Language-independent Induction of Part of Speech Class Labels Using Only Language Universals

Patrick Schone and Daniel Jurafsky
University of Colorado, Boulder CO 80309
{schone, jurafsky}@cs.colorado.edu

Abstract

We introduce a language-independent strategy for inducing part of speech tags from corpora. Unlike other techniques that use language-specific lexicons, rulesets, and so forth to tag, our algorithm bootstraps only from cluster properties and language universals. We describe the theory and illustrate an introductory experiment which verifies the feasibility of this near-knowledge-free tag induction strategy. We induce tags for perfect syntactic clusters generated from the Brown corpus getting 77% of our hypotheses correct.

1 Introduction

Knowledge of part of speech (POS) categories for words in text corpora is an important ingredient for various tasks in computational linguistics. In the past, there have been five principal methods for labeling syntactic categories:

(1) tag words by hand
(2) apply hand-built or learned rules/statistics
(3) stochastically tag using a lexicon
(4) infer tags using parallel corpora and tags from another language
(5) automatically induce syntactic clusters.

Each approach has strengths and weaknesses. When approaching a new domain, the first, using human-provided labels, is sometimes needed due to lack of linguistic resources. Yet the expense and tedium of hand-tagging a corpus for each new language makes this option undesirable.

The second strategy is more powerful than the first: once rules are determined, the tagger can readily provide POSs for multiple corpora in the same domain without any additional effort. However, using hand-built rules [Harris, 1962; Klein and Simmons, 1963] can be brittle. It is possible to use a pretagged corpus or lexicon to learn robust rules or statistical properties for tagging [Brill, 1995; Voutilainen, 1995; Samuelsson and Voutilainen, 1997], but the need for an initial expert when moving across languages cannot be escaped.

Electronic dictionaries containing POS tags are becoming more prevalent, so the third approach [Bahl and Mercer, 1976; Jelinek, 1985; Church, 1989; Merialdo, 1991; Kupiec, 1992; etc.] is widely applicable. Similarly, since several hand-tagged corpora exist (such as the Brown Corpus [Francis and Kučera, 1982] in English) and since parallel corpora are increasing in availability, the fourth approach [Yarowsky, *et al.*, 2001] is also very powerful. Yet, there remain languages where lexicons and parallel corpora do not exist. In such cases, neither approach is applicable.

The last approach is appealing in that one might expect that the induction of syntactic clusters [Brown, *et al.*, 1992; Chater and Finch, 1992; Schütze, 1993; Pereira, *et al.*, 1993, Niesler, *et al.*, 1998; Clark, 2000] is data and language-independent and is certainly inexpensive in term of human expert labor. Yet having syntactic clusters without corresponding POS tags has limited applicability. Many algorithms which make use of POS need knowledge of actual syntactic classes of words instead of the arbitrary clusters to which they belong.

A highly desirable solution would be a knowledge-free algorithm that automatically induces appropriate POS clusters *and* tags in many languages and requires little or no human input. To our knowledge, such an algorithm has never been proposed. We describe an approach for tackling a subset of this problem. We illustrate an algorithm which, given syntactic clusters, automatically determines the corresponding POS tags for those clusters. Moreover, our algorithm requires *no* language-dependent human input or resource. The only human-provided information available to our algorithm is a list of existing language universal rules [Greenberg, 1966; Comrie, 1989; Croft, 1991; Plank and Filiminova, 1996] and a rudimentary knowledge of word spacing and punctuation for tokenizing. Using existing universals has the advantage that no additional human effort is needed to port the tagging algorithm across languages.

Our algorithm makes decisions by taking advantage of join-tree-based Bayesian Networks [Jensen, 1996], induced morphology [Schone and Jurafsky, 2000], and automatically-extracted features. Furthermore, it reaches its conclusions without explicit knowledge of the POS tags available in the language of application. Though we expect this approach to be applicable to multiple languages, we illustrate preliminary results of its performance on the Brown corpus in English.

2 Data Precursors

Before describing the induction process, it is first useful to illustrate the data needed to automatically induce POS tags. Essentially we need three pieces of data. We first assume we have a reasonably-sized text corpus. Secondly, we need a set of syntactic clusters generated from that corpus. It has been shown that using text corpora, the distributional properties of words make it possible to cluster words according to their syntactic functions. These clusters provide statistical clues which signal the likelihood of their being one POS more than another. The final ingredient needed is a collection of syntactic language universals, i.e., syntactic properties that hold true across languages. Such universals exist and identify the expected types of classes in general languages, potential orderings between classes, and "symbiotic" classes which tend to bind to one another. In this section, we will discuss each of these pieces of information with particular emphasis on the features derivable from the clusters and from the language universals since these are key ingredients of the inductive process.

2.1 Data and syntactic clusters

As mentioned before, there have been numerous efforts to induce syntactic clusters. We could choose any of these techniques or develop a separate one. However, to avoid confounding issues between clustering errors and tagging errors, we assume here that we are given the output of some "perfect" clustering algorithm. We can obtain perfect clusters by collecting together all the nouns, verbs, etc. from a truth-marked corpus and then throwing away the initial POS tags. Having truth markings also allows us to accurately assess performance upon completion of the induction stage.

We choose the Brown Corpus [Francis and Kučera, 1982] for our induction corpus. This corpus contains about one million words. It also serves as an excellent dataset for assembling "perfect clusters" since every word has been labeled by hand with a POS [Marcus, *et al.*, 1993]. The corpus has 48 POS tags, but we consolidate these by grouping punctuation into single clusters and grouping subsets of larger classes (such as collapsing VBG, the gerund form of verbs, into the more general class for verbs, VB). This consolidation leaves 20 general classes: determiners, proper nouns, common nouns, verbs, prepositions, genitives, adjectives, punctuations, adverbs, question terms, conjunctions, infinitive markers, pronouns, numbers, modal verbs, pronominal determiners, existentials, list indices, interjections, and foreign words. A given word *W* can be a member of several classes, so we provide our algorithm a list, $\{ c_1, p_1, c_2, p_2, ...\}$ where c_i is the *i*th cluster and p_i is the probability that *W* occurs in cluster *i*.

2.2 Cluster Properties

We extract numerous features from the clusters, but we will here only make mention of those that we have currently employed in the induction process. In particular, we identify openness, affixation, optionality, numeracy, cluster order, and bound classes. The following sections describe each feature as well as how we obtain and employ them in the induction.

Openness

A POS class is called *open* if the number of unique words contained within it will continue to grow as the corpus size increases toward infinity. All other finite classes are *closed*. Nouns and verbs exist in most every language, and since they are the principal conveyors of information, one would expect that for most languages, these classes will be open. Although adjectives typically exist across languages, there are languages which only allow for a limited number of adjectival types (such as shape and color). Adverbs do not have to exist at all. Nonetheless, if *any* classes are open, one would expect these four to be open, as well as numbers and, perhaps, interjections and foreign words. All other classes are closed.

Identification of openness versus closedness turns out to be a rather simple problem. If we recognize that closed classes are finite, we would expect that if we start at the beginning of a corpus looking for all words in that class, we will quickly reach a point where we have seen practically all of its class elements. If we consider the average frequency count of words in that set and divide by the cardinality (size) of the set as it grows over time, we expect that after some point, that ratio will simply increase monotonically. On the other hand, with open classes, the cardinality will continue to grow as the frequency counts grow, so one might almost expect a fairly constant ratio through time.

From every experiment we have conducted, these notions seem to be true. We process our corpus in blocks of 5000 words at a time. For the *n*th block (out of *N*), we compute the cumulative frequency count for the cluster divided by *n* times the cardinality of the set (i.e., for exaggeration effects, we actually use the largest cardinality the cluster will ever have in the whole dataset). When all blocks have been processed, we can compute the mean and standard deviations of the size ratios as they progressed through time. We use as our final openness score, *N* times the largest cardinality divided by the standard deviation. This measure strongly separates open classes from closed classes.

Affixation

Schone and Jurafsky [2000] illustrated a knowledge-free approach for inducing morphology in inflectional languages. A goal of this approach was to identify, for any word X, the set of words that conflate with X. For example, if X were "talk," one might expect the conflation set to include the words "talk," "talks," "talking," and "talked." These conflation sets give information about inflections within a language which can be useful for inducing appropriate POS tags. For example, if we consider the conflation set of talk as well as those of "climb" and "jump", namely, {climb, climbs, climbing, climbed} and {jump, jumps, jumping, jumped}, we would note a pattern {X, X+s, X+ing, X+ed} forming. Suppose now that "climb," "jump," and "talk" all fall into cluster C_1, and "climbs," "jumps," and "talks" fall into cluster C_2. Then we could view the suffixing operation "+s" as a mapping from a word in C_1 to a word in C_2. If the number of possible X's in C_1 that map to another word in C_2 exceeds some threshold, we could say that clusters C_1 and C_2 have strong evidence of affixation.

We could record the types of affixes present in each cluster so as to identify multiple clusters having the same POS. Yet we compute a measure of how "affixy" a cluster is: the percentage of words in any given cluster that have above-threshold affixing maps to words in other classes. By directly applying the Schone/Jurafsky morphology-induction to our corpus, we find that 21.8% of words from the verb cluster have above-threshold maps to either the same or to other classes. Of nouns and adjectives, there are 10.7% and 1.0%, respectively. All others have less than 1% (and typically 0%).

Language universals reveal that across languages, if a noun has inflections, so do verbs and pronouns. If adjectives or numbers have inflections, the nouns will typically carry them as well. Therefore, we can use affixation as a means of constraining the number of possibilities in induction.

Numeracy

The Schone/Jurafsky technique can also be used to find numbers. Consider the numbers 17, 19, 37, and 39. In a way, these can be thought of as root forms of 1 and 3 with possible "suffixes" 7 and 9; or they could be viewed as roots 7 and 9 with "prefixes" 1 and 3. Numbers also tend to be used in similar contexts. For these reasons, they frequently appear in each other's conflation sets. Moreover the characters that numbers are composed of have a nearly mutually exclusive usage from other, non-numeric words. These facts set numbers apart from other words. A class containing a majority of words whose sole character content comes from these induced numerals is highly likely to be numbers.

Optionality

In many languages, adjectives and adverbs act as modifiers rather than heads, and hence are syntactically optional. Even in languages like Chinese in which predicative adjectives are not syntactically optional, attributive adjectives are optional. In other words, in such languages, a sentence which is formed by deleting an adjective is still grammatical. To illustrate in English, "the big dog" or "the dog" are both grammatical noun phrases. This suggests we could identify all situations in the text where there is a word X followed by words Y and Z such that removing Y and leaving only X Z still provides another valid word pair from somewhere else in the text. If Z, for example, were a noun, and if Y could be removed from, say, 10 word triples in the corpus and if the number of valid removals constituted 15% or more of Y's frequency, then we might be inclined to believe that Y is an adjective. Moreover, we may believe that this language has adjectives that occur before their head noun. If X were the noun, we might have reason to believe that adjectives appear after their head noun.

We actually count the number of deletable, Y-type words that show up in connection with each of the open classes. For illustration, Table 1 indicates the percentage of words with frequencies of at least 10 from several clusters that could be deleted directly before (B) or after (A) each of the major open classes. As expected, adjectives strongly show optionality and occur typically before nouns in English. Adverbs also show up as optional for both verbs and adjectives in English, but as expected, their order is less restricted (though they tend to *follow* their heads more often than preceding them).

	Side	Noun	Verb	Prop Noun	Adj	Adverb
Adj	B	**21.7%**	0	0.2%	0	0
Adj	A	0	0	0	0	0
Adv	B	0.7%	**6.2%**	0	1.6%	0
Adv	A	1.8%	**8.2%**	0	0	0
Det	B	0	0	0	0	0
Det	A	0	0	0	0	0

Table 1 Percentage of words from several clusters (row indices) with frequencies at least 10 and which are deletable when preceding or following one of the open class clusters shown as column indices.

Cluster Order

We next compute statistics on cluster order. For every pair of clusters, X and Y, we compute the number of times X precedes Y either by one or two words, and vice versa. These frequencies are used in connection with expected order conveyed by language universals (as will be discussed later).

Bindings

Class X is *bound* to class Y if X cannot exist without Y. Consider determiners. One would rarely expect determiners in text without finding corresponding nouns nearby. In a certain sense, determiners owe their existence to nouns.

Let P(Y) represent the maximum likelihood estimate of class Y's probability, and let P(Y|X) indicate the maximum conditional probability of finding class Y either before or after class X. Using these assignments, we can obtain a probability of X's (e.g., determiners) being bound to class Y (e.g., nouns):

$$\frac{P(Y|X) - P(Y)}{1 - P(Y)} .$$

The numerator indicates the degree of surprise, if any, of finding classes X and Y juxtaposed as frequently as they are, and the denominator normalizes the formula to a maximum of 1.0. If the numerator is less than zero, it is set to zero.

2.3 Use of language universals

In the 1960s, Joseph Greenberg ran a well-known project with the goal of tabulating many language universals. He analyzed

30 diverse languages and proposed 45 rules similar to the following [Greenberg,1966, p.110]:

[Rule] 3. Languages with dominant VSO order are always prepositional .

One could interpret such a rule to mean "if the language is verb-subject-object, then adpositions tend to always *precede* their noun phrase. For rules that are not as strongly universal, Greenberg attached such phrases as "with far better than chance" or "almost always." One could interpret these phrases as rough estimates of the rule's *a priori* probability of existence across languages.

These notions suggest it ought to be feasible to take Greenberg-like universals and convert them into probabilistic firing rules. For example, if the rule is

"**if** W *precedes* X **and** S *precedes* T,
then Y *precedes* Z [most of the time],"

one can assign some arbitrary likelihood to the phrase "most of the time" (say, 70%) to produce the following rule:

"{W X} & {S T} =>{Y Z} (p=70%)." (A)

where braces represent precedence order. If a sufficient number of universals existed, it would be feasible to create a part-of-speech labeler which bootstraps only from language universals...i.e., using no language-dependent resources.

Fortunately, significant progress has been made in identifying syntactic universals since Greenberg's seminal paper. The Universität Konstanz has established project Sprachbaupläne [Plank and Filiminova, 1996], devoted to assembling a compendium of such universals. Their list consists of over 1600 language universals of which several hundred are applicable to syntax. Our first step is to select and tabularize these syntactic rules. With the exception of rudimentary word boundary information, this is the <u>only</u> direct human-provided data our algorithm received.

2.4 Manipulating Rules for Ordering: Bayes Nets

A typical approach to dealing with probabilistic rules is to use Bayesian networks. Briefly speaking, Bayesian networks are probabilistic networks designed to compute the probabilities of all the desired parameters of a given model, making only independence assumptions when such are specified by the model. The underlying model of a Bayesian network is a directed acyclic graph, so the networks will only function properly when the parameters have non-cyclical behavior (i.e., if parameter *a* implies something about *b*, *b* cannot imply something else about some other parameter which affects *a*).

Pearl's [1988] approach to building Bayesian networks has a fast method for updating probabilities using a simple message-passing scheme. Yet since nodes in our graph can have multiple parents, Pearl's method will not work. Hence, we use Jensen's [1996] *join tree* approach. The interested reader should consult the original reference for significant detail, but we will provide a short description at this time.

Suppose we wanted to represent previously-mentioned rule (A) in terms of a Bayesian network. We might want the symbol "{W X}&{S T}" to form an entry in the network, as well as "{Y Z}." These become nodes in our DAG which we will say are binary (either true or false). If {W X}&{S T} is true, it implies not only that {Y Z} is 70% likely to be true, but also that two other new nodes, "{W X}" and "{S T}," are *definitely* true. These, in turn, imply that the individual components, W, X, S, and T are themselves nodes that are true. Suppose that in addition to rule A, another rule, B, says

"{W T}=>{Y Z} (p=65%)" (B)

This means that all components could be tied together to create the DAG depicted in Figure 1.

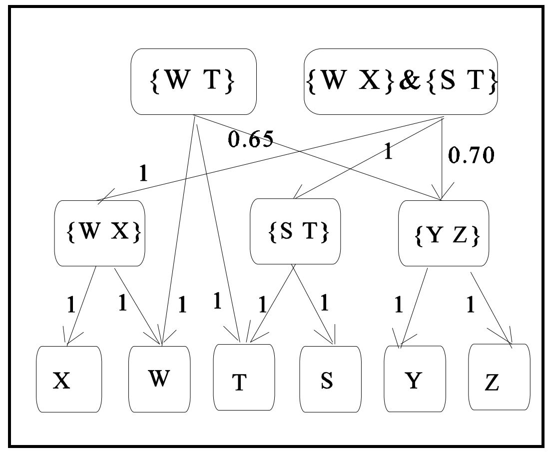

Figure 1: Elementary Bayesian network

In the figure, the weights on each edge represent the probability that the child node will be true if the parent node is true. These probabilities are derivable from the language universals using the methodology mentioned before. Yet we also need to somehow assign unconditioned probabilities for nodes, probabilities of child nodes when their parents are false, and probabilities for child nodes when they have multiple parents. For instance, what is the probability of the node {Y Z} if both its parents are true? The actual probability can be shown to fall withing a range of possibilities, but no further conclusions can be drawn. This gives us some freedom to make assignments.

Therefore, we create an *ad hoc* formulation based on the "noisy -or" computation. The noisy-or strategy for computing $p(x|y_{1,n})$ given $p(x|y_i)$ for each *i*, (where $y_{1,k}$ indicates the sequence $y_1,y_2,...y_k$) would be obtained recursively by:

$$p(x|y_{1,i})=p(x|y_{1,i-1}) + p(x|y_i)- p(x|y_{1,i-1})p(x|y_i)$$

The difficulty with using this formulation is that it assumes that whenever a conditional probability is non-zero, it is actually stating something positive about the conditioned variable. Yet if we have binary variables, then without any information, we may generally expect $p(x|y_i)$ is 0.5, regardless of x and y_i. However, if we have randomly claimed that $p(x|y_1)=0.5$ and $p(x|y_2)=0.5$, we likely expect $p(x|y_1,y_2)=0.5$ as opposed to the 0.75 which the noisy-or condition would suggest. Therefore, we define $p(x|y_{1,n})$ to be $\lambda_p + \lambda_n - 0.5$, where λ_p and λ_n are initialized as 0.5, and are computed recursively according to

$\lambda_p=2(\lambda_p+(1-\lambda_p)p(x|y_i))-1 \quad \forall p(x|y_i) \geq 0.5$, and
$\lambda_n=2\lambda_n\, p(x|y_i) \quad \forall p(x|y_i) < 0.5$.

These equations are similar to the noisy-or condition. Yet they separately handle favorable (p>0.5) and unfavorable (p<0.5) conditional probabilities. One might expect that if most conditional probabilities are unfavorable, the multi-conditioned probabilities should tend toward zero. Likewise, if most conditional probabilities are favorable, the overall probability should tend toward 1.0. It can be readily determined that these equations do have this kind of behavior. Furthermore, the probability will indeed be 0.5 if, in the long run, all conditional probabilities had been 0.5. There are additional steps to the implementation, but the interested reader should consult Jensen [1996] for details.

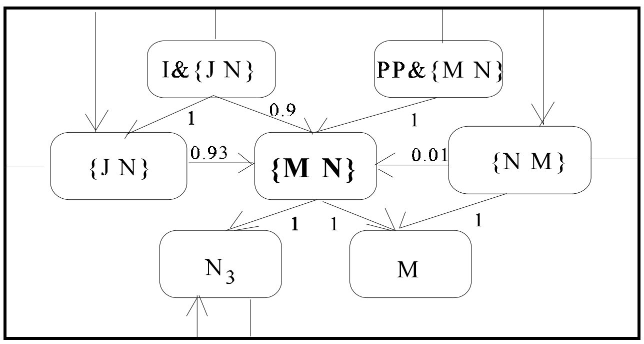

Figure 2 Subsection of net involving numbers (M) preceding nouns (N). Adjectives (J) and pre/postpositions (I/PP) also play a role.

A subsection of our final network (which only handles SVO languages for now) is shown in Figure 2. The full network has 102 nodes. The time to build the network depends on the size of the largest *clique*, which, simply put, is the maximum number of dependent nodes required for the message-passing scheme. One can reduce this number by splitting nodes (for example, N_3 is a split node of nouns, N). For our SVO network, the maximum clique has 16 nodes, so that clique's probability table has 2^{16} probability entries. This size limits the frequency with which the network can be updated.

For the reader's sake, we provide the following probability table (Table 2), which illustrates the probabilities of {M N} given different conditions of its parent nodes. From this table, we can find that the most likely condition for {M N} and its parent nodes is for {J N} to be true and {M N} to also be true with the rest of the nodes being false. This has a probability of

	---	1--	-2-	12-	--3	1-3	-23	123
¬MN,¬NM	.163	.013	.029	.001	.013	.013	.003	~0
¬MN, NM	.120	.056	.016	.034	.011	.011	.002	.004
MN,¬NM	.054	.168	.025	.122	.007	.007	.003	.015
MN, NM	~0	.044	.014	.034	.006	.006	.002	.004

Table 2: Probabilities of {M N} given conditions of its parents. The 1, 2, 3 represents respectively indicate that {J N}, I&{J N}, and PP&{M N} are true. Dashes mean that nodes are false.

0.168. By marginalizing (summing across the top two and bottom two rows), we can see that the probability of {M N} is 0.511 and the probability of ¬{M N} is 0.489. If our algorithm proposed the ordering {M N} as a potential ordering, this suggests we would tend to favor that ordering.

3 The Induction Process

In the previous section, we described the architecture and components needed for the induction process. However, we have not described how the induction process itself takes place. In this section, we describe the induction mathematics.

3.1. General induction strategy

Before describing strategy, we must define several symbols:

K_i	*i*th cluster
$\mathbf{\Phi}_i$	feature vector for K_i
T_i	POS tag assigned to K_i
$f_m(\mathbf{\Phi}_i)$	the *m*th feature of $\mathbf{\Phi}_i$
$\{\ \}_{i=1}^{m}$	a sequence of *m* items

Given these symbols, we can say that the task of finding the best POS tags for our set of clusters can be expressed as:

$$\underset{T_1,T_2,\ldots,T_N}{argmax}\; Pr(\{\mathbf{\Phi}_i, T_i\}_{i=1}^{N} \mid U) \qquad (1)$$

where U are the available language universals. We can expand on this equation to give us more focus on the quantities we need to compute. For simplicity, we assume that the probabilities are always conditioned on U. As typical, our first expansion of (1) is two applications of the chain rule:

$$\underset{\forall T_i}{argmax} \prod_{i=1}^{N} Pr(\mathbf{\Phi}_i, T_i \mid \{\mathbf{\Phi}_r, T_r\}_{r=1}^{i-1}) =$$

$$\underset{\forall T_i}{argmax} \prod_{i=1}^{N} Pr(T_i \mid \{\mathbf{\Phi}_r, T_r\}_{r=1}^{i-1})\, Pr(\mathbf{\Phi}_i \mid T_i, \{\mathbf{\Phi}_r, T_r\}_{r=1}^{i-1})$$

Since our clusters are unordered, we can process them in whatever order we prefer. We choose to process them in dependency order. By this we mean that we can first look for nouns and verbs, and then we can look for items that are dependent only upon them, and so forth. This means that we can effectively remove dependence of T_i on feature vectors for already-defined clusters. If we define a temporary history function $\boldsymbol{H}(i)=\{\mathbf{\Phi}_r, T_r\}_{r=1}^{i}$, we can simplify to

$$\underset{\forall T_i}{argmax} \prod_{i=1}^{N} Pr(T_i \mid \{T_r\}_{r=1}^{i-1})\, Pr(\mathbf{\Phi}_i \mid T_i, \boldsymbol{H}(i-1)). \quad (4)$$

The second component of (4) can be broken down by the chain rule based on the features of $\mathbf{\Phi}$. If we assume the features are mutually independent, the chain rule gives us

$$\underset{\forall T_i}{argmax} \prod_{i=1}^{N} Pr(T_i \mid \{T_r\}_{r=1}^{i-1}) \prod_{m=1}^{M} Pr(f_m(\mathbf{\Phi}_i) \mid T_i, \boldsymbol{H}(i-1)).$$

If we consider the first component of the product in (4), we could note that it depends on two pieces of information: (1) the *a priori* probability of the *class* in the language A_i and (2) the probability that the tag exists given the constraints placed on it by the universal rules and the distributional properties of the clusters. This yields:

$$Pr(T_i|\{T_r\}_{r=1}^{i-1}) = A_i * Pr(\hat{T}_i|\{T_r\}_{r=1}^{i-1})$$

The *A* function is unknown. It could potentially be estimated. However, since we have assumed that only one cluster can have a given tag, the fact that we are seeking argmax effectively eliminates the need to know *A*. The second component can be obtained directly from the ordering universals stored in the junction tree, JT. This means we have

$$\underset{\forall T_i}{argmax} \prod_{i=1}^{N} Pr(\hat{T}_i|JT_i) \prod_{m=1}^{M} Pr(f_m(\mathbf{\Phi}_i)\,|\,T_i, \boldsymbol{H}(i-1)).$$

The last product is sufficiently simplified if we recognize again what our key features were: openness, optionality, affixation, numeracy, and binding. Openness, numeracy, and optionality are not dependent on $\boldsymbol{H}$ but merely on universal or heuristic properties about T. For affixation, we essentially only need to remember for isolated situations whether we have previously constituted nouns as affixed. For binding, if we claim POS *A* is bounded to POS *B* and if we propose that cluster K_A represents class *A*, then if we have assigned cluster K_B to be a *B*, K_A must show binding properties to K_B.

3.2 Processing order

The last component of the description of the induction process is to declare how we can actually do this computation. Given ρ possible parts of speech and *N* clusters, it is clear that since we do not allow duplicate POS tags, there are ρ!/(ρ-N)! necessary hypotheses to examine if we are to compute the whole space of possibilities. We mentioned that, for this study, ρ=20 (and N=20), so this would mean there would be roughly 2.43×10^{18} settings to have to compute. Clearly this is impossible with current resources.

However, by carefully arranging the order that we hypothesize POSs, we can reduce this computation significantly. Effectively, the goal is to remove garden paths early in processing. We will explain how this is done (though it should be mentioned that the start-to-finish process still takes about 30-40 minutes on a 550 MHZ Pentium III).

As was mentioned before, the computation must be ordered so that if class X relies on class Y, then class Y has to be hypothesized before X. A convenient way for doing this is to assume each cluster is Y, compute the probability of each assignment, and retain a probability-sorted list of all the assignments which had positive probabilities. Then, for each such configuration, one can compute the probability that each of the remaining categories is X. This strategy only saves computation if classes that are induced early have numerous zero-probability settings, so we will try to order them according to this need. However, even with such ordering, the number of computations can still become large. Therefore, we retain only the top 1000 answers from one level of hypothesis to the next.

We first identify the classes of punctuation, since we were given these, as well as classes containing letters (which we derive from the Schone/Jurafsky output as being non-numerals and non-punctuation). Since there are a limited number of open classes and since many classes bind to or modify nouns, nouns are the ideal next hypothesis. We then hypothesize numbers, verbs, and adjectives, each of which are open classes and which relate to or modify nouns. Furthermore, these classes exist in most languages so one would expect that there is a cluster representing each.

The remaining set of POSs are induced according to three principles: (1) their expected likelihood, (2) the degree to which other classes are dependent upon them, and (3) the degree to which the statistics at this point are strong enough to clearly identify them. From the junction tree, we can compute the language universal probability of each of the classes. Since some POSs do not exist in all languages, we want to hypothesize less likely classes later in the induction process. However, as suggested by (2), if other classes are dependent upon their existence, they should be hypothesized before their dependents. The last point, (3), addresses the fact that in language, POSs do not all operate at the same linguistic level. For example, in English, adjectives directly precede their corresponding nouns. Although determiners precede their head noun, there may be an adjective between the determiner and the noun. Prepositions precede their nouns, but they may be separated by adjectives or determiners. Question words actually precede whole sentences.

For (3), then, it seems beneficial to hypothesize in stages. In the first stage, one may want to find nouns, verbs, and the noun-bound and verb-bound classes. Then, before the second stage begins, one can "purge" all the bounded items from the data and recompute statistics. In other words, if we sought prepositions as a second stage of processing, and if we had already found nouns, determiners, and adjectives, we could throw away all the determiners and adjectives and thus bring the preposition in direct connection with its corresponding noun. In subsequent levels of processing, we can delete other classes, making it easier to find classes which operate on chunks rather than on words.

Note, though, that since not every class exists in every language, we may end up trying to hypothesize a class that does not exist in the language we are testing in. To remedy this, we create a dummy cluster (-1) which represents the empty set. The probability that a class may need to be assigned to the dummy class is one minus the *a priori* probability of that class's existence across languages.

A last comment is in order. Since updating the junction tree is expensive, an appropriate time for doing this is between levels. If we determined that class X precedes Y, we can feed this information to the junction tree between stages.

4 Performance

The results using this system are preliminary and a number of desirable features are currently being implemented (such as making full use of the junction tree). Despite their preliminary nature, our early results are quite promising, as will be seen.

As mentioned, our dataset from the Brown corpus was reduced to 20 arbitrary clusters ordered as follows:

1: DT determiners	2: NP proper nouns	3: V verbs	4: N nouns
5: I prepositions	6: G genitive	7: J adjectives	8: PU punctuation
9: R adverbs	10: RQ questioners	11: C conjunctions	12: TO infinitive mrk
13: P pronoun	14: M numbers	15: VM modal verbs	16: PDT pronom. DT
17: EX existentials	18: LS lists	19: UH interjection	20: FO foreign words

Table 3: Identifiers and ordering of the "perfect" clusters.

Since our algorithm uses information from language universals, it is unaware of PDT, EX, LS, and FO categories. On the other hand, the universals introduce additional categories such as X (negative), PP (postposition), AUX (auxiliary), QUAL (qualifier), and BE (copula).

We illustrate in Table 3 the 1-best output of our system after each stages of the initial hypotheses are made. After each stage of hypothesizing (phases 12 and 17), the algorithm retains as valid only those tags whose cumulative scores across all of the final 1000 hypotheses that exceed 75% of the cumulative scores for all 1000 hypotheses. For this reason phase 13 and 18 have fewer entries than 12 and 17.

Symbols in bold represent the first time the correct prediction was made. Slanted symbols indicate partially correct assignments. Brackets represent errors. Let us consider the errors. It chose in phase 6 to assign as proper nouns the class that actually belongs to interjections, which makes some sense since both classes are open. In phase 9, the algorithm chose X in lieu of EX. This is completely understandable since both are tightly bound to verbs and since language universals indicate that most languages have an X category and make no mention of EX. The algorithm failed to find the infinitive marker in phase 11 since the probability of its assignment was less than the *a priori* probability of it *not* existing. Phases 13 and 14 are interesting: if one united I and PP into "adpositions," the algorithm would have made the right choice. Since "N PREP ... N" occurs frequently, as in "Secretary of state," it reasoned that the first "N PREP" was the right way to view the data rather than "PREP...N." If we assume half credit for this case and full credit for remaining boldfaced symbols, we see that it properly assigned 14 out of

Phase	Output	Prob
1	**PU: 8**	1.0
2	PU:8, **N: 4**, 2, or 7	1.0
3	PU:8, N: 4, **M:14**	0.008
4	Phase 3 + **V: 3**	0.005
5	Phase 4 + **J: 7**	0.0087
6	[Phase 5 + NP: 19]	4.0e-3
7	Phase 6 + **G: 6**	1.8e-5
8	Phase 7 + **R:9**	8.3e-7
9	[Phase 8 + X: 17]	2.2e-7
10	Phase 9 + **DT:1**	4.9e-8
11	Phase 10 + **AUX: -1**	2.5e-9
12	[Phase 11 + TO:-1]	1.2e-9
13	[Phase 10 + I: -1	0.22
14	Phase 13 + ***PP: 5***]	0.03
15	Phase 14 + **QUAL: -1**	0.11
16	Phase 15 + **BE: -1**	0.006
17	Phase 16 + **C: 11**	0.003
18	Phase 13+ C:11+ I:-1	0.22
19	Phase 18: QUAL: -1	0.082
20	Phase 19: **P: 13**	0.009

Table 4: 1-best output and score at each of the first 20 stages.

18 of the non-redundant assignments: 77.7%. Given that the algorithm used only language universals and otherwise no human information, this is a tremendous accomplishment.

As mentioned, we still are working to make optimal use of the junction tree. One may wonder: how well would the algorithm have done if the junction tree had not existed at all? For the first 20 phases, the algorithm with the Bayes net made assignments with 55% accuracy. This suggests that our selected features carry a significant degree of strength for the algorithm and that the universal ordering rules that exist in the Bayesian network help to nudge the algorithm in the right direction when it is indecisive.

Conclusions

We presented a theoretical approach to automatically tagging syntactic clusters with corresponding POS tags. Unlike other approaches to tagging, this algorithm uses no hand-tagged corpus, no lexicon, and no language-specific information for making decisions. Rather, we proposed six features that it extracted from the syntactic clusters (openness, affixation, numeracy, optionality, binding, and word order), which, when combined with readily-accessible information on language universals, can help automatically tag the clusters for POS. To our knowledge, such a task has never been attempted. We also illustrated a preliminary evaluation of this process on the Brown corpus and showed that the system can accurately tag most of the major categories of POS as well as classes which are noun-bound or verb-bound. We firmly believe that performance will improve as we continue testing, incorporating more of the information from the language universals, and taking advantage of other extractable features which we are currently omitting. Our future directions are to expand the algorithm to handle actual, automatically-induced clusters in English and, more importantly, in other languages.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments.

References

[Bahl and Mercer, 1976] L.R. Bahl and R.L. Mercer. Part of speech assignment by a statistical decision algorithm. *Proceedings of IEEE International Symposium on Information Theory*, pages 88-89.

[Brown, *et al.*, 1992] Peter Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. della Pietra, Jennifer C. Lai. Class-based n-gram models of natural language, *Proc. of the ACL 1992*, pages 467-479, 1992.

[Brill, 1995] Eric Brill, Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4), pages 543-566, 1995

[Church, 1989] Kenneth Ward Church. A stochastic parts program and noun phrase parser for unrestricted text, *Proceedings of ICASSP-89*, Glasgow, Scotland, pages 695-698, 1989.

[Clark, 2000] Alexander Clark. Inducing syntactic categories by context distribution clustering", *Proc. of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, pages 91-94, 2000.

[Comrie, 1989] Bernard Comrie. *Language Universals & Linguistic Typology*, 2ed. University of Chicago Press, Chicago, 1989.

[Croft, 1991] William Croft. *Syntactic Categories and Grammatical Relations*. University of Chicago Press, Chicago, 1991.

[Finch and Chater, 1992] Finch, S. and N. Chater. Bootstrapping syntactic categories. Proc. of the 14th Cognitive Science Society Meeting, pages 820-825, 1992.

[Francis and Kučera, 1982] Francis, W.N. Kučera, H. *Frequency analysis of English usage: Lexicon and Grammar*. Houghton Mifflin, Boston, 1982.

[Greenberg, 1966] Joseph Greenberg. Some universals of grammar with particular reference to the order of meaningful elements. In J. Greenberg, ed., *Universals of Language*, MIT Press, Cambridge, MA, pages 73-113, 1966.

[Harris, 1962] Zelig Harris. *String Analysis of Sentence Structure*, Moulton, The Hague, 1962.

[Jelinek, 1985] Fred Jelinek. Markov source modeling of text generation. In J.K. Skwirzinski, Ed. Nijhoff, Dordrecht, *The Impact of Processing Techniques on Communications*, 1985.

[Jensen 1996] Finn V. Jensen. *An Introduction to Bayesian Networks*. UCL Press, London, 1996.

[Klein and Simmons, 1963] S. Klein and R.F. Simmons. A computational approach to grammatical coding of English words. *Journal of the Association for Computing Machinery*, 10(3): 334-347, 1963.

[Kupiec, 1992] J. Kupiec. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, 6:225-242, 1992.

[Marcus, et al., 1993] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. Computational Linguistics, Vol. 2, No. 19, pages 313-330, 1993.

[Merialdo, 1991] Bernardo Merialdo. Tagging text with a probabilistic model. CH2977-7/91/0000-0809/IEEE, 1991.

[Niesler, *et al.*, 1998] T.R. Niesler, E.W.D. Whittaker, Philip C. Woodland. (1998) "Comparison of part of speech and automatically derived category-based language models for speech recognition," *ICASSP98*, 0-7803-4428-6/98, IEEE.

[Pearl, 1988] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, San Francisco, CA, 1988.

[Pereira, *et al.*, 1993] Fernando Pereira, Naftali Tishby, Lillian Lee. Distributional clustering of English words. *ACL*, 1993.

[Plank and Filiminova, 1996] Frans Plank and Elena Filiminova. *Universals Archive*, http://www.ling.uni-konstanz.de/pages/proj/sprachbau.htm, Universität Konstanz, 1996.

[Samuelsson and Voutilainen, 1997] Comparing a linguistic and a stochastic tagger, *Proc. of 35th ACL and 8th EACL*, Madrid, 1997.

[Schone and Jurafsky, 2000] Patrick Schone and Daniel Jurafsky. Knowledge-free induction of morphology using Latent Semantic Analysis. *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, pages 67-72.

[Schütze, 1993] Hinrich Schütze. Distributed syntactic representations with an application to part-of-speech tagging. *Proc. of the IEEE International Conference on Neural Networks*, pages 1504-1509, 1993.

[Voutilainen, 1995] Atro Voutilainen. Morphological disambiguation. In F. Karlsson, A. Voutilainen, J. Heikkilä, and A. Antilla (Eds.), *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*, pages 165-284, Mouton de Gruyter, Berlin, 1995.

[Yarowsky, *et al.*, 2000]. David Yarowsky, Grace Ngai, Richard Wicentowski. Inducing multilingual text analysis tools via robust projection across aligned corpora. *Proc. of HLT 2001*, pages 109-116, San Diego, CA, 2001.