked along the branches to determine whether the part of speech of "out" is IN ("general preposition") or RP ("prepositional adverb which is also particle").

*(Q1) Is the next word "of"?*

*(Q2) Is the part of speech of the previous word "VBN"?*

*(Q3) Is the distance to the nearest verb on the right-hand side less than or equal to 8?*

*(Q4) Is the distance from the nearest verb on the left-hand side less than or equal to 8?*

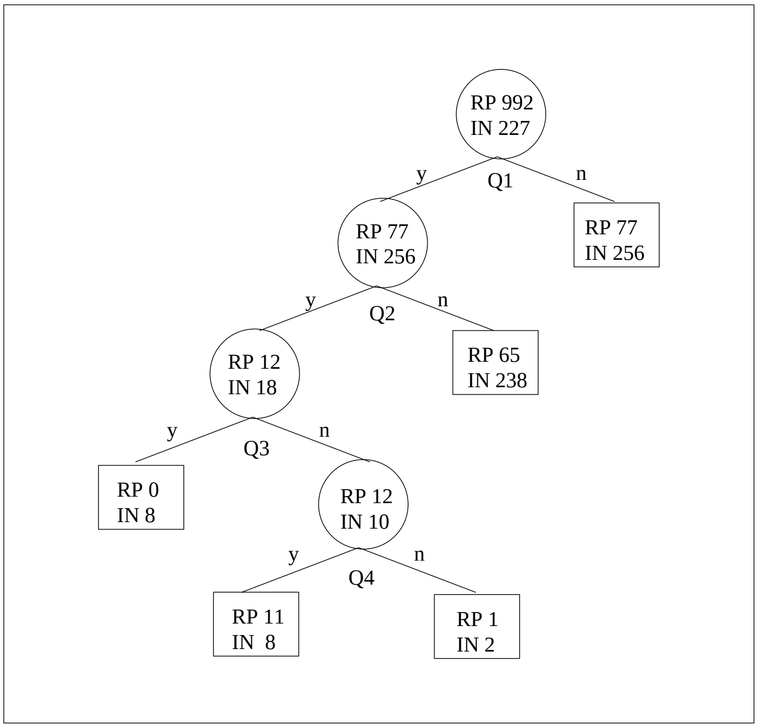


**Figure 2** Example: the pruned classification tree for the word "out" [Lin 95].

### 3.6 Clustering

An effective way to improve the sparse data problem is to reduce the number of parameters by clustering events into classes. The members in the same class possess similar characteristics. For example, the words *in*, *on*, and *at* can be clustered into the class designated as 'prepositions'; the words *Sunday*, *Monday*, *Wednesday*, …, *Saturday* can be assigned to the class designated as 'the days of a week.' Many classes are well defined in traditional linguistics. For instance, parts of speech correspond to syntactic classes, which have been proved to be useful in many NLP applications. In many applications, however, the required class information may not be available. In the following sections, two *automatic* clustering techniques, namely *dynamic* clustering and *hierarchical* clustering [Devijver 82], are introduced.

### 3.6.1 Dynamic Clustering

The dynamic clustering approach is an iterative algorithm employed to optimize a clustering criterion function. In the dynamic clustering algorithm, the number of clusters K is usually specified beforehand. At each iteration of the dynamic clustering algorithm, data are assigned to one of the clusters according to a distance (similarity) function. A new partition is thus formed. Afterwards, the representative of each cluster, which is usually defined as the mean of the data in the cluster, are updated based on the new partition. The new cluster model is then used successively in the next interation to reclassify the data. The iterative procedure continues until a desired criterion is satisfied. A typical example of dynamic clustering is the K-means clustering method [Devijver 82, Schalkoff 92], which is described as follows:

*Initialization*: Arbitrarily partition the data set $Y = \{y_1, y_2, \ldots, y_n\}$ into $K$ clusters, $C_1$, $C_2$, …, $C_K$, where $C_j$ is represented by its mean vector $\mu_j$ over $n_j$ data, j=1, 2, …, K, and

$$\mu_j = \frac{1}{n_j} \sum_{i=1}^{n_j} y_{ji}, \; y_{ji} \in C_j,$$

where $y_{ji}$ is the *i*-th token of class j, and $\mu_j$ is the mean of class j.

(One way to do this is to randomly pick out K tokens as the initial centroids of the K clusters and then use the K centroids to classify the data.)

*Step1:* Assign each data $y_i$, i=1, 2, …, n, to cluster $C_j$ if

$$j = \underset{k}{\operatorname{argmin}} \; D(y_i, \mu_k),$$

where $D(y_i,\mu_k)$ is the distance between data $y_i$ and the mean of cluster $k$.

(Note: the minimum distance criterion can be replaced by other criteria.)

*Step2:* Recalculate the mean vectors $\mu_j$, as in the Initialization step, j=1, 2, …, K.

*Step3:* Terminate the clustering procedure if the mean vectors remain unchanged or the convergence criterion is satisfied. Otherwise, go to step 1.

A few variants of the K-mean algorithm are possible. For instance, we can start with one cluster and generate a new cluster in each iteration by splitting one existing cluster until K clusters are obtained. In each iteration, one cluster is selected based on a selection criterion. The mean vector $\mu$ of the selected cluster is replaced by two vectors, $\mu_1$ and $\mu_2$, that are slightly shifted from the original mean by a small vector $\delta$ in two opposite directions, i.e.,

$$\mu_1 = \mu + \delta$$
$$\mu_2 = \mu - \delta.$$

Then, all the tokens in the selected cluster are re-classified with respect to the two new means.

### 3.6.2 Hierarchical Clustering

In contrast to the dynamic clustering procedure, the hierarchical clustering is noniterative. The hierarchical clustering algorithm is performed in a bottom-up fashion, where two of the most similar clusters are merged to form a new cluster at each stage. Since each merging action will reduce the number of clusters by one, this algorithm terminates after *n*-1 steps, where *n* is the number of data. In addition, the number of clusters in the hierarchical clustering algorithm need not be known *a priori*. The algorithm of the hierarchical clustering algorithm is shown as follows:

*Initialization*: Each data point in $Y = \{y_1, y_2, \ldots, y_n\}$ represents an individual cluster, i.e.,

$C_j = \{y_j\}$, $j=1, 2, \ldots, n$.

*Step1:* Find $C_p$ and $C_r$ such that $(p,r) = \underset{\forall(j,k), j \neq k}{\operatorname{argmin}} \mathrm{D}\left(C_j,\ C_k\right)$, where $D(C_p, C_r)$ is the distance measure between clusters *p* and *r*.

*Step2:* Merge $C_p$ into $C_r$ , and then delete $C_p$.

*Step3:* Go to step1 until the number of clusters is equal to 1.

## 3.7 Supervised Learning and Unsupervised Learning

When estimating the system parameters, the maximum likelihood criterion is often used so that the joint likelihood of the training data, as calculated using the MLE-estimated values, is maximal among all the estimates. Estimated values which maximize the likelihood of the training data, however, do not necessarily maximize the system performance (e.g., minimal error rate) when they are applied for classification by the classifiers. The major goal of an NLP system, however, is to maximize system performance. Therefore, it is desirable to adjust the initial estimates to achieve the best system performance. This can be done by adjusting the parameters according to the scores given to a misclassified instance when an input is misclassified. Such parameters are then adjusted so as to reduce the number of errors.

Parameter learning can be conducted in two different modes: supervised and unsupervised. The major difference between supervised learning and unsupervised learning depends on whether there is a pre-labeled corpus available for learning the system parameters. With a pre-labeled corpus, the parameters can be trained (adapted) in supervision of correct labels. Otherwise, unsupervised learning must be adopted, which usually performs a labeling step and a re-estimation step iteratively. At each iteration, re-estimation is realized according to the labels produced in the labeling stage, which, in turn, is based on the current estimates of the parameters. These two kinds of learning algorithms are described as follows.

### 3.7.1 Supervised Adaptive Learning

Although MLE possesses many nice properties [Kendall 79], the criterion for maximizing the likelihood value is not equivalent to that for minimizing the error rate in the training set. This is because correct classification (disambiguation) only depends on the **ranks**, rather than on the likelihood values, of the competing candidates. Therefore, adaptive learning algorithms aimed at enhancing the model's discrimination power or minimizing the training set error rate have been widely used [Su 94b, Chiang 92a]. A general adaptive learning procedure is used to iteratively adjust model parameters so as to minimize the risk, i.e., the average loss, according to the following steps:

*Initialization:* Initialize the parameters using maximum likelihood estimation and some parameter smoothing methods.

*Step1:* Calculate the mis-classification distance $d$ for each training token and then determine the corresponding loss function $l(d)$, which is a function of the distance $d$. An example of the miss-classification distance and loss function is shown below [Amari 67]:

$$d = {}^{1}SC - {}^{c}SC$$
$$l(d) = \begin{cases} \tan^{-1}\left(\frac{d}{d_0}\right), & if\ d > 0 \\ 0, \text{ otherwise,} & \end{cases}$$

where ${}^{1}SC$ and ${}^{c}SC$ denote the scores of the top and correct candidates, respectively.

*Step2:* Adjust the parameters such that the expected risk function $\overline{R} = E[l(\boldsymbol{d})]$ decreases. Adjustment of parameters $\Delta\Lambda^{(t)}$ at the *t*-th iteration can be expressed as follows:

$$\Lambda^{(t+1)} = \Lambda^{(t)} + \Delta\Lambda^{(t)},$$
$$\Delta\Lambda^{(t)} = -\varepsilon(t) U \nabla \overline{R},$$
$$\overline{R} \approx \frac{1}{N} \sum_{i=1}^{N} l\left(d_i\right),$$

where $\varepsilon(t)$ is a learning constant which is a decreasing function of *t; U* is a positive definite matrix for controlling the speed of convergence, which is usually set to a unity matrix in most applications; and $\overline{R}$ is approximated as the statistical mean of the loss for all the N misclassified instances.

*Step3:* Terminate if a predefined criterion is satisfied.

By adjusting the parameters of the misclassified instances, the performance over the training set can be maximized. However, such parameters do not guarantee satisfactory performance for unseen testing data due to possible statistical variation between the training set and the testing set. In order to maintain good performance in the testing set, robustness issues need be considered [Su 94b]. In [Su 94b], the learning procedure is modified such that the parameters are adapted not only for misclassification cases, but also for correct classification cases when the score difference between the top-two candidates is less than a preset margin. This will ensure that the correct candidate has a score that is larger than the score of the most competitive candidate (i.e., the one having the second highest score) by a sufficiently large safety margin in the training set. With such a safety margin, the correct candidates, even in the unseen testing set, are likely to have the highest scores among all the competitors. The robustness of the system can thus be improved.

### 3.7.2 Unsupervised Learning

Since supervised learning needs a pre-labeled corpus, it may not be affordable for certain applications in which the cost of pre-labeling is quite expensive. In this case, an unsupervised learning procedure, which adopts the re-estimation procedure to self-label the corpus, is preferred. Two commonly used re-estimation procedures, namely the EM algorithm and Viterbi-training algorithm, are discussed in the following sections.

#### Expectation and Maximization (EM) Algorithm [Dempster 77]

An EM algorithm is an unsupervised learning process which iteratively conducts an *expectation* step, followed by a *maximization* step until a predefined criterion is satisfied. Formally, suppose that X and Y are two random vectors whose density functions are $f(\boldsymbol{x} \mid \Lambda)$ and $g(\boldsymbol{y} \mid \Lambda)$, respectively, where $\Lambda$ is the parameter set under consideration. The random vector X cannot be observed directly unless through the random vector Y. The mapping from X to Y is a many-to-one mapping. The major goal of an EM algorithm is to find the values of $\Lambda$ which maximize $g(\boldsymbol{y} \mid \Lambda)$ given $\boldsymbol{y}$ by making use of the associated density $f(\boldsymbol{x} \mid \Lambda)$, under which a refined model can be made or the modeling work is easier. Furthermore, let $\boldsymbol{t}(\boldsymbol{x})$ denote sufficient statistics [Papoulis 90] of $\boldsymbol{x}$, which contains sufficient information for estimating $\Lambda$, and let $\Lambda^{(p)}$ denote the parameter values after $p$ iterations. The next iteration can be expressed in the following two steps:

*Expectation step*: Estimate the sufficient statistics $\boldsymbol{t}(\boldsymbol{x})$ by finding

$$\boldsymbol{t}^{(\boldsymbol{p})} = E\left(\boldsymbol{t}(\boldsymbol{x}) \middle| \boldsymbol{y}, \Lambda^{(p)}\right).$$

*Maximization step*: Determine $\Lambda^{(p+1)}$ by using maximum likelihood estimation for maximizing $h(t(\boldsymbol{x}) \mid \Lambda)$. $h(.)$ is the density function of the model from which $t(\boldsymbol{x})$ is generated, which can be easily obtained from $f(\boldsymbol{x} \mid \Lambda)$.

The EM procedure continues iteratively until the parameters $\Lambda$ converges. An example of automatic part-of-speech tagging using the EM procedure is given as follows. The tagging task can be formulated as determining the parts-of-speech $T\ \left(= t_1 t_2 \ldots t_n\right)$ given the input word sequence $W$ of $n$ words $w_1 w_2 \ldots w_n$ such that the probability $P(T|W)$, or equivalently $P(T,W)$, is maximized. Using the commonly adopted bigram tagging model, the tagging formula can be represented as follows [Church 88]:

$$P(T,W) \approx \prod_{i=1}^{n} P\left(w_i \middle| t_i\right) P\left(t_i \middle| t_{i-1}\right) .$$

The parameters which need to be specified are the probabilities $P\left(w_i \middle| t_i\right)$ and $P\left(t_i \middle| t_{i-1}\right)$, which are assumed to be uniformly distributed initially.

The EM algorithm computes the following two statistics in the expection step:

(1) the expected number of transitions from tag $t_{i-1}$ to tag $t_i$ (which may not be an integer), and

(2) the expected number of occurrences for $w_i$ given that the word is assigned a tag $t_i$.

In the EM algorithm, this expectation is taken over all the possible tagging sequences. In the maximization step of the EM algorithm, the new values of parameters $P\left(t_i \middle| t_j\right)$ are reestimated using the following equation:

$$P_{EM}\left(t_i \middle| t_j\right) = \frac{\text{expected number of transitions from tag } t_j \text{ to } t_i}{\text{expected number of tag } t_j} .$$

## Viterbi Training Algorithm [Rabiner 86]

The Viterbi training procedure for the model parameters can be summarized in the following steps:

*Initial step*: Determine the initial values of the parameters according to some a priori information. Usually, the initial parameters are assumed to be uniformly distributed if there is no a priori knowledge.

*Decoding step*: Search for the optimal path using the current parameters. Generally, the optimal path is found by using a decoding procedure travelling through the underlying states, such as the parts-of-speech sequences with respect to the input words for an automatic part-of-speech tagging task.

*Reestimation step*: Re-estimate new values of the parameters based on the decoded states by MLE.

*Repeat*: Repeat the decoding and reestimation steps until stable parameter values are obtained or until the decoded states do not change any more.

For instance, in the above tagging problem, the Viterbi algorithm will utilize the dynamic programming method to find the best tag sequence $\hat{T}$ based on the current parameter values, i.e.,

$$\hat{T} = \underset{T=\{t_1 t_2 \dots t_n\}}{\operatorname{argmax}} \prod_{i=1}^{n} P\left(w_i \middle| t_i\right) P\left(t_i \middle| t_{i-1}\right) .$$

After all the sentences in the training corpus have been decoded via the Viterbi algorithm, the corpus can be viewed as if it is annotated with the "correct" parts-of-speech. Then, the typical MLE procedure can be followed.

In contrast to the EM method, in the Viterbi training algorithm, the probabilities $P(t_i \mid t_j)$ are reestimated from the training corpus, which is annotated in the previous decoding step by using the following equation:

$$P_{VT}(t_i \mid t_j) = \frac{\text{total number of transitions from tag } t_j \text{ to } t_i}{\text{total number of transitions from tag } t_j}.$$

Similar methods are employed to reestimate the values of $P(w_i \mid t_i)$.

## 4. Some Applications

The CBSO techniques have been applied to a variety of natural language tasks. Some of these tasks are discussed in the following sections.

- *Class-based Modeling* — The class-based models use automatic clustering procedures to classify data into categories based on given characteristic functions and similarity metrics. The class-based models need a smaller number of parameters; therefore, these parameters can be estimated more reliably using the same training data, compared to models which do not identify underlying classes in advance. Therefore, systems with class-based approaches are more robust. In addition, the classes formed using automatic clustering procedures usually represent some kinds of regularity in language behavior. In such cases, the classes can be helpful for enhancing lexicon features.

- *Detection* — The applications of CBSO techniques in detection include compound word detection, unknown word detection and grammar checking.

- *Disambiguation* — The ambiguities encountered in natural language processing arise at the lexical level, syntactic level and semantic level. The resolution of the ambiguities at these levels using statistical approaches has been investigated for several years. The CBSO techniques are especially useful for disambiguation tasks.

- *Error Recovery* — Natural language texts contain not only ambiguous sentences, but also ill-formed constructs. Robust parsers, which are capable of tolerating ill-formed sentences and even of recovering errors, are therefore very important for a system that must handle real world texts.

- *Alignment of parallel bilingual corpor*a — The related tasks include sentence alignment, word correspondence finding, finding collocation correspondence, and structure mapping. The aligned materials are useful for bilingual lexicography, word sense disambiguation, and machine translation.

- Machine Translation — Statistical machine translation with CBSO techniques has become more and more popular, including target lexicon selection and transfer-based translation. Constructing a CBSO MT, however, is very challenging in terms of its complexity. A two way training method will be reviewed later.

## 4.1 Class-based Modeling

Word n-gram models have been used extensively as language models both in speech recognition and in natural language processing applications. However, the number of parameters is usually too large to be estimated reliably if the number of words is large. To reduce the number of parameters, the word n-gram probability $P\left(w_k \middle| w_{k-n+1}^{k-1}\right)$ is approximated to an *n-gram class* model [Brown 92] as follows:

$$\begin{aligned} P\left(w_k \middle| w_{k-n+1}^{k-1}\right) &= P\left(w_k, c_k \middle| w_{k-n+1}^{k-1}, c_{k-n+1}^{k-1}\right) \\ &= P\left(w_k \middle| c_k, w_{k-n+1}^{k-1}, c_{k-n+1}^{k-1}\right) P\left(c_k \middle| w_{k-n+1}^{k-1}, c_{k-n+1}^{k-1}\right) \\ &\approx P\left(w_k \middle| c_k\right) P\left(c_k \middle| c_{k-n+1}^{k-1}\right), \end{aligned}$$

where $c_i$ denotes the class to which $w_i$ belongs $((k-n+1) \le i \le k)$. Suppose that there are *V* words in the lexicon, which are categorized into *C* classes, and *C is much smaller than V*, then the number of parameters will be reduced drastically from $V^n$ with the word n-gram model to $C^n + CV$ using the class-based n-gram model. For example, considering a vocabulary of 100,000 words that are categorized into 100 classes, i.e., $V = 100{,}000$, $C = 100$ and $n = 3$, the number of parameters for the word trigram model is $V^3 = 10^{15}$ while the number of parameters for the class-based trigram model is only $\left(C^n + CV\right) = 1.1 \times 10^7$ ; the number of parameters will be different by 8 orders of magnitude. Therefore, class-based models greatly reduce the parameter space in comparison with the word n-gram model.

The success of a class-based language model depends heavily on the classes being used. In most class-based approaches, classes are either pre-defined in terms of syntactic categories and semantic categories or determined automatically using some clustering methods, such as the vector quantization technique [Rabiner 93] or the clustering methods described in Section 3. For instance, parts of speech are the most commonly used classes in NLP applications. They are widely used in part-of-speech tagging [Church 89, Merialdo 91, Kupiec 92, Lin 95]. Classes of higher level have also been applied in NLP tasks. For instance, semantic scores for disambiguating syntactic ambiguities have been exploited based on certain semantic classes [Su 88, Chang 92, Chiang 96], where classes are defined in terms of semantic categories.

Automatic clustering is also of great interest in many frameworks [Ney 91, Brown 92, Chang 93]. In those approaches, automatic clustering has been carried out to optimize a characteristic function, such as mutual information or *perplexity*. The automatic clustering mechanisms used in these frameworks can be viewed as variants of the hierarchical clustering and dynamic clustering mechanisms described in Section 3. [Brown 92], for instance, automatically acquires a few classes examplified as follows before the model is used:

Friday Monday Thursday Wednesday ...
June March July ...
people guys folks fellows ...
water gas coal liquid acid ...
down backwards ashore ...

In addition, clustering researches on Chinese words have also been investigated in several works [Chang 93b, 張 94]. In [張 94], for instance, the vector quantization technique was adopted in clustering 5000 frequently used Chinese words into classes, and these classes were used to improve the performance of a speech recognition system. A few examples of the results are given here:

密集 低迷 老舊 激烈 .... 熱烈
六月 十月 三月 四月 ... 元月
凌晨 清晨 深夜 上午 ...

In the vector quantization process, the ‘vector’ for a word consists of a number of frequencies, each frequency being the co-occurrence frequency between this word and a particular left hand side neighbor of this word in all word bigram pairs; two words which have similar distribution in their co-occurrence frequencies with their left neighbors are considered similar to each other. This similarity is then used in clustering

## 4.2 Detection

CBSO techniques have been applied to detection problems for years. Some detection applications, including compound noun detection, unknown word detection and grammar checkers, will be reviewed in this section.

### Compound Noun Detection

In [Su 94a], the compound detection problem was formulated as a two class classification problem in which a word n-gram is classified as either a compound or a non-compound based on the normalized frequency of occurrence of the n-gram, the mutual information among the constituents of the n-gram and the parts of speech associated with the constituents. (See [Su 94a] for the definition of the normalized frequency and mutual information for trigrams.)

First, a list of bigram and trigram compound noun candidates are extracted from the input text. Then, the relative frequency count and the mutual information of these candidates are calculated. Next, the parts-of-speech of the words in the corpora are assigned by a probabilistic part-of-speech tagger. Given the statistics of the mutual information $I$, relative frequency count $r$, and parts of speech $T$, the classifier regards a candidate as a compound if the *likelihood ratio*,

$$\lambda = \frac{P(I,r,T|M_c) \times P(M_c)}{P(I,r,T|M_{nc}) \times P(M_{nc})},$$

is greater than 1 (or a threshold $\lambda_0$). The probabilities $P(I,r,T|M_c)$ and $P(I,r,T|M_{nc})$ represent how likely the candidate, having mutual information $I$, relative frequency count $r$, and parts of speech $T$, is to belong to the compound noun class and the non-compound class, respectively; $P(M_c)$ and $P(M_{nc})$ stand for the prior proabilities for a candidate to be generated from the compound model, $M_c$, and non-compound model, $M_{nc}$, respectively.

To make the computation feasible, $P(I,r,T|M_c)$ is simplified as follows:

$$P(I,r,T|M_c) \approx P(I,r|M_c)P(T|M_c),$$

where $P(I,r|M_c)$ is assumed to have bi-variate normal distribution, and $P(T|M_c)$ is simplified as

$$P(T|M_c) \approx P(c_0)P(c_1|c_0)P(c_2|c_1)P(c_3|c_2)$$

for any bigram whose parts of speech are $[c_1, c_2]$ and whose left/right neighbors have the parts of speech $c_0/c_3$, respectively. (Similar simplification is adopted for trigrams.) The recall is 96.2%, and the precision is 48.2% for bigrams using the above formulation. Trigrams are identified with a recall rate of 96.6% and a precision of 39.6%. With this formulation, compound words can be identified with small cost.

### Unknown Word Detection

Unknown words in the input text have been important error sources in many research works [Chen 92, Chiang 92, Lin 93]; the performance of an NLP system may deteriorate seriously if unknown words exist. For example, the Chinese word segmentation problem can be resolved with an accuracy rate of nearly 99% if there is no unknown words in the text [Chiang 92]; however, only 90-95% accuracy rate can be achieved in the case where there are unknown words [Lin 93]. To enumerate all the words in a dictionary is often impossible in real applications because new words and compound words are created every day. Therefore, it is important to develop a mechanism for detecting unknown words.

In [Lin 93], unknown words were detected by way of morphological analysis and using an unknown word detection model. A set of morphological rules were ordered and selected via the SFS algorithm to detect unknown words having regular forms, such as derived words. To identify the irregular unknown words, a two-cluster classifier, which is similar to the compound detection model, was used. Such a probabilistic model was shown to be useful in identifying irregular unknown words in the input text. Using the unknown word identification model in the Chinese word segmentation task, the number of segmentation errors was reduced by 78.34% in words and 81.87% in sentences for technical manuals. In the newspaper domain, the error reduction rates for words and for sentences were 40.15% and 34.78%, respectively. In [Tung 94], the entropy of neighboring words of an n-word chunk was used to extract potential unknown words. Manual editing was then conducted to acquire a list of unknown words. With this approach, 5366 unknown words were identified from a corpus of 178,027 sentences in about 3 hours through three identification/editing passes.

### Automatic Rule Selection in Grammar Checker

Pattern matching methods are widely used in most grammar checkers because they can be implemented easily. However, the patterns used in most such approaches are usually hand-tuned. Therefore, it is hard for such approaches to maintain consistency among the patterns when the system is scaled up.

In addition, new patterns tend to be added whenever a specific type of error occurs. Adding new patterns sometimes introduces too many false alarms. Furthermore, bad rules cannot be effectively discarded unless a good objective measure of rule-preference is provided. Considering the above-mentioned problems, the SFS algorithm, as described in Section 3.5.1, was used in [Liu 93] to automatically select a subset of patterns according to an objective function, e.g., error count. The system with the selected patterns was shown to have a better score (corresponding to the number of detected errors minus the number of false alarms) compared with the system using the whole set of patterns.

## 4.3 Disambiguation

CBSO techniques are particularly useful in handling NLP tasks which have nondeterministic characteristics. Therefore, they have been used extensively for ambiguity resolution in lexical [Church 89, Merialdo 91, Kupiec 92, Lin 95], syntactic [Chiang 92, 95] and semantic [Chang 93, Chiang 96] levels. The following sections describe how CBSO techniques are employed to the disambiguation applications in some frameworks.

### Part-of-Speech Tagging

Let $W\left(=w_1^K=w_1,w_2,\ldots,w_K\right)$ represents an input sentence of $K$ words; the tagging score of the part-of-speech sequence, $T\left(=t_1^K=t_1,t_2,\ldots,t_k\right)$, with respect to the input sentence is defined as the conditional probability $P(T|W)$, where $t_i$ denotes the part-of-speech of the *i*-th word $w_i$. The best part-of-speech sequence $\hat{T}$ can be selected among all possible sequences according to the following equation to maximize the likelihood function:

$$\hat{T} = \operatorname*{argmax}_T P(T|W) = \operatorname*{argmax}_T P(T,W) = \operatorname*{argmax}_T P(W|T)P(T).$$

For an n-gram model, the probabilities $P(W|T)$ and $P(T)$ can be simplified as follows, resulting in the well-known POS tagging model widely used in the baseline systems of many taggers [Church 89, Merialdo 91, Kupiec 92, Lin 95]:

$$P(W|T) = P\left(w_1^K \middle| t_1^K\right) \approx \prod_{i=1}^{K} P\left(w_i \middle| t_i\right), \quad \text{and} \quad P(T) = P\left(t_1^K\right) \approx \prod_{i=1}^{K} P\left(t_i \middle| t_{i-n+1}^{i-1}\right).$$

The above formulation assumes that the part-of-speech $t_i$ of the word $w_i$ depends only on the word itself and the part-of-speech of the preceding $n$ words. By using this simplification and by using the log-likelihood, the baseline model for part of speech tagging can be formulated as follows:

$$\hat{T} = \underset{T}{\operatorname{argmax}} P(T|W) = \underset{T}{\operatorname{argmax}} \sum_{i=1}^{K} \left[\log P\left(w_i \middle| t_i\right) + \log P\left(t_i \middle| t_{i-n+1}^{i-1}\right)\right].$$

A systematic analysis of part-of-speech tagging, including the effects of parameter smoothing, adaptive learning, CART, was given in [Lin 95]. A parameter tying procedure was also proposed in that approach to tie together unreliably estimated parameters using *correlated* parameters that are reliably estimated. [Lin 95] reported that the tying approach greatly reduced the number of parameters from 578,759 to 27,947 and reduces the error rate of ambiguous words by 10.4% in tagging the Brown Corpus. Readers interested in details are referred to [Lin 95].

In the case where no annotated corpus is available, part-of-speech tagging can be carried out by using the Viterbi training method or the EM algorithm as in [Merialdo 91, Kupiec 92]. The tagging performance with respect to different sizes of the seed corpus used for estimating initial parameter values was discussed in [Merialdo 91]. Especially, in [Kupiec 92], the Hidden Markov Model (HMM) [Rabiner 92], which is widely used for speech recognition, was used to model the tagging process.

### Syntax Disambiguation

Syntax disambiguation has been accomplished in various approaches, including stochastic grammars [Wright 91], probabilistic context free models [Briscoe 93] and probabilistic context sensitive models [Su 88, 91b, Chiang 92, 95]. Conventional probabilistic context free grammar relies on the following scoring function for selecting the best syntax tree:

$$P(T) = P\left(r_1^K\right) \approx \prod_{i=1}^{K} P\left(r_i\right),$$

where T is a syntax tree which is generated by applying K production rules $r_1, r_2, \ldots, r_K$. This formulation does not take context into account. Therefore, it may not be appropriate for processing natural language. In [Su 88, 91b, Chiang 92, 95], a context sensitive formulation was proposed. In this formulation, the disambiguation model is formulated as the process of finding the most likely syntactic structure $\boldsymbol{L}$ given the input word sequence $W$, i.e.,

$$\widehat{Syn} = \underset{L}{\operatorname{argmax}} \; P(\boldsymbol{L} \mid W).$$

The probability $P(\boldsymbol{L} \mid W)$ is further simplified as

$$P(\boldsymbol{L} \mid W) \quad = \quad P(\boldsymbol{L}, T \mid W) \quad \approx \quad P\left(\boldsymbol{L} \mid T\right) \times P(T|W),$$

where $\boldsymbol{T}$ stands for the parts-of-speech associated with the syntactic structure $\boldsymbol{L}$. The lexical score, $P(T \mid W)$, is defined in the same way as the standard part-of-speech tagging formula described in the previous section; and the syntactic score, $P(\boldsymbol{L} \mid T)$, can be estimated according to the following model.

Any syntactic structure can be uniquely defined by the sequence of rewriting rules applied to the input words; the rewriting sequence, in turn, can be expressed in terms of the sets of intermediate phrases (i.e., terminal or nonterminal nodes) derived in the rewriting process. Each set of such intermediate phrases, referred to as a *phrase level* [Su 88], can thus be used in representing the syntactic structure of the input sentence.

To compute the syntactic score of the following syntax tree, for instance, the syntactic structure can be decomposed into the phrase levels $L_1, L_2, \ldots, L_8$ shown in the following figure, in which $L_1$ to $L_8$ simulate the actions of an LR parser. The transitions between the phrase levels can be formulated as *context-sensitive* rewriting processes, and the transition probability corresponds to the probability of applying a particular phrase structure rule conditioned on a particular context. For example, the transition from $L_6$ to $L_7$ corresponds to applying the rule $C \rightarrow F \;\; G$ when the left context of [F G] is the phrase [B] and the right context is the sentence stop.

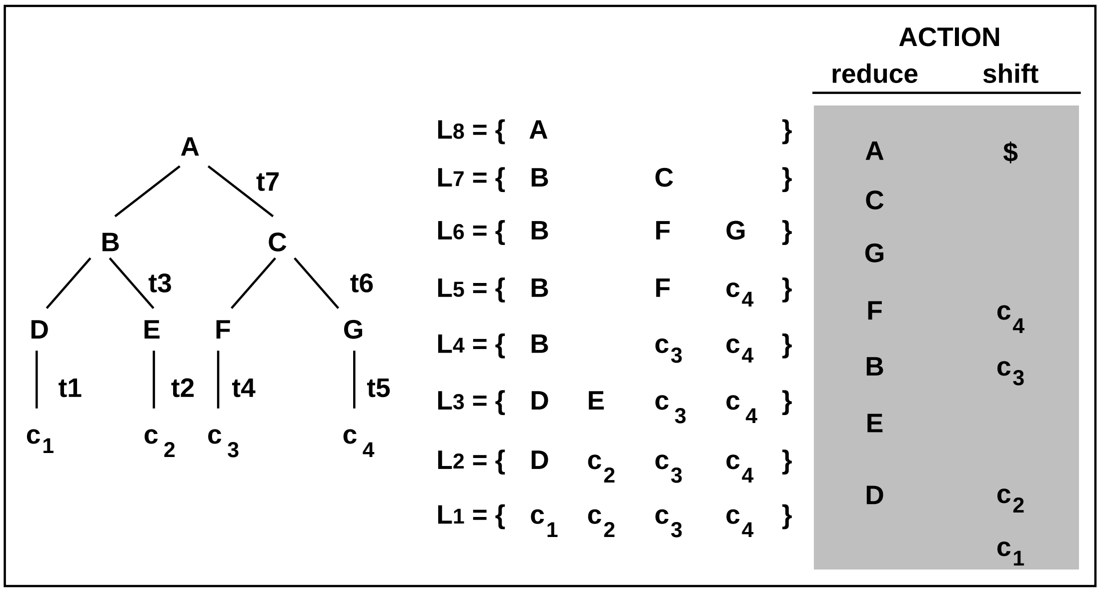

**Figure 3** The decomposition of a syntactic tree into different phrase levels.

Let $\boldsymbol{L_i}$ be the i-th phrase level and the label $t_i$ in the figure be the time index for the *i*-th state transition corresponding to a reduce action. The syntactic score of the tree in the above figure is defined as [Su 88, 89, 91, Chiang 92, 95]:

$$\begin{aligned} P(\boldsymbol{L}|T) &\equiv P(L_2^8|L_1) \approx P(L_8,L_7,L_6|L_5)P(L_5|L_4)P(L_4,L_3|L_2)P(L_2|L_1) \\ &\approx P(L_8|L_5)P(L_5|L_4)P(L_4|L_2)P(L_2|L_1). \end{aligned}$$

Each pair of phrase levels in the formula corresponds to a change in the LR parser's stack before and after an input word is consumed by a *shift* operation. Because the total number of shift actions is the same for all alternative syntax structures with respect to the same input sentence, the normalization problem, which is introduced due to the different numbers of branches in the parse trees, is resolved in such a formulation. In addition, the formulation also provides a way for considering both the *intra-level context-sensitiveity* and *inter-level correlation* of the underlying context-free grammar. With this formulation, the capability of *context-sensitive parsing* (in probabilistic sense) can be achieved using a context-free grammar.

In [Chiang 95], the effects of using the technologies of parameter smoothing, tying and robust learning on syntactic ambiguity resolution were investigated. After these technologies were applied, 70.3% parse tree accuracy was obtained, resulting in 36.6% error reduction compared with the baseline system which used the ML estimated parameters and attained only 53.1% parse tree accuracy.

## Semantics Disambiguation

Ambiguities encountered in semantics comprise the thematic roles (*cases*) of constituents and meanings (senses) of words. Statistical approaches for these two topics are numerous, e.g., Liu et al. (1990) on the PP-attachment problem; Chang (1992) on semantic modeling; Brown et al. (1991b), Dagan (1991), Gale et al. (1992), Yarowsky (1992, 1994) on word sense disambiguation. In [Chiang 96], a probabilistic model was proposed to resolve the ambiguities of both case assignment and word-sense assignment. A score function $P(NF_i,PT_j,POS_k|W)$ is defined to find the best combination of the semantic interpretation (or **normal form**) $\widehat{NF}$, the parse tree $\widehat{PT}$ and the parts-of-speech $\widehat{POS}$ for a given input sentence, $W$, using the following decision rule:

$$\begin{aligned} (\widehat{NF},\widehat{PT},\widehat{POS}) &= \underset{NF_i,PT_j,POS_k}{\text{argmax}} P(NF_i,PT_j,POS_k|W) \\ &= \underset{NF_i,PT_j,POS_k}{\text{argmax}} \{ P(NF_i|PT_j,POS_k,W) \times P(PT_j|POS_k,W) \times P(POS_k|W) \} \\ &= \underset{NF_i,PT_j,POS_k}{\text{argmax}} \{ S_{sem}(NF_i) \times S_{syn}(PT_j) \times S_{lex}(POS_k) \}, \end{aligned}$$

where $S_{syn}(\cdot)$ and $S_{lex}(\cdot)$ denote the syntactic score and the lexical score, respectively, which were discussed in previous sections. The following figure shows an example of a normal form which represents the semantic relationship between sentence constructs.

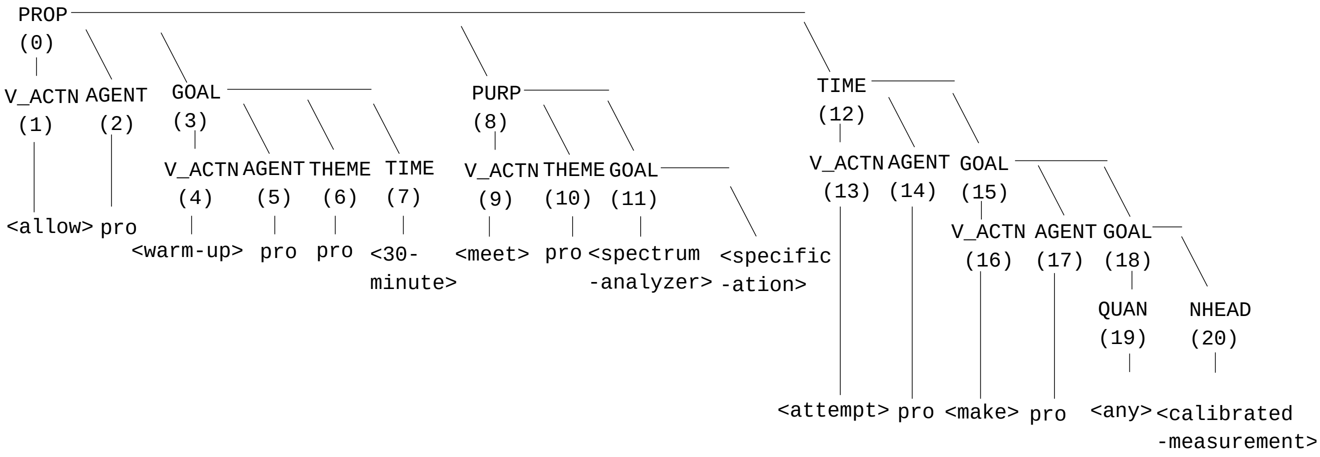


**Figure 4** An Example of a Normal Form (NF) Tree.

The semantic score, $S_{sem}(\cdot)$ , on the other hand, can be estimated as follows

$$\begin{aligned} S_{sem}\left(NF_i\right) &= P\left(NF_i \mid PT_j, POS_k, W\right) \\ &= \sum_{I_i} P\left(NF_i, I_i \mid PT_j, POS_k, W\right) \approx \sum_{I_i} P\left(s_{i,1}^{i,n}, \Gamma_{i,1}^{i,M_i} \mid L_{i,1}^{i,M_i}\right) \\ &= \sum_{I_i} P\left(s_{i,1}^{i,n} \mid \Gamma_{i,1}^{i,M_i}, L_{i,1}^{i,M_i}\right) \times P\left(\Gamma_{i,1}^{i,M_i} \mid L_{i,1}^{i,M_i}\right) , \end{aligned}$$

where $I_i = L_{i,1}, \ldots, L_{i,M_i}$ denotes a normalized structure of the corresponding parse tree, which is introduced to reduce the number of parameters; $\Gamma_{i,1}^{i,M_i} = \Gamma_{i,1}, \ldots, \Gamma_{i,M_i}$ denotes the $M_i$ case subtrees, which define the *structure* of the normal form; $s_{i,1}^{i,n} = s_{i,1}, \ldots, s_{i,n}$ stands for the word senses of the input sentence $W$. $P\left(s_{i,1}^{i,n} \mid \Gamma_{i,1}^{i,M_i}, L_{i,1}^{i,M_i}\right)$ , thus, defines the *sense score*, and $P\left(\Gamma_{i,1}^{i,M_i} \mid L_{i,1}^{i,M_i}\right)$ represents the *case score* for a particular case and sense assignment. Based on the integrated score function, the candidate with the best score value is then picked using the dynamic programming technique. When this model was tested on a testing set of 800 sentences, the MLE-based baseline system achieved accuracy rates of 56.3% for parse trees, 77.5% for case and 86.2% for word sense. After Good-Turing's parameter smoothing and the robust learning procedure were applied, the accuracy rates were improved to 77% for parse trees, 88.9% for case and 88.6% for word sense. Interested readers are referred to [Chiang 96] for details.

## 4.4 Error Recovery

Error recovery has been considered in NLP tasks for two areas, spelling correction and parsing error recovery. Spelling correction aims at correcting spelling errors of three kinds: (1) words that cannot be found in the lexicon of a given language, (2) valid but unintended words, such as *peace* and *piece*, and (3) improper usage of words, such as *among* and *between* . The conventional spelling checkers, such as Unix *spell*, are concerned only with the first type of error, and the other types of errors remain undetected. Some statistics-oriented approaches have been exploited including word-trigram methods [Mays 91], Bayesian classifiers [Gale 93], decision lists [Yarowsky 94] and a hybrid method [Golding 95, 96]. In [Golding 96], the part-of-speech trigram model was combined with the Bayesian classifier for context-sensitive spelling correction, using which substantially higher performance was reported compared with the grammar checker in Microsoft Word.

CBSO approaches are also applicable to robust parsing and error recovery. If the input sentences are grammatical with respect to the grammar defined by an NLP system, then the best parse can be selected using the syntax disambiguation mechanism as mentioned in previous sections. However, not all real world sentences strictly follow the grammar of a particular NLP system. Therefore, the syntax analysis module of a system should be able to recover the complete parse when the input sentence does not fit the grammar of the system. When there are errors in a parse, it is desirable to recover from the errors so that proper interpretation can be given to the parse for better processing in the subsequent processing steps (such as lexical transfer and structure transfer in an MT system.)

Two types of errors are commonly encountered in real text. The first type is called an isolated error, which means that such an error will not prevent other parts from being constructed except at the root node (i.e., sentence) level. For example, if an ill-formed input sentence "?? are made by DRAM" is encountered (where '??' is the missing part), then we are able to produce a verb phrase V2 as shown in the following figure. In this case, it is possible to insert a nonterminal node such as N3 (i.e., a Noun Phrase, such as "Printer buffers") so that we have a complete parse (S). In this case, recovery is not too difficult since the lack of N3 does not prevent other parts (up to V2, the Verb Phrase) from being generated until S is to be constructed. Another type of error is called an non-isolated error whose occurrence will prevent other parts from being constructed correctly. For instance, if the ill-formed sentence is "Printer buffers are ?? by DRAM", then recovering the complete parse is much more difficult since the word and hence the parts of speech is missing, which may prevent other parts (such as V1, V2 and S) from being generated correctly.

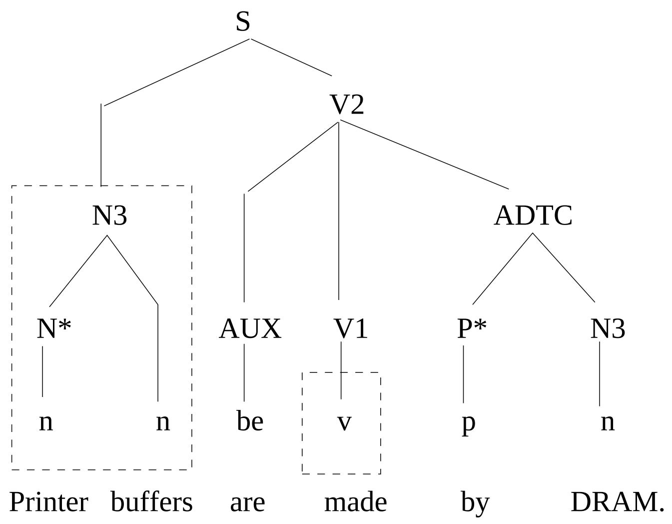


**Figure 5** Examples of parsing errors.

To recover such errors, the error recovery process was modeled in [Lin 96] as follows. The system assumes that there are several possible modification actions which can be used to patch an incomplete parse into a parse that fits the grammar of the system; the modification actions include insertion, deletion, and substitution of a node (terminal node or non-terminal node) during the parsing process. The recovery process include two stages. In the first stage, attempts are made to correct non-terminal nodes. In the second stage, part of speech (terminal node) errors are recovered using proper modification actions. The various modification actions will result in different patched parse trees. The best patch will be determined by the following scoring function, which estimates how likely a patched parse tree is to be generated in a real world text due to insertion/deletion/substitution errors made by writers:

$$P\left(L_N, R_{N-1}, \tilde{R}_{N-1}, L_{N-1}, \ldots, R_1, \tilde{R}_1, L_1 | L_1\right) \approx \frac{P\left(L_N\right)}{P\left(L_1\right)} \prod_{i=2}^{N} P\left(R_{i-1}, \tilde{R}_{i-1} | L_i\right),$$

where $R_i$ and $\tilde{R}_i$ refer to the normal reduction and modification action by which the phrase level $L_i$ is reduced and patched into $L_{i+1}$ .

Note that the score is governed by the ‘rules’ (i.e., phrase structure rules and the modification actions, if any) applied to each phrase level. In addition, the factor $P\left(L_N\right)$, which stands for the probability of the topmost phrase level, i.e., the probability of producing the particular set of root nodes, assigns a preference to the set of subtrees if they can not be reduced to a complete parse tree.

The above formulation makes it possible to recover certain types of errors in a real world written text. Hence, it provides a useful mechanism for robustly parsing ill-formed sentences as well as recovering the errors. It was reported that when errors are recovered by fitting partial parses and modifying the part of speech errors, 35% of the ill-formed inputs could be parsed to the correct full parse trees. The recall rate of brackets, measured on full trees, was increased from 68.49% to 76.6%.

### 4.5 Alignment of Parallel Bilingual Corpora

Parallel corpora, such as the Hansards corpus [Brown 91a, Gale 91a], are very useful knowledge sources for automatic acquisition of bilingual (and monolingual) knowledge. A variety of researches have investigated the use of bilingual corpora, including sentence alignment [Brown 91a, Gale 91a, Wu 94], word correspondence [Brown 91b, Gale 91b, Dagan 93], collocation correspondence [Smadja 92, Kupiec 93], word sense disambiguation [Brown 91b, Dagan 91, Gale 92] and machine translation [Brown 93, Su 95, Wu 95]. In particular, the aligned bilingual corpora at the word level are valuable for bilingual lexicography [Smadja 96] and translation. For example, the *Termight* system [Dagan 94] uses part-of-speech tagging and word-alignment technologies to extract potential candidate terms and translations, providing users with a semi-automatic way to construct a terminology glossary. In such a way, users can identify and translate technical terminology easily.

For alignment of sentences in parallel bilingual texts, sentence length was used as a feature in [Gale 91a, Brown 91a] for English-French bilingual corpora and in [Wu 93] for English-Chinese bilingual corpora. In [ Gale 91a; Wu 93], sentence length was defined as the number of characters in the sentence, while in [Brown 91a], sentence length was defined as the number of words in the sentence.

Intuitively, short source sentences tend to be translated into short sentences, and long sentences tend to be translated into long sentences or a combination of several short sentences. Therefore, most current research works use two factors to estimate the likelihood of a particular alignment. The first factor is the matching type of the aligned passages, which is a tuple consisting of the number of source sentences and the number of target sentences in matching pairs (such as 1-2 matching); the second factor is the length of the aligned source-target passages (say, 20 words in the source passage vs 24 words in the target passage). The alignment which has the maximum likelihood of being aligned according to these two factors is then considered as the best alignment.

Formally, the sentence alignment problem can be regarded as an optimization problem, which tries to find the best alignment $A^*$ between two texts $T_1$ and $T_2$ from all possible alignments. That is, to choose

$$\begin{aligned} A^*(T_1,T_2) &= \underset{A}{\operatorname{argmax}}\ P\big(A(T_1,T_2)\big) \\ &= \underset{A}{\operatorname{argmax}} \prod_{i:L_{1i}\Leftrightarrow L_{2i}\in A} P\big(\mathbf{X}(L_{1i}\Leftrightarrow L_{2i})\big) \\ &= \underset{A}{\operatorname{argmin}} \sum_i -\log\ P\big(\mathbf{X}(L_{1i}\Leftrightarrow L_{2i})\big), \end{aligned}$$

where A is a typical alignment consisting of aligned passages $\{L_{1i}\Leftrightarrow L_{2i}\}$, the $i$ -th source passage $L_{1i}$ is zero, one, or two (or more) sentences in $T_1$, and $L_{2i}$ is the corresponding sentence(s) in $T_2$. (Passages that are more than two sentences in length are not considered in most current alignment research works.) In the above formulation, $\mathbf{X}\left(L_{1i} \Leftrightarrow L_{2i}\right)$ is a set of random variables, i.e., a random vector, for the aligned passages which is used to define the likelihood of a particular alignment. Moreover, in the above formulation, the passage pairs are considered as being independently aligned with one another. In [Gale 91] and [Wu 94], the random vector, **X**, consisted of the type of match, M, and a length function $\delta$ defined according to the length, $l_{1i}$, of the source passage and the length, $l_{2i}$, of the corresponding target passages. The maximum likelihood selection then is equivalent to

$$\begin{aligned} A^* &= \underset{A}{\operatorname{argmin}} \sum_i -log\ P\left(X\left(L_{1i} \Leftrightarrow L_{2i}\right)\right) \\ &\equiv \underset{A}{\operatorname{argmin}} \sum_i -log\ P\left(\left[M\left(L_{1i} \Leftrightarrow L_{2i}\right),\ \delta\left(l_{1i}, l_{2i}\right)\right]\right) \\ &= \underset{A}{\operatorname{argmin}} \sum_i -log\{P(\delta(\cdot)|M(\cdot)) \times P(M(\cdot))\}, \end{aligned}$$

where $P\left(M\left(L_{1i} \Leftrightarrow L_{2i}\right)\right)$ is the probability associated with the type of match between $L_{1i}$ and $L_{2i}$; and $P\left(\delta\left(l_{1i}, l_{2i}\right)|M(\cdot)\right)$ is the likelihood of observing the value $\delta\left(l_{1i}, l_{2i}\right)$ if the particular type of match is known.

Although the above formulation produces encouraging alignment performance based on the length information and the matching type information, we remind readers that there are other useful features which can be included in defining the random vector **X** so as to achieve better alignment performance. Such information may include bilingual lexicon information (such as the translations of the words within the passages), syntactic information (such as parts of speech and parse trees of the sentences) and semantic information (such as case information and word sense information). Depending on the information available, various alignment models can be developed based on the above general score function for alignment.

The **argmin** operation implies that the maximum likelihood selection problem can be regarded, alternatively, as a minimum distance problem if the negative of the log-scaled likelihood function is regarded as a distance measure between two aligned passages. Such a *minimum distance* searching problem is actually resolved very efficiently by using the dynamic programming technique given in [ Gale 91a] with the distance function defined as follows:

$$d = -\log\{P(\delta|\,match\_type) \times P(match\_type)\},$$

where $P(match\_type)$ (corresponding to $P\left(M\left(L_{1i} \Leftrightarrow L_{2i}\right)\right)$ ) is the prior probability of a particular type of match (e.g., 1-1 match, 1-0 match) and $P(\delta \mid match\_type)$ (i.e., $P\left(\delta\left(l_{1i}, l_{2i}\right)|M(\cdot)\right)$ ) is the likelihood of the difference in lengths between the source-target sentences for a particular type of match.

More precisely, $\delta$ is defined as $\frac{(l_2 - l_1 \cdot c)}{\sqrt{l_1 \cdot s^2}}$, where $l_1$ denotes the total length of the source sentence(s) in a matched pair, and $l_2$ denotes the total length of the corresponding sentence(s); $c$ and $s^2$ stand for the mean and the variance of $\left(\frac{l_2}{l_1}\right)$, respectively. With such a formulation, the minimum distance path and, hence, the best aligned passages, can be easily identified using the dynamic programming techniques.

The method used in [Gale 91a] produced 36 errors out of 621 total alignments (5.8%) for English-French texts, and 19 errors out of 695 (2.7%) alignments for English-German texts. Overall, there were 55 errors out of a total of 1316 alignments (4.2%). For the English-Chinese alignment case in [Wu 94], 86.4% accuracy was reported (and 95.2% could be achieved if the introductory session headers were discarded.)

For word correspondence, Gale and Church (1991b) used the $\phi^2$ statistic, a $\chi^2$-like statistic [Hoel 71], as the measure of association of pairs of words. The statistic is derived from a two by two contingency table shown as follows:

| | | |
|---|---|---|
| | *T* | |
| *S* | *a* | *b* |
| | *c* | *d* |

*S: the word in the source language*
*T: the word in the target language*
*a: the number of aligned regions that contain both S and T*
*b: the number of aligned regions that contain S but not T*
*c: the number of aligned regions that contains T but not S*
*d: the number of aligned regions that contain neither T nor S.*
*N (=a+b+c+d) is the number of aligned regions.*

The problem of testing whether the two words *S* and *T* are statistically independent is formulated as the problem of testing the hypothesis: $H_0$: $p_{ij} = p_i p_j$, where $p_{ij}$ is the probability that an (S,T) word pair will fall into the category corresponding to the *i*-th row and *j*-th column of the table; $p_i = \sum_j p_{ij}$ denotes the probability of the events corresponding to the *i*-th row; $p_j = \sum_i p_{ij}$ is the probability of the events corresponding to the *j*-th column. This test can be conducted based on the counts in the contegency table using the $\phi^2$ statistic defined as follows:

$$\phi^2 = \frac{(ad - bc)^2}{(a+b)(a+c)(b+d)(c+d)}.$$

In this way, $\phi^2$ is bounded between 0 and 1, and a high $\phi^2$ value indicates that the words $S$ and $T$ are strongly associated and, thus, may be translations of each other.

Once a set of word pairs which have high $\phi^2$ are selected, a matching procedure is conducted to match English and French words within the aligned regions using the selected pairs. When there are several possibilities for matching one source word with a target word at different target word positions, the matching procedure will select the best correspondence based on a correspondence score. Intuitively, the correspondence score prefers a correspondence which has less change in the word order of the target words when the target words are re-sorted according to their corresponding source word indices; this change in word order is defined in terms of the difference between the word index of the current target word and the word index of the preceding target word in the re-sorted list, referred to as the *slope* of the current target word. The correspondence score for J source-target pairs is, thus, defined as

$$\sum_{j=1}^{J} \log P\left(slope_j | match\right) P(match),$$

where J is the number of source words which have non-null correspondence in the target side, $P(match)$ is the prior probability of the number of source words which can be matched with a target word (the number of matching source words is classified into three cases: *match*= 1, 2 or 'many'), and $P\left(slope_j | match\right)$ indicates how likely it is that the *j*-th target word will have a slope of $slope_j$ if the number of source words which can be mapped to the *j*-th target word is known. The best correspondence is obtained by using the dynamic programming technique over all possible correspondences. The performance was evaluated on a sample of 800 sentences, where 61% of the English words were matched with some French words, and about 95% of these pairs were judged as being correctly matched. Readers who are interested in the details are referred to [Gale 91b].

Besides single word correspondences, collocational correspondences have also been explored in [Smadja 92; Kupiec 93]. In these approaches, mutual information is used as a measure of correspondence, and an iterative or EM-based re-estimation procedure is used to find the best correspondences. Such aligned bilingual materials are applicable to a variety of applications, such as word sense disambiguation [Gale 92b] and lexicography [Smadja 96]. The most complicated system for full alignment at various levels will be a system which allows two-way training for acquiring the translation knowledge of a machine translation system. Such issues will be addressed in the following sections.

## 4.6 Machine Translation and Translation Knowledge Acquisition

Another important application for CBSO techniques is statistical machine translation [Chen 91, Brown 92, Su 92, 93, Chen 95]. Since the success of a CBSO MT system relies heavily on the translation knowledge acquired, the training issues for automatically acquiring such knowledge are particularly important for a large practical system. In particular, Su et al. (1995) proposed a two-way design method for automatically acquiring the translation knowledge of a parameterized MT system from a bilingual corpus. The two-way design can also prevent literal translations from being produced as occurs with the conventional one-way design scheme [Su 95]. In [Su 95], the training process was characterized as a two-end constraint optimization problem to reflect its attempts at two-way training from a bilingual corpus. Under such an optimization approach, the system designer can concentrate on the acquisition or compilation of lexicon knowledge and the tagging of shallow syntactic and semantic features of the lexicons. Only a brief overview of the two-way MT design will be given in the following; readers should refer to [Su 95] for details.

In [Su 95], it was assumed that a sentence is analyzed into a parse tree; the parse tree is normalized into a level-1 normal form, denoted as NF1, which is acquired by compacting excessive nodes in the parse tree; NF1 is then further normalized into a second level normal form, referred to as NF2, by labeling each constituent with case information. An example of such a normal form is shown in Section 4.3. The analysis and normalization processes are governed by the phrase structure grammar G and two sets of normalization rules, $NR1_s$ and $NR2_s$ , as shown in the next figure, where the source sentence is denoted by S, and the parse tree is denoted by $PT_s$. (We shall use 's' and 't' in the subscripts of the symbols to refer to the 'source' and 'target' languages, respectively, unless there is no confusion. ) In addition, the generation operations, which are the inverse operations of the normalization operations of the target language, are directed by sets of generation rules of the various levels (GR2, GR1, and GR0); the output of such operations are the target normal forms, $NF2_t$ and $NF1_t$, target parse tree, $PT_t$, and target sentence T's that can be enumerated in the reverse direction of the analysis processes.

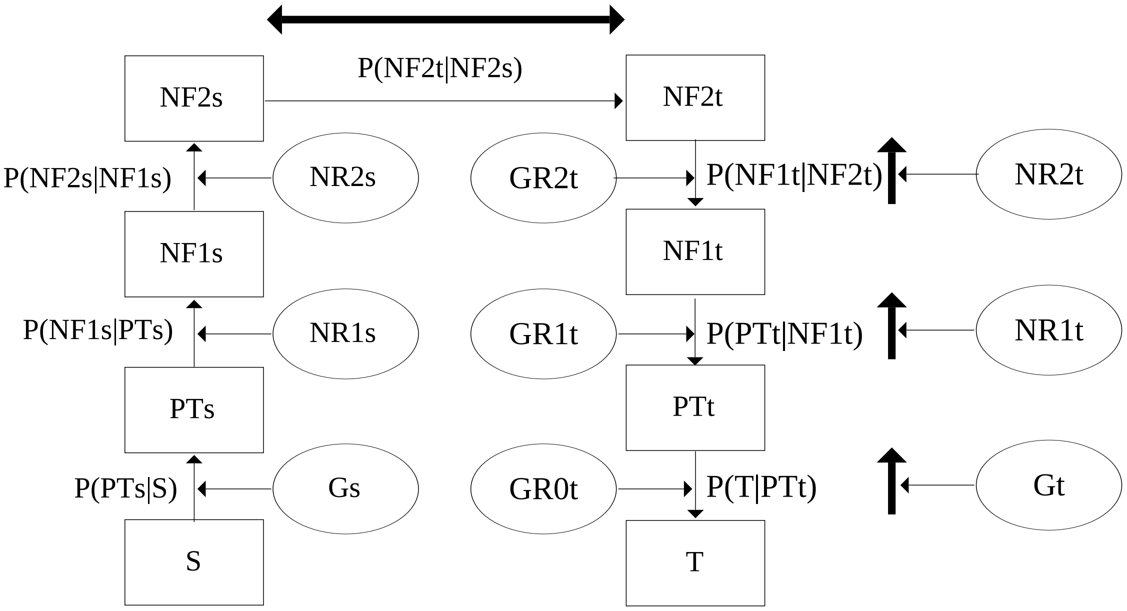


**Figure 6** Translation Model for a Parameterized MT System.

(The bold arrows indicate the flow for training the required parameters of the two languages.)

The best translation will be selected based on the transition probabilities as shown in the above translation flow. Such transition probabilities can be mapped to the analysis, transfer and generation phases of a conventional transfer based MT [Chen 91]. To acquire the translation knowledge with special emphasis on producing target sentences that fall within the target grammar, Gt, a two-way training method was proposed as follows [Su 95].

The bold arrow symbols in Figure 6 indicate that the PT's, NF1's and NF2's for *both* the source and target sentences are directly derived from the source and target sentences, respectively, based on their own phrase structure grammars and normalization rules. (The normalization rules for the target language are the reverse of the respective generation rules in Figure 6.) Therefore, all such intermediate representations will fall within the range of the sentences that will be produced by native speakers of the source and target languages; the transfer phase only *select*s the preferred candidates among such constructs. In addition, the transfer parameters are estimated based on such intermediate representations, which are derived from both the source and the target sentences of an aligned bilingual corpus. A mathematical formulation for translation and translation knowledge acquisition can be described as follows [Su 95].

According to the previously mentioned optimal classifier, the Bayesian decision to get the best target sentence , $T_{opt}$, corresponds to finding the target sentence such that

$$T_{opt} = \underset{T_i}{\operatorname{argmax}} \; P\left(T_i | S_i, I_1^{i-1}\right),$$

where $I_1^{i-1}$ represents the sets of normal forms $I_1, I_2, \ldots, I_{i-1}$ acquired earlier for the previous sentence pairs, which represent the discourse information around the current sentence, and where

$$I_i = \left\{ PT_t(i), NF1_t(i), NF2_t(i), NF2_s(i), NF1_s(i), PT_s(i) \right\}$$

represents one possible combination of the intermediate normal forms for the *i*-th sentence pair, which are derivable from the source and target sentences.

When the discourse information, namely $I_1^{i-1}$, is ignored, we can find the translation $T_i$ that maximizes the following *translation score* [Chang 93]:

$$P\left(T_i | S_i\right) = \sum_{I_i} P\left(T_i, I_i | S_i\right).$$

To reduce the computation load, the target translation, $T_i$, which has the highest *maximum path score* (i.e., $\max_{T_i, I_i} P\left(T_i, I_i | S_i\right)$) is chosen as the most preferred output instead of choosing the one whose *sum of path scores* over all possible paths (i.e., $\sum_{I_i} P\left(T_i, I_i | S_i\right)$ ) is the highest. In other words, we would prefer translation $T_i^*$ of $S_i$, where $T_i^* = \underset{T_i}{\operatorname{argmax}} \left\{ \max_{I_i} \left[ P\left( T_i, I_i | S_i \right) \right] \right\}$.

To further make the computation feasible, it is assumed that the intermediate form of the current processing phase only depends on the information acquired in the previous processing step, and that the information from other earlier processing steps can be ignored. Thus, the above formula can be further approximated as follows:

$$
\begin{aligned}
T_i^* = \underset{T_i}{\operatorname{argmax}} \{ \max_{I_i} & \left[ P\left( T_i | PT_t(i) \right) \times P\left( PT_t(i) | NF1_t(i) \right) \times P\left( NF1_t(i) | NF2_t(i) \right) \right. && \ldots(1) \\
& \times P\left( NF2_t(i) | NF2_s(i) \right) && \ldots(2) \\
& \left. \times P\left( NF2_s(i) | NF1_s(i) \right) \times P\left( NF1_s(i) | PT_s(i) \right) \times P\left( PT_s(i) | S_i \right) \right] \}. && \ldots(3)
\end{aligned}
$$

where the first three factors in (1) correspond to the generation score, the factor in (2) corresponds to the transfer score and the three factors in (3) correspond to the analysis score for one particular path (i.e., the particular set of intermediate normal forms); the individual factors (from the last factor to the first one), on the other hand, correspond to the preference for syntactic parsing, syntactic normalization, semantic normalization, semantic tree selection, normalized structure generation, surface structure generation and morphological generation, respectively. Under such circumstances, all the translation output will fall within the range of those sentences that will be produced by native post-editors; therefore, source dependency is removed.

If we have annotated the corpora with the most preferred $I_i$ for each sentence pair, then the parameters can simply be estimated by counting the number of occurrences of the various intermediate forms. A practical problem is that such annotated corpora may not exist due to the high construction cost or because the amount of annotated corpora is too small, which may induce large estimation errors. Since an unannotated bilingual corpus is usually much easier to set up than is a fully annotated corpus, one promising solution to such a problem is to train the parameters using only a bare bilingual corpus, and to consider such an optimization procedure as a two-end constrain optimization problem. By two-end constrain, we mean that we are given only a parallel corpus of the source sentence ($S_i$) and the preferred target translation ($T_i$), including the preferred lexicon information, the functional forms of the syntax structures (defined by a phrase structure grammar) and the functional forms of the semantic trees. Here, there are only the two-end constraints on the given (S,T) pairs; the other intermediate representations are left unspecified.

To estimate the parameters from an untagged bilingual corpus, the Viterbi training approach [Rabiner 86], an unsupervised learning method, is adopted. The estimation procedure is described as follows:

1. Initial Selection: For each ($S_i$, $T_i$) pair, we derive all possible parse trees, NF1 trees and NF2 trees for

$S_i$ and $T_i$, respectively, and we randomly choose a path corresponding to the analysis processes from the sentences to the NF2 trees. The source and target NF2 trees randomly selected in this way are considered to be a transfer pair for the translation.

2. Estimation: The parameters estimated using the Maximum Likelihood Estimator (MLE) for the system are estimated from the corresponding transfer pairs, parse trees and normal forms along the selected translation path. The parameters are uniquely determined once the translation paths are specified.

3. Re-selection: Compute the translation scores for the various possible paths (each path corresponding to a combination of parse tree, NF1 and NF2) with the new parameters acquired in step 2, and select the path with the largest score.

4. Re-estimation: Go to step 2, unless the parameters converge or satisfy a given stopping criterion.

To make the above procedure better, the Initial Selection step starts with a small annotated bilingual corpus as the seed [Su 94b]. This annotated seed corpus is then mixed with the other untagged corpus for estimation. In addition, to eliminate the large search space in the above process, some of the low-score combinations can be eliminated from consideration as the process proceeds; the number of truncated candidates becomes a time increasing function as the parameters are estimated better and better.

## 5. Concluding Remarks

In this paper, a brief review on the various CBSO approaches and applications in natural language processing has been given. We had identified the knowledge acquisition problem as the most challenging task in building a large scale NLP system. Hence, corpus-based statistics-oriented approaches, which are good at automatically extracting inductive knowledge, are preferred. To avoid having a large parameter space, which may not be affordable for most practical system, special emphasis has been placed on the use of probabilistic language models over high level linguistic constructs.

The central problem in designing a system based on the CBSO philosophy is to develop a language model that can optimize some performance criteria such as the minimum error rate, based on features observed in a training set. The features must be discriminative so that they can be used for better decision making. However, it is not easy to identify discriminative features; therefore, automatic feature selection approaches are desirable. The parameters to be used in the language model must be estimated from the training corpus based on an optimal estimation criterion, such as the maximum likelihood criterion, over the training set. Parameters estimated using this criterion, however, may assign zero probabilities to unseen events and, thus, may have unsatisfactory results when they are used for unseen testing set data. To resolve this problem, smoothing over the parameter values is necessary. In some cases, class information, which is required to reduce the number of parameters of a language model, is not available; automatic feature clustering techniques have, thus, been addressed in the paper. When estimating the system parameters, the training corpus may or may not be annotated using associated features. If the corpus has been annotated, robust adaptive learning, which attempts to adjust the system parameters based on misclassified instances, can be adopted to adjust the MLE estimated parameter, so that the adjusted parameters can minimize the error rate instead of maximizing the likelihood. Without an annotated corpus, unsupervised learning techniques can be applied to estimate the parameters iteratively. All such techniques form the basis for building a large scale NLP system.

CBSO approaches have been widely used and are becoming popular in research communities. Their applications range from part-of-speech tagging, syntax analysis, semantic analysis, machine translation, lexicon acquisition, corpus annotation, error recovery and more. Since large training corpora are becoming more and more available, computing power can be accessed at very low cost, and statistical optimization techniques are now well developed, it is expected that the CBSO paradigm will be one of the most promising approaches for future natural language processing development.

## Bibliography


[Amari 67] Amari, S., "A theory of adaptive pattern classifiers," *IEEE Trans. on Electronic Computers*, Vol. EC-16, pp. 299-307, June 1967

[Bahl 83] Bahl, L. R., F. Jelinek, and R. Mercer., "A Maximum Likelihood Approach to Continuous Speech Recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 2, pp. 179-190, March 1993.

[Blahut 87] Richard E. Blahut, 1987. *Principles and Practice of Information Theory*, Addison-Wesley Publishing Company. 1987.

[Breiman 84] Breiman, L., J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification And Regression Trees,* Wadsworth Inc., CA, USA, 1984.

[Briscoe 93] Briscoe, Ted, and Carroll, J., "Generalized Probabilistic LR Parsing of Natural Language (Corpora) with Unification-based Grammar," Computational Linguistics, Vol. 19, No. 1, pp. 25-59, 1993

[Brown 90] Brown, Peter F., John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin, "A Statistical Approach to Machine Translation," *Computational Linguistics*,, vol. 16, no. 2, pp. 79-85, June 1990.

[Brown 91a] Brown, P. et al., "Aligning Sentences in Parallel Corpora," *Proceedings of 29th Annual Meeting of the Association for Computational Linguistics*, pp. 169-184, California, USA, June 18-21, 1991.

[Brown 91b] Brown, P. et al., "Word-Sense Disambiguation Using Statistical Methods," *Proceedings of 29th Annual Meeting of the Association for Computational Linguistics*, pp. 264-270, California, USA, June 18-21, 1991.

[Brown 92] Brown, Peter F., Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer, "Class-Based N-gram Models of Natural Language," *Computational Linguistics*,, vol. 18, no.4, pp. 467-479, 1992.

[Brown 93] Brown Peter F., Vincent J. Della Pietra, Stephen A. Della Pietra and Robert L. Mercer, "Mathematics of Statistical Machine Translation: Parameter Estimation," Computational Linguistics, Vol. 19, No. 2, pp. 263-311, 1993.

[Chang 92] Chang, J.-S., Y.-F. Luo and K.-Y. Su, "GPSM: A Generalized Probabilistic Semantic Model for Ambiguity Resolution," *Proceedings of ACL-92*, pp. 177--184, 30th Annual Meeting of the Association for Computational Linguistics, University of Delaware, Newark, DE, USA, 28 June--2 July, 1992.

[Chang 93a] Chang, J.-S. and K.-Y. Su, "A Corpus-Based Statistics-Oriented Transfer and Generation Model for Machine Translation," *Proceedings of TMI-93*, pp. 3-14, 1993.

[Chang 93b] Chang Chao-Huang and Cheng-Der Chen, "Automatic Clustering of Chinese Characters and Words," *in Proceedings of ROCLING-V*I, pp. 57-78, ShiTao, Taiwan, R.O.C., 1993.

[Chen 91] Chen, S.-C., J.-S. Chang, J.-N. Wang and K.-Y. Su, "ArchTran: A Corpus-Based Statistics-Oriented English-Chinese Machine Translation System," *Proceedings of Machine Translation Summit III*, pp. 33--40, Washington, D.C., USA, July 1--4, 1991.

[Chen 92] Chen, Keh-Jiann and Shing-Huan Liu, "Word Identification for Mandarin Chinese Sentences", *Proceedings of COLING-92*, vol. I, pp. 101-107, 15th Int. Conference on Computational Linguistics, Nantes, France, July 23--28, 1992.

[Chen 95] Kuang-hua Chen and Hsin-Hsi Chen, "Machine Translation: an Integration Approach," *Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation*, TMI-95, pp. 287-294, Leuven, Belgium, July 5-7, 1995.

[Chiang 92a] Tung-Hui Chiang, Y.-C. Lin and K.-Y. Su, "Syntactic Ambiguity Resolution Using A Discrimination and Robustness Oriented Adaptive Learning Algorithm", *Proceedings of COLING-92*, vol. I, pp. 352--358, 14th Int. Conference on Computational Linguistics, Nantes, France, July 23--28, 1992.

[Chiang 92b] Tung-Hui Chiang, Jing-Shin Chang, Ming-Yu Lin and Keh-Yih Su, "Statistical Models for Word Segmentation and Unknown Word Resolution," *Proceedings of ROCLING-V*, pp. 123-146, Taipei, Taiwan, R.O.C., 1992.

[Chiang 95] Tung-Hui Chiang, Yi-Chung Lin and Keh-Yih Su, "Robust Learning, Smoothing, and Parameter Tying on Syntactic Ambiguity Resolution," *Computational Linguistics*, pp. 322-349, Vol 21, No. 3, 1995.

[Chiang 96] TungHui Chiang and Keh-Yih Su, "Statistical Models for Deep-structure Disambiguation,". to appear in *Proceedings of 4th Workshop on Very Large Corpora*, 1996.

[Church 88] Church, K., "A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text," *ACL Proc. 2nd Conf. on Applied Natural Language Processing*, pp. 136-143, Austin, Texas, USA, 9-12 Feb. 1988.

[Church 89] Church, K. and P. Hanks, "Word Association Norms, Mutual Information, and Lexicography," *Proc. 27th Annual Meeting of the ACL*, pp. 76-83, University of British Columbia, Vancouver, British Columbia, Canada, 26-29 June 1989.

[Cottrell 89] Cottrell, G. W., *A Connectionist Approach to Word Sense Disambiguation*, Morgan Kaufmann Publishers, 1989.

[Dempster 77] Dempster A. P., N. M. Laird and D. H. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society*, 39(B), 1--38, 1977.

[DeRose 88] DeRose, Steven. J., "Grammatical Category Disambiguation by Statistical Optimization," *Computational Linguistics*, vol. 14, no. 1, pp. 31-39, 1988.

[Devijver 82] Devijver, P.A., and J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice-Hall, London, 1982.

[Dagan 91] Dagan I. A. Itai, U. Schwall, "Two Language Are More Informative Than One," Proceedings of 29th Annual Meeting of the Association for Computational Linguistics, pp. 130-137, California, USA, June 18--21, 1991.

[Dagan 93] Dagan, Ido, Church, K. W, and Gale, W. A., "Robust Bilingual Word Alignment for Machine-Aided Translation," in Proceedings of Workshop on Very Large Corpora: Academic and Industrial Perspectives, " Columbus, Ohio, pp. 1-8, 1993.

[Dagan 94] Dagan, Ido and Church, K. W., "Termight: Identifying and Translating Technical Terminology," in Proceedings of Fourth Conference on Applied Natural Language Processing (ANLP-94), pp. 34-40, Stuttgart, Germany, 1994.

[Duda 73] Duda, O. R., P. E. Hart, Pattern Classification and Scene Analysis, Hohn Wiley and Sons, Inc., USA, 1973.

[Gale 91a] Gale, W. A., and K. W. Church, "A Program for Aligning Sentences in Bilingual Corpora," Proceedings of 29th Annual Meeting of the Association for Computational Linguistics, pp. 177-184, California, USA, June 18-21, 1991.

[Gale 91b] Gale, W. A., and K. W. Church, "Identifying Word Correspondences in Parallel Texts," Proceedings of DARPA Speech and Natural Language Workshop, pp. 152-157, Pacific Grove, California, USA, 1991.

[Gale 92] Gale, W. A., K. W. Church, and D. Yarowsky, "Using Bilingual Materials to Develop Word Sense Disambiguation Methods," Proceedings of TMI-92, pp. 101-112, 4th Int. Conf. on Theoretical and Methodological Issues in Machine Translation, Montreal, Canada, June 25--27, 1992.

[Garside 87] Garside, Roger, Geoffrey Leech and Geoffrey Sampson (eds.), The Computational Analysis of English: A Corpus-Based Approach, Longman Inc., New York, 1987.

[Golding 95] Golding, A. R. "A Bayesian Hybrid Method for Context-Sensitive Spelling Correction," in Proceedings of the third Workshop on Very Large Corpora, pp. 39-53, Boston, USA, 1995.

[Golding 96] Golding, A. R. and Y. Schabes, "Combining Trigram-based and Feature-based Methods for

Context-Sensitive Spelling Correction," in Proceedings of the 34th Annual Meeting of the Associated for Computational Linguistics, pp. 71-78, Santa Cruz, California, USA, 1996.

[Good 53] Good, I. J., "The population frequencies of species and the estimation of population parameters," Biometrika, vol.40, no. 3, 4, pp. 237--264, 1953.

[Hoel 71] Hoel, P. G., S. C. Port, C. J. Stone, *Introduction to Statistical Theory,* Houghtone Mifflin Company, USA, 1971.

[Hsu 95] Hsu, Yu-Ling Una and Keh-Yih Su, "The New Generation BehaviorTran: Design Philosophy and system architecture," *Proceedings of ROCLING IIX*, pp. 65-79, Aug. 1995.

[Hutchins 86] W.J. Hutchins, *Machine Translation: Past, Present, Future*, Ellis Horwood Limited, West Sussex, England, 1986.

[Katagiri 91] Katagiri, S., C. H. Lee, and B. H. Juang, "New Discriminative Training Algorithm Based on the Generalized Probabilistic Decent Method," Proceedings of 1991 IEEE Workshop Neural Networks for Signal Processing, pp. 299-308, Piscataway, NJ, Aug. 1991.

[Katz 87] Katz, S. M., "Estimation of Probabilities From Sparse Data for the Language Model Component of a Speech Recognizer," IEEE Transactions on Acoustics, Speech and Signal Processing, No. 35, pp. 400--401, 1987.

[Kupiec 92] Kupiec, J. "Robust part-of-speech Tagging Using a Hidden Markov Model," Computer Speech and Language Vol. 6, pp. 225-242, 1992.

[Kupiec 93] Kupiec, J., "An Algorithm for Finding Noun Phrase Correspondences in Bilingual Corpora," Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics, pp. 17-21, Ohio, USA, 22-26 June 1993.

[Lee 88] Lee, K. F., "Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System," PhD Dissertation, Computer Science Department, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. 1988.

[Lin 92] Lin, Yi-Chung, Tung-Hui Chiang and Keh-Yih Su, "Discrimination Oriented Probabilistic Tagging," *Proceedings of ROCLING V*, pp. 87-96, 1992.

[Lin 93] Lin, Ming-Yu, Tung-Hui Chiang and Keh-Yih Su, "A Preliminary Study on Unknown Word Problem in Chinese Word Segmentation," *Proceedings of ROCLING V*I, pp. 119-142, 1993.

[Lin 95] Lin, Yi-Chung, Tung-Hui Chiang and Keh-Yih Su, "The effects of Learning, Parameter Tying, and Model Refinement for Improving Probabilistic Tagging," Computer Speech and Language, pp. 37-61, Vol 9, 1992.

[Lin 96] Lin, Yi-Chung, *A Level Synchronous Approach to Ill-formed Sentence Parsing and Error Recovery*. PhD Dissertation, Department of Electrical Engineering, National Tsing-Hua University, Hsinchu, Taiwan 30043, ROC. July, 1995.

[Liu 90] Liu, C.-L., J.-S. Chang and K.-Y. Su, "The Semantic Score Approach to the Disambiguation of PP Attachment Problem," Proceedings of ROCLING-III, pp. 253-270, National Tsing-Hua Univ., Taipei, R.O.C., Sept. 21-23, 1990.

[Liu 93] Liu, Yuan-Ling, Shih-Ping Wang and Keh-Yih Su, "Corpus-based Automatic Rule Selection in Designing a Grammar Checker," *Proceedings of ROCLING V*I, pp. 161-171, Sept. 1993.

[Mays 91] Mays, E, F. J. Damerau, and R. L. Mercer, "Context based Spelling Correction," Information Processing and Management, Vol 27, No. 5., pp. 517-522, 1991.

[King 95] King, Maghi, "An Introduction to MT," Tutorial session, TMI-95, 1995.

[Merialdo 91] Merialdo, Bernard, "Tagging Text with a Probabilistic Model," in Proceedings of the IEEE 1991 International Conference on Acoustic, Speech, and Signal Processing, pp. 809-812, Toranto, 1991.

[Michalski 83] Michalski, R. S., J. G. Carbonell and T. M. Mitchell (Eds.), 1983. *Machine Learning: An Artificial Intelligence Approach, Vol. I,* Morgan Kaufmann Publishers, 1983.

[Michalski 86] Michalski, R. S., J. G. Carbonell and T. M. Mitchell (Eds.), 1986. *Machine Learning: An*

*Artificial Intelligence Approach, Vol. II,* Morgan Kaufmann Publishers, 1986.

[Ney 91] Ney, H. and U. Essen, "On smoothing Techniques for Bigram-based Natural Language Modelling, in Proceedings of the IEEE 1991 International Conference on Acoustic, Speech, and Signal Processing, pp. 251-258, Toranto, 1991.

[Papoulis 84] Papoulis A., *Probability, Random Variable, and Stochastic Processes*, 2nd Ed. McGraw-Hill, USA. 1984.

[Papoulis 90] Papoulis A., *Probability && Statistics*, Prentice Hall, NJ, USA. 1990.

[Rabiner 86] Rabiner, L. R., J. G. Wilpon, and B. H. Juang, "A Segmental k-Means Training Procedure for Connected Word Recognition," AT&T Tech. Journal, Vol. 65, No. 3, pp.21-31, May-June 1986.

[Rabiner 89] Rabiner, L. R., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proceeding of IEEE, pp.1404-1413, Vol. 77, No. 2, Feb. 1989.

[Rabiner 93] Rabiner, Lawrence and Biing-Hwang Juang, 1993. *Fundamentals of Speech Recognition*, Prentice Hall International, 1993.

[Salton 93] Gerard Salton and Michael J. McGill, 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, 1983.

[Smadja 91] Smadja F. A., "From N-Grams to Collocations: An Evaluation of Xtract," Proceedings of 29th Annual Meeting of the Association for Computational Linguistics, pp. 279-284, California, USA, June 18-21, 1991.

[Smadja 96] F. Smadja, K. R. McKeown, and V. Hatzivassiloglou, "Translating Collocations for Bilingual Lexicons: A Statistical Approach," *Computational Linguistics*, Vol 22, No. 1, 1996.

[Schalkoof 92] Schalkoof, R., "Pattern Recognition: statistical, structural and neural approaches," John Wiley & Sons, Inc. Singapore 1992.

[Su 88] Su, K.-Y. and J.-S. Chang, "Semantic and Syntactic Aspects of Score Function," Proc. of COLING-88, vol. 2, pp. 642--644, 12th Int. Conf. on Computational Linguistics, Budapest, Hungary, August 22--27, 1988.

[Su 89a] Su, K.-Y., J.-N., Wang, M.-H. Su and J.-S. Chang, "A Sequential Truncation Parsing Algorithm Based on the Score Function," Proceedings of International Workshop on Parsing Technologies (IWPT-89), pp. 95--104, CMU, Pittsburgh, PA, USA, August 28-31, 1989.

[Su 89b] Su, K.-Y., M.-H. Su and L.-M. Kuan., "Smoothing Statistic Databases in a Machine Translation System," Proceedings of ROCLING-II, pp. 333-347, Academia Sinica, Taipei, Taiwan, R.O.C., Sept. 22-24, 1989.

[Su 90] Su, K.-Y. and J.-S. Chang, "Some Key Issues in Designing MT Systems," Machine Translation, vol. 5, no. 4, pp. 265-300, 1990.

[Su 91a] Su, K.-Y. and C.-H. Lee, "Robustness and Discrimination Oriented Speech Recognition Using Weighted HMM and Subspace Projection Approach," Proceedings of IEEE ICASSP-91, vol. 1, pp. 541-544, Toronto, Ontario, Canada. May 14-17, 1991.

[Su 91b] Su, K.-Y., J.-N. Wang, M.-H. Su and J.-S. Chang, "GLR Parsing with Scoring," In M. Tomita (ed.), Generalized LR Parsing, Chapter 7, pp. 93-112, Kluwer Academic Publishers, 1991.

[Su 91c] Su, Keh-Yih, Yu-Ling Hsu and Claire Saillard, "Constructing A Phrase Structure Grammar By Incorporating Linguistic Knowledge And Statistical Log-Likelihood Ratio," *Proceedings of ROCLING IV*, pp. 257-275, National Chiao-Tung Univ., Taiwan, August 18-20, 1991.

[Su 92a] Su, Keh-Yih and Jing-Shin Chang, "Why Corpus-Based Statistics-Oriented Machine Translation," Proceedings of TMI-92, pp. 249-262, *Proceedings of 4th Int. Conf. on Theoretical and Methodological Issues in Machine Translatio*n, Montreal, Canada, June 25-27, 1992.

[Su 92b] Su, K.-Y., M.-W. Wu and J.-S. Chang, "A New Quantitative Quality Measure for Machine Translation Systems," Proceedings of COLING-92, vol. II, pp. 433--439, 14th Int. Conference on Computational Linguistics, Nantes, France, July 23--28, 1992.

[Su 94a] Su, Keh-Yih, Ming-Wen Wu and Jing-Shin Chang, "A Corpus-based Approach to Automatic Compound Extraction," *Proceedings of ACL-94*, pp. 242--247, 32nd Annual, USA, 27 June - 1 July, 1994.

[Su 94b] Su, Keh-Yih and Chin-Hui Lee, "Speech Recognition Using Weighted HMM and Subspace Projection Approaches," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no.1, pp.69-79, Jan. 1994.

[Su 94c] Su, Keh-Yih, Tung-Hui Chiang, and Jing-Shin Chang, "Introduction to Corpus-based Statistics-oriented (CBSO) Techniques," *Pre-Conference Workshop on Corpus-based NLP*, ROCLING VII, National Tsing-Hua Univ., Taiwan, ROC, Aug. 1994.

[Su 95] Su, K.-Y, J.-S. Chang, and Yu-Ling Una Hsu, "A Corpus-based Two-Way Design for Parameterized MT Systems: Rationale, Architecture and Training Issues," *Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation*, TMI-95, vol. 2, pp. 334-353, Sixth Int. Conf. on Theoretical and Methodological Issues in Machine Translation, Leuven, Belgium, July 5--7, 1995.

[Tung 94] Tung, Cheng-Huang and Hsi-Jian Lee, "Identification of Unknown Words from Corpus," *Computer Processing of Chinese & Oriental Languages*, Vol. 8, pp. 131-145, Dec. 1994.

[Wright 91] Wright, J. H. and Wrigley, E. N., "GLR Parsing with Probability," In M. Tomita (ed.), Generalized LR Parsing, Chap. 8, pp.123-128, Kluwer Academic Publishers, 1991.

[Wu 94] Wu, Dekai, "Aligning A Parallel English-Chinese Corpus Statistically With Lexical Criteria," *Proceedings of ACL-3*2, pp. 80-87, New Mexico, USA, June 27-30, 1994.

[Wu 95] Wu, Dekai, "Grammarless Extraction of Phrasal Translation Examples from Parallel Texts," *Proceedings of TMI-95*, pp. 354-371, Leuven, Belgium, July 5-7, 1995.

[Yarowsky 92] Yarowsky D., "Word-Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora" *Proceedings of COLING-92*, vol. I, pp. 454-460, 14th Int. Conference on Computational Linguistics, Nantes, France, July 23--28, 1992.

[Yarowsky 94] Yarowsky D. "Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French," in Proceedings of the 32th Annual Meeting of the Association for Computational Linguistics, pp. 88-95, Las Cruces, USA.

[Yarowsky 95] Yarowsky D. "Unsupervised Word Sense Disambiguation Rivaling Supervised Methods," in Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, pp. 189-196, MIT, MA, USA, June 1995.

[張 94] 張元貞，林頌堅，簡立峰，陳克健，李琳山，「國語語音辨認中詞群語言模型之分群方法與應用」，中華民國八十三年第七屆計算語言學研討會論文集， 17 - 34頁。