Why Build Another Part-of-Speech Tagger? A Minimalist Approach.

Leonid Peshkin
Department of Computer Science
Harvard University
Cambridge, MA 02138
pesha@eecs.harvard.edu

Virginia Savova
Department of Cognitive Science
Johns Hopkins University
Baltimore, MD 21218
savova@jhu.edu

Abstract

We use a Dynamic Bayesian Network (DBN) to build a compact representation of the features relevant to Part-of-Speech (PoS) tagging (Word, Suffix, Prefix, Capitalization, Hyphen, Numeric and Previous Tag). The outcome is a flexible tagger (LegoTag) with state-of-the-art performance (3.6% error on a benchmark corpus). We explore the effect of radically reducing the size of feature vocabularies for Word and Suffix. We find that reducing the Suffix vocabulary to a linguistically motivated set, results in improved cross-corpora generalization. Furthermore, relying on function words alone is sufficient to achieve reasonable performance, but minimizing the vocabularies for both Word and Suffix results in degradation.

1 Part of Speech Tagging

Many NLP applicationssss are faced with the dilemma whether to use statistically extracted or expert-selected features. There are good arguments in support of either view. Statistical feature selection does not require extensive use of human domain knowledge, while feature sets chosen by experts are more economical and generalize better to novel data.

Most currently available PoS taggers perform with a high degree of accuracy. However, it appears that the success in performance can be overwhelmingly attributed to an across-the-board lexicalization of the task. Indeed, Charniak et al. [1993] notes that a simple strategy of picking the most likely tag for each word in a text leads to 90% accuracy. If so, it is not surprising that taggers using vocabulary lists, with number of entries ranging from 20k to 45k, perform well. Even though a unigram model achieves an overall accuracy of 90%, it relies heavily on lexical information and is next to useless on nonstandard texts that contain lots of domain-specific terminology.

The lexicalization of the PoS tagging task comes at a price. Since word lists are assembled from the training corpus, they hamper generalization across corpora. In our experience, taggers trained on the Wall Street Journal (WSJ) perform poorly on novel text e.g. e-mail or newsgroup messages (a.k.a. Netlingo). At the same time, alternative training data are scarce and expensive to create.

This paper explores an alternative to lexicalization. Using linguistic knowledge, we construct a minimalist tagger with a small but efficient feature set, which maintains a reasonable performance across corpora.

A look at the previous work on this task reveals that the unigram model is at the core of even the most sophisticated taggers. The best-known rule-based tagger [Brill'94] works in two stages: it assigns the most likely tag to each word in the text; then, it applies transformation rules of the form "Replace tag X by tag Y in triggering environment Z". The triggering environments span up to three sequential tokens in each direction and refer to words, tags or properties of words within the region. The Brill tagger achieves less than 3.5% error on the Wall Street Journal (WSJ) corpus. However, its performance depends on a com-

prehensive vocabulary (70k words) employed in the first stage.

Statistical tagging is a classic application of Markov Models (MMs). Brants [2000] argues that second-order MMs can also achieve state-of-the-art accuracy, provided they are supplemented by smoothing techniques and mechanisms to handle unknown words. TnT handles unknown words by estimating the tag probability given the suffix of the unknown word and its capitalization. The reported 3.3% error for Trigrams 'n Tags (TnT) tagger on the WSJ (trained on 1 million words and tested on 10,000) appears to be a result of overfitting. Indeed, this is the maximum performance obtained by training TnT until only 2.9% of words are unknown in the test corpus. A simple examination of WSJ shows that such percentage of unknown words in the testing section (10% of WSJ corpus) requires simply building a unreasonably large lexicon of nearly all (about 44k) words seen in the training section (90% of WSJ), thus ignoring the danger of overfitting. Hidden MMs (HMMs) are trained on a dictionary with information about the possible PoS of words [Jelinek'85; Kupiec'92]. This means HMM taggers also rely heavily on lexical information.

Obviously, PoS tags depend on a variety of sub-lexical features, as well as on the likelihood of tag/tag and tag/word sequences. In general, all existing taggers have incorporated such information to some degree. The Conditional Random Fields (CRF) model [Lafferty et al.'02] outperforms the HMM tagger on unknown words by extensively relying on orthographic and morphological features. It checks whether the first character of a word is capitalized or numeric; it also registers the presence of a hyphen and morphologically relevant suffixes (*-ed, -ly, -s, -ion, -tion, -ity, -ies*). The authors note that CRF-based taggers are potentially flexible because they can be combined with feature-induction algorithms. However, training is complex (AdaBoost + Forward-backward) and slow (1000 iterations with optimized initial parameter vector; fails to converge with unbiased initial conditions). It is unclear what is the relative contribution of features in this model.

The Maximum Entropy (MaxEnt) [Ratnaparkhi'96] tagger accounts for the joint distribution of PoS tags and features of a sentence with an exponential model. Its features are along the lines of the CRF model:

- Does the token contain a **capital** letter;
- Does the token contain a **hyphen;**
- Does the token contain a **number;**
- Frequent **prefix**es, up to 4 letters long;
- Frequent **suffix**es, up to 4 letters long;

In addition, Ratnaparkhi uses lexical information on frequent words in the context of five words. The sizes of the current word, prefix, and suffix lists were 6458, 3602 and 2925, respectively. These are supplemented by special Previous Word vocabularies. Features frequently observed in a training corpus are selected from a candidate feature pool. The parameters of the model are estimated using the computationally intensive procedure of Generalized Iterative Scaling to maximize the conditional probability of the training set given the model. MaxEnt tagger has 3.4% error rate.

Our investigation examines to what extent the information carried by word lists can be subsumed under the information supplied by sublexical features. In order to address these issues we reuse the feature set of MaxEnt in a new model, which we subsequently minimize with the help of linguistically smart vocabularies.

2 PoS Tagging Bayesian Net

This section presents our tagger, which combines the features suggested in the literature to date into a Dynamic Bayesian Network (DBN). We briefly introduce the essential aspects of DBNs here and refer the reader to a recent PhD thesis [Murphy'02] for an excellent survey. A DBN is a Bayesian network unwrapped in time, such that it can represent dependencies between variables at adjacent

time slices. More formally, a DBN consists of two models B^0 and B^+, where B^0 defines the *initial distribution* over the variables at time 0, by specifying:

- set of variables $X_1, \ldots, X_n$;
- directed acyclic graph over the variables;
- for each variable X_i , a table specifying the conditional probability of X_i given its parents in the graph $\Pr(X_i \mid \mathrm{Par}\{X_i\})$.

The joint probability distribution over the initial state is $\Pr(X_1,...,X_n) = \prod_1^n \Pr(X_i \mid \mathrm{Par}\{X_i\})$. The *transition model* B^+ specifies the conditional probability distribution (CPD) over the state at time t given the state at time *t-1* . B^+ consists of:

- a directed acyclic graph over the variables $X_1,\ldots,X_n$ and their predecessors $X_1^-,...,X_n^-$ - roots of this graph;
- for each X_i (but not X_i^-), a conditional probability table $\Pr(X_i \mid \mathrm{Par}\{X_i\})$.

The transition probability distribution is: $\Pr(X_1,...,X_n \mid X_1^-,...,X_n^-) = \prod_1^n \Pr(X_i \mid Par\{X_i\})$.

Between them, B^0 and B^+ define a probability distribution over the realizations of a system through time, which justifies calling these BNs "dynamic". In our setting, the word's index in a sentence corresponds to time, while realizations of a system correspond to correctly tagged English sentences. Probabilistic reasoning about such system constitutes inference.

Standard inference algorithms for DBNs are similar to those for HMMs. Note that, while the kind of DBN we consider could be converted into an equivalent HMM, that would render the inference intractable due to a huge resulting state space. In a DBN, some of the variables will typically be observed, while others will be hidden. The typical inference task is to determine the probability distribution over the states of a hidden variable over time, given time series data of the observed variables. This is usually accomplished using the forward-backward algorithm. Alternatively, we might obtain the most likely sequence of hidden variables using the Viterbi algorithm. These two kinds of inference yield resulting PoS tags. Note that there is no need to use "beam search", (cf. [Brants'00]).

Learning the parameters of a DBN from data is generally accomplished using the EM algorithm. However, in our model, learning is equivalent to collecting statistics over co-occurrences of feature values and tags. This is implemented in GAWK scripts and takes minutes on the WSJ training corpus. Compare this to GIS or IIS (Improved Iterative Scaling) used by MaxEnt. In large DBNs, exact inference algorithms are intractable, and so a variety of approximate methods has been developed. However, as we explain below, the number of hidden state variables in our model is small enough to allow exact algorithms to work. For the inference we use the standard algorithms, as implemented in the Bayesian network toolkit (BNT) [Murphy'02].

We base our original DBN on the feature set of Ratnaparkhi's MaxEnt: the set of observable nodes in our network consists of the current word W_i , a set of binary variables C_i , H_i and N_i (for Capitalization, Hyphen and Number) and multi-valued variables P_i and S_i (for Prefix and Suffix), where subscript i stands for position index. There are two hidden variables: T_i and M_i (PoS and Memory). Memory represents contextual information about the antepenultimate PoS tag. A special value of Memory ("Start") indicates the beginning of the sentence. The PoS values are 45 tags of the Penn Treebank tag set [Marcus'94].

Figure 1 represents dependencies among the variables. Clearly, this model makes a few unrealistic assumptions about variable independence and Markov property of the sequence. Empirically this does not present a problem. For the discussion of these issues please see Bilmes [2003] who is using similar models for speech recognition.

Thus, probability of a complete sequence of PoS tags $T_1 \ldots T_n$ is modeled as:

$$\Pr(T_1 \ldots T_n) = \Pr(T_1) \times \Pr(F_1|T_1)$$
$$\times \Pr(T_2|T_1, \text{Start}) \times \Pr(F_2|T_2) \times \Pr(M_2|T_1)$$
$$\times \prod_{i=3}^{n-1} \Pr(T_i|T_{i-1}, M_{i-1}) \Pr(M_i|T_{i-1}, M_{i-1}) \Pr(F_i|T_i)$$
$$\times \Pr(T_n|T_{n-1}, M_{n-1}) \times \Pr(F_n|T_n),$$

where F_i is a set of features at index $i \in [1..n]$ and

$$\Pr(F_i|T_i) = \Pr(S_i|T_i) \times \Pr(P_i|T_i) \times \Pr(W_i|T_i)$$
$$\times \Pr(C_i|T_i) \times \Pr(H_i|T_i) \times \Pr(N_i|T_i).$$

These conditional probabilities are directly estimated from training corpus. We use sections 0-22 of WSJ for training and sections 23, 24 as a final test set. The same split of the data was used in recent publications [Toutanova &Manning'02; Lafferty et al.'01] that report relatively high performance on out-of-vocabulary (OoV) items. The test sections contain 4792 sentences out of about 55600 total sentences in WSJ corpus. The average length of a sentence is 23 tokens. The Brown corpus is another part of UPenn TreeBank dataset, which is of a similar size to WSJ (1016277 tokens) but quite different in style and nature. Brown corpus has substantially richer lexicon and was chosen by us to test the performance on novel text.

3 Experiments and Results

We begin our experiments by combining the original MaxEnt feature set into a DBN we call LegoTag to emphasize its compositional nature. The performance of this initial network is 3.6% of overall error (see Table 1) and closely matches that of MaxEnt (3.4%).

Our first step is to reduce the complexity of our tagger because performing inference on the DBN containing a conditional probability table of 45^3 elements for Memory variable is cumbersome. At the cost of minor deterioration in performance (3.9%, see Table 1), we compress the representation by clustering Memory values that predict similar distribution over Current tag values. The clustering method is based on Euclidian distance between

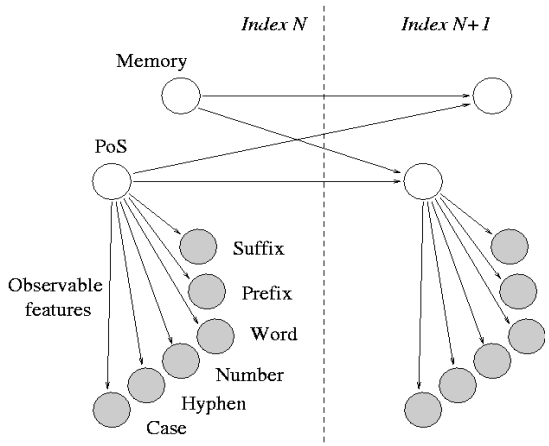

Figure 1: DBN for PoS Tagging.

45^2-dimensional probability vectors $\Pr(T_i|T_{i-1})$. We perform agglomerative clustering, minimizing the sparseness of clusters (by assigning a given point to the cluster whose farthest point it is closest to). As a result of clustering, the number of Memory values is reduced nine times. Consequently, the conditional probability table of Memory and PoS become manageable.

As a second step to simplification of the network, we eliminate feature redundancy. We leave only the lowercase form of each word, prefix and suffix in the respective vocabulary; remove numbers and hyphens from the vocabulary, and use prefix, suffix and hyphen information only if the token is not in the lexicon. The size of the factored vocabularies for word, prefix and suffix is 5705, 2232 and 2420 respectively (a reduction of 12%, 38% and 17%). Comparing the performance of LegoTag with factored and unfactored features clearly indicates that factoring pays off (Table 1). Factored LegoTag is better on unknown words and at the sentence level, as well as overall. In addition, factoring simplifies the tagger by reducing the number of feature values.

We report four kinds of results: overall error, error on unknown words (OoV), per sentence error and confusion matrices over PoS. Our first result (Table 2) shows the performance of our network without the variable

Word, in order to understand whether lexical information is crucial to the performance of a tagger, and how well other features compensate for its absence.

Memory (# values)	Features	Error (%)		
		Ave	OoV	Sentence
Clustered (5)	Unfactored	4.4	13.0	58.5
Clustered (5)	Factored	3.9	10.8	55.8
Full (45)	Factored	3.6	9.4	51.7

Table 1: Results for Full LegoTag on WSJ.

Simply bypassing the word takes performance down to below 90%. However, it is encouraging that even when all words in the text are unknown, the features carry enough information to tag almost 89% of the corpus.

Type of LegoTag						Error (%)		
H	N	C	P	S	W	Ave	OoV	Sentence
+	+	+	+	+	-	11.3	11.3	84.0
-	-	+	-	-	+	6.1	30.6	69.0
-	-	-	-	-	+	9.3	47.6	77.7

Table 2: Results of de-lexicalized and fully lexicalized LegoTag for WSJ corpus.

Next, we test two degenerate variants: one, which contains only lexical information, and another, which contains lexical information plus capitalization only. Lexical information alone does very poorly on unknown words, which comes to show that the sequence is not enough to uncover the correct PoS.

We now turn to the issue of using the morphological cues in PoS tagging and create a linguistically "smart" network (Smart LegoTag), whose vocabularies contain a collection of function words, and linguistically relevant prefixes and suffixes assembled from preparatory materials for the English language section of college entrance examination (Scholastic Aptitude Test). The vocabularies are very small: 315, 100, and 72, respectively. The percentage of unknown words depends on vocabulary size (Table 4). For the large lexicon of LegoTag it is less than 12%, while for the Smart LegoTag (whose lexicon contains only function words which are few but very frequent), it is around 50%. In addition, two hybrid networks are created by crossing the suffix set and word lexicon of the Full LegoTag and Smart LegoTag.

The results for the Smart LegoTag, as well as for the Hybrid LegoTags are presented in Table 3. The results of Smart LegoTag indicate that non-lexical information is sufficient to assure a stable, albeit not stellar, performance across corpora. The network was trained on WSJ and tested on both WSJ and Brown corpora with very similar results. The sentence accuracy is generally lower for the Brown corpus than for the WSJ corpus, due to the difference in average length. The Hybrid LegoTag with big suffix set and small word lexicon was a little improvement over Smart LegoTag alone. Notably, however, it is better on unknown words than Full LegoTag on the Brown corpus.

The best performance across corpora was registered by the second Hybrid LegoTag (with big word lexicon and small suffix set). These is a very interesting result indicating that the non-linguistically relevant suffixes in the big lexicon contain a lot of idiosyncratic information about the WSJ corpus and are harmful to performance on different corpora.The confusion matrix for Full LegoTag and Smart LegoTag are presented in Tables 4 and 5.

LegoTag Ftr Size		Error (%)						Unknown Words (%) WSJ	Unknown words (%) Brown
		WSJ			BROWN				
Word	Suffix	Ave	OoV	Sentence	Ave	OoV	Sentence		
5705	2420	3.9	10	55.4	10.1	23.4	67.9	11.6	15.4
5705	72	4.4	14	58.7	7.7	21.9	69.3		
315	2420	6.4	10.5	70.3	10.1	17.8	76.7	49.2	40.8
315	72	9.6	17.1	82.2	11.4	22.3	82.9		

Table 3: Results for Smart and Hybrid LegoTags.

The correct PoS marks the rows of the table, while the mistakenly assigned PoS marks the columns. The number in each cell refers to the number of times the column PoS was incorrectly assigned instead of the (correct) row PoS, e.g. noun was mislabeled as adjective 536 times.

The column labeled "Table total" sums all errors given in the table for each PoS. The "TOTAL" column contains the number of errors LegoTag made for each PoS (including confusions not presented in the table). The "Contribution to total" is the percent of total error due to the mislabeling of particular PoS.

Per PoS Error	Contribution to error	TOTAL	Table Total	Part of Speech												
				VBP	VBN	VBG	VBD	VB	RB	NNP	NNS	NN	JJ	IN	DT	
1%	2%	103	56	0	0	0	0	0	21	1	0	8	1	25		DT
4%	10%	466	287	0	0	1	0	0	236	1	0	2	9		38	IN
8%	13%	563	547	3	93	47	7	10	114	114	4	143		9	3	JJ
6%	19%	865	850	19	5	79	8	27	9	135	30		536	1	1	NN
2%	3%	145	67	0	0	0	0	0	0	24		15	28	0	0	NNS
3%	7%	300	109	0	0	1	0	4	2		10	28	52	3	9	NNP
10%	8%	374	288	0	0	1	0	4		8	3	29	74	156	13	RB
6%	4%	181	179	104	6	0	3		1	4	0	40	20	1	0	VB
8%	6%	263	263	1	234	0		7	0	4	0	6	11	0	0	VBD
8%	3%	136	136	0	0		0	0	0	3	0	84	49	0	0	VBG
13%	7%	296	295	1		0	134	7	0	2	0	3	148	0	0	VBN
8%	3%	120	119		3	0	8	77	1	1	0	22	6	1	0	VBP
	86%	3812		128	341	129	160	136	384	297	47	380	934	196	64	Table total
			3520	129	341	129	161	136	418	376	133	436	950	240	71	TOTAL

Table 4: Confusion matrix for Full LegoTag on WSJ.

Finally, the "Per PoS error" is the percent of time LegoTag handles incorrectly the given PoS. The row labeled "Table total" sums the number of "false positives" for each PoS.

The row "TOTAL" contains the number of false positives overall (including confusions not presented in the table). Shaded in gray are the sums over all errors in rows and columns, respectively.

Per PoS Error	Contribution to error	TOTAL	Table Total	Part of Speech												
				VBP	VBN	VBG	VBD	VB	RB	NNP	NNS	NN	JJ	IN	DT	
1%	1%	99	54	0	0	0	0	0	11	2	0	8	2	31		DT
6%	7%	734	523	13	17	1	109	23	213	11	12	57	39		28	IN
26%	17%	1830	1781	8	196	83	53	60	217	249	23	875		9	8	JJ
16%	23%	2405	2332	24	64	225	179	66	109	209	261		1177	15	3	NN
6%	4%	380	215	2	4	0	8	5	14	66		90	26	0	0	NNS
3%	3%	366	203	1	3	1	5	1	19		24	36	98	4	11	NNP
25%	9%	939	850	11	33	1	34	44		66	24	225	193	201	18	RB
14%	4%	429	419	42	31	1	48		43	49	6	137	60	2	0	VB
21%	7%	731	721	42	354	0		42	28	3	0	203	30	19	0	VBD
8%	1%	135	135	0	0		0	0	0	3	0	110	22	0	0	VBG
21%	5%	490	489	9		0	132	18	72	14	0	65	175	4	0	VBN
33%	5%	515	491		10	0	292	55	16	4	1	78	32	3	0	VBP
	85%	9053		152	712	312	860	314	742	676	351	1884	1854	288	68	Table total
			9577	178	719	312	885	333	784	880	759	2303	1984	363	77	TOTAL

Table 5: Confusion matrix for Smart LegoTag on WSJ.

The confusion matrices of the Full LegoTag and the Smart LegoTag (Tables 4 and 5) are qualitatively similar. Quantitatively, the percent errors are much larger for the Smart LegoTag, which is a corollary of the higher error

rate overall. Since function words are part of the lexicon of both networks, there is no significant change in the success rate over function words, with the exception of the surprising confusion between prepositions (IN) and past tense verbs (VBD). The biggest source of error in both matrices is the noun/adjective (NN/JJ) pair. By and large, both networks accurately classify the proper nouns, while mislabeling adverbs as prepositions and vice versa. The latter mistake is probably due to inconsistency within the corpus (see [Ratnaparkhi'96] for discussion). One place where the two networks differ qualitatively is in their treatment of verbs. Smart LegoTag often mistakes bare verb forms for nouns. This is likely due to the fact that a phrase involving "to" and a following word can be interpreted either analogously to "to mom" (to + NN) or analogously to "to go" (to + VB) in the absence of lexical information. Similar types of contexts could account for the overall increased number of confusions of verb forms with nouns with Smart LegoTag. On the other hand, Smart LegoTag is much better at separating bare verb forms (VB) from present tense verbs (VBP) because it does not rely on lexical information that is potentially confusing since both forms are identical. However, it often fails to differentiate present verbs (VBP) from past tense verbs (VBD), presumably because it does not recognize frequent irregular forms. One way to improve Smart LegoTag is to put irregular verbs into the lexicon.

4 Conclusion

Our results show that PoS tagging without lexicalization is possible, although more research is required to determine the relevant feature set. One important direction of future work is developing ways to induce the features automatically. We would like to build optimized lexicons with as little handcrafting as possible.

The versatility of the DBN-based tagger makes it adaptable to other languages. We would particularly like to try it on languages whose PoS labels are determined by different kinds of features, such as ones exhibiting richer morphological structure (see [Cucerzan'02]).

References

J. Bilmes. 2003. *Graphical Models and Automatic Speech Recognition*, in "Mathematical Foundations of Speech and Language Processing", Springer.

T. Brants. 2000. *TnT - a statistical part-of-speech tagger.* In Proc. of the 6th ANLP.

E. Brill. 1994. *Some Advances In Rule-Based Part of Speech Tagging*. In Proceedings of the 12th AAAI.

E. Charniak, C. Hendrickson, M. Jacobson and M. Perkowitz. 1993. *Equations for part-of-speech tagging*. In Proc. of the 11th AAAI.

S. Cucerzan, D. Yarowsky. 2002. *Bootstrapping a multilingual part-of-speech tagger in one person-day*. In Proc. of CoNLL.

F. Jelinek. 1985. *Markov source modeling of text generation*. In J. Skwirzinski, ed., Impact of Processing Techniques on Communication, Dordech.

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

J. Lafferty, A. McCallum and F. Pereira. 2001. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*, In Proc. of the 18th ICML.

M. Marcus, G. Kim, M. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz and B. Schasberger. 1994. *The Penn Treebank: Annotating predicate argument structure.* In Proc. of the HLT.

C. Manning and H. Schutze. 1999. *Foundations of Statistical Natural Language Processing.* The MIT Press. Cambridge, MA.

K. Murphy. 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis. UC Berkeley.

A. Ratnaparkhi. 1996. *A maximum entropy model for part-of-speech tagging*. In Proceedings of EMNLP.

C. Samuelsson. 1993. *Morphological Tagging Based Entirely on Bayesian Inference*. In Proc. of the 9th Nordic Conference on Computational Linguistics.

K. Toutanova and C. Manning. 2000. *Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger*. Proc. of EMNLP/VLC.