**RANDOMIZING OUTPUTS TO INCREASE PREDICTION ACCURACY**

Bagging and boosting reduce error by changing both the inputs and outputs to form perturbed training sets, grow predictors on these perturbed training sets and combine them. A question that has been frequently asked is whether it is possible to get comparable performance by perturbing the outputs alone. Two methods of randomizing outputs are experimented with. One is called output smearing and the other output flipping. Both are shown to consistently do better than bagging.In recent research in combining predictors, it has been recognized that the critical thing to success in combining low-bias predictors such as trees and neural nets has been through methods that reduce the variability in the predictor due to training set variability. Assume that the training set consists of N independent draws from the same underlying distribution.Conceptually, training sets of size N can be drawn repeatedly and the same algorithm used to construct a predictor on each training set. These predictors will vary, and the extent of the variability is a dominant factor in the generalization prediction error. Given a training set {(*yn* ,**x***n* ),*n*=1,...*N*} where the y's are either class labels or numerical values, the most common way of reducing variability is by perturbing the training set to produce alternative training sets, growing a predictor on each perturbed training set, and either averaging the outcomes (regression) or letting them vote for the most popular class (classification). Examples are bagging (Breiman[1996b]) and arcing-boosting However, in work (Breiman[1996b]) on subset selection in linear regression, I found the somewhat surprising result that adding noise to the y's while leaving the input **x**-vectors unchanged worked just as well as bagging. This paper extends those earlier results to non-linear contexts in both regression and classification and also tries to understand why perturbing outputs only works well. Although we use trees in the experimental results, the ideas have more general validity.

**Comparative Analysis of Viterbi Training and Maximum Likelihood Estimation for HMMs**

We present an asymptotic analysis of Viterbi Training (VT) and contrast it with a more conventionalMaximum Likelihood (ML) approach to parameter estimation in Hidden Markov Models. While ML estimator works by (locally) maximizing the likelihood of the observed data, VT seeks to maximize the probability of the most likely hidden state sequence. We develop an analytical framework based on a generating function formalism and illustrate it on an exactly solvable model of HMM with one unambiguous symbol. For this particular model the ML objective function is continuously degenerate. VT objective, in contrast, is shown to have only finite degeneracy. Furthermore, VT converges faster and results in sparser (simpler) models, thus realizing an automatic Occam’s razor for HMM learning. For more general scenario VT can be worse compared to ML but still capable of correctly recovering most of the parameters.Hidden Markov Models (HMM) provide one of the simplest examples of structured data observed through a noisy channel. The inference problems of HMM naturally divide into two classes [20, 9]:

*i)* recovering the hidden sequence of states given the observed sequence, and *ii)* estimating themodel parameters (transition probabilities of the hidden Markov chain and/or conditional probabilities of observations) from the observed sequence. The first class of problems is usually solved via the maximum a posteriori (MAP) method and its computational implementation known as Viterbi algorithm [20, 9]. For the parameter estimation problem, the prevailing method is maximum likelihood (ML) estimation, which finds the parameters by maximizing the likelihood of the observed data. Since global optimization is generally intractable, in practice it is implemented through an expectation–maximization (EM) procedure known as Baum–Welch algorithm [20, 9].An alternative approach to parameter learning is Viterbi Training (VT), also known in the literature as segmental K-means, Baum–Viterbi algorithm, classification EM, hard EM, etc. Instead of maximizing the likelihood of the observed data, VT seeks to maximize the probability of the most likely hidden state sequence. Maximizing VT objective function is hard [8], so in practice it is implemented via an EM-style iterations between calculating the MAP sequence and adjusting the model parameters based on the sequence statistics. It is known that VT lacks some of the desired features of ML estimation such as consistency [17], and in fact, can produce biased estimates [9]. However, it has been shown to perform well in practice, which explains its widespread use in applications such as speech recognition [16], unsupervised dependency parsing [24], grammar induction [6], ion channel modeling [19]. It is generally assumed that VT is more robust and faster but usually less accurate, although for certain tasks it outperforms conventional EM [24]. The current understanding of when and under what circumstances one method should be preferred over the other is not well–established. For HMMs with continuos observations, Ref. [18] established 1 an upper bound on the difference between theML and VT objective functions, and showed that both approaches produce asymptotically similar estimates when the dimensionality of the observation space is very large. Note, however, that this asymptotic limit is not very interesting as it makes the structure imposed by the Markovian process irrelevant. A similar attempt to compare both approaches on discrete models (for stochastic context free grammars) was presented in [23]. However, the established bound was very loose.

**A Nonparametric Method for Extraction of Candidate Phrasal Terms**

This paper introduces a new method for identifying candidate phrasal terms (also known as multiword units) which applies a nonparametric, rank-based heuristic measure. Evaluation of this measure, the *mutual rank ratio* metric, shows that it produces better results than standard statistical measures when applied to this task. The ordinary vocabulary of a language like English contains thousands of *phrasal terms* -- multiword lexical units including compound nouns, technical terms, idioms, and fixed collocations. The exact number of phrasal terms is difficult to determine, as new ones are coined regularly, and it is sometimes difficult to determine whether a phrase is a fixed term or a regular, compositional expression. Accurate identification of phrasal terms is important in a variety of contexts, including natural language parsing, question answering systems, information retrieval systems, among others.Insofar as phrasal terms function as lexical units, their component words tend to cooccur more often, to resist substitution or paraphrase, to follow fixed syntactic patterns, and to display some degree of semantic noncompositionality (Manning, 1999:183-186). However, none of these characteristics are amenable to a simple algorithmic interpretation. It is true that various term extraction systems have been developed, such as Xtract (Smadja 1993), Termight (Dagan & Church 1994), and TERMS (Justeson & Katz 1995) among others (cf. Daille 1996, Jacquemin & Tzoukermann 1994, Jacquemin, Klavans, & Toukermann 1997, Boguraev & Kennedy 1999, Lin 2001). Such systems typically rely on a combination of linguistic knowledge and statistical association measures. Grammatical patterns, such as adjective-noun or noun-noun sequences are selected then ranked statistically, and the resulting ranked list is either used directly or submitted for manual filtering.Generating Typed Dependency Parses from Phrase Structure Parses This paper describes a system for extracting typed dependency parses of English sentences from phrase structure parses. In order to capture inherent relations occurring in corpus texts that can be critical in real-world applications, many NP relations are included in the set of grammatical relations used. We provide a comparison of our system with Minipar and the Link parser. The typed dependency extraction facility described here is integrated in the Stanford Parser, available for download. We describe a system for automatically extracting typed dependency parses of English sentences from phrase structure parses. Typed dependencies and phrase structures are different ways of representing the structure of sentences: while a phrase structure parse represents nesting of multi-word constituents, a dependency parse represents dependencies between individual words. A typed dependency parse additionally labels dependencies with grammatical relations, such as subject or indirect object. There has been much linguistic discussion of the two formalisms. There are formal isomorphisms between certain structures, such as between dependency grammars and one bar-level, headed phrase structure grammars (Miller, 2000). In more complex theories there is significant debate: dominant Chomskyan theories (Chomsky, 1981) have defined grammatical relations as configurations at phrase structure, while other theories such as Lexical-Functional Grammar has rejected the adequacy of such an approach (Bresnan, 2001). Our goals here are more practical, though in essence we are following an approach where structural configurations are used to define grammatical roles.

**Extracting Noun Phrases from Large-Scale Texts: A Hybrid Approach and Its Automatic Evaluation**

To acquire noun phrases from running texts is useful for many applications, such as word grouping, terminology indexing, *etc.* The reported literatures adopt pure probabilistic approach, or pure rule-based noun phrases grammar to tackle this problem. In this paper, we apply a probabilistic chunker to deciding the implicit boundaries of constituents and utilize the linguistic knowledge to extract the noun phrases by a finite state mechanism. The test texts are SUSANNE Corpus and the results are evaluated by comparing the parse field of SUSANNE Corpus automatically. The results of this preliminary experiment are encouraging.From the cognitive point of view, human being must recognize, learn and understand the entities or concepts(concrete or abstract) in the texts for natural language comprehension. These entities or concepts are usually described by noun phrases. The evidences from the language learning of children also show the belief (Snowand Ferguson, 1977). Therefore, if we can grasp the noun phases of the texts, we will understand the texts to some extent. This consideration is also captured by theories of discourse analysis, such as Discourse Representation Theory (Kamp, 1981).

**Noun-Phrase Analysis in Unrestricted Text for Information Retrieval:**

Information retrieval is an important application area of natural-language processing where one encounters the genuine challenge of processing large quantities of unrestricted natural-language text. This paper reports on the application of a few simple, yet robust and efficient nounphrase analysis techniques to create better indexing phrases for information retrieval.

In particular, we describe a hybrid approach to the extraction of meaningful (continuous or discontinuous) subcompounds from complex noun phrases using both corpus statistics and linguistic heuristics. Results of experiments show that indexing based on such extracted subcompounds improves both recall and precision in an information retrieval system. The noun-phrase analysis techniques are also potentially useful for book indexing and automatic thesaurus extraction. Information retrieval (IR) is an important application area of naturaManguage processing (NLP). 1 The IR (or perhaps more accurately "text retrieval") task may be characterized as the problem of selecting a subset of documents (from a document collection) whose content is relevant to the information need of a user as expressed by a query. The document collections involved in IR are often gigabytes of unrestricted natural-language text. A user's query may be expressed in a controlled language (e.g., a boolean expression of keywords) or, more desirably, a natural language, such as English.A typical IR system works as follows. The documentsto be retrieved are processed to extract *indexing terms* or *content carriers,* which are usually single words or (less typically) phrases. The indexing terms provide a description of the document's content. Weights are often assigned to terms to indicate how well they describe the document. A (natural-language) query is processed in a similar way to extract *query terms.* Query terms are then matched against the indexing terms of a document to determine the relevance of each document to the quer3a