

# Recurrent Probabilistic Modeling and Its Application to Part-of-Speech Tagging

Gerald Chao

Department of Computer Science  
University of California, Los Angeles  
Los Angeles, California 90095  
gerald@cs.ucla.edu

## Abstract

Recurrent modeling for resolving natural language ambiguities, where downstream information such as structural relationships and word senses is recurrently fed back in successive passes to upstream processes such as part of speech (POS) taggers, is proposed and tested for its ability to improve disambiguation accuracy. In this experiment, POS tagging is used to demonstrate that recurrent information from parsers and word sense disambiguation can be integrated efficiently to improve tagging accuracy. Two types of probabilistic models, Bayesian networks and maximum entropy models, are evaluated for their performance under this recurrent framework. Also, a new search algorithm that determines the best tagging across the whole sentence efficiently and accurately is introduced. The models are trained and evaluated on standard corpora, and we demonstrate that recurrent information does indeed improve the accuracy over traditional, non-recurrent POS taggers.

## 1 Introduction

One of the fundamental problems of natural language processing is resolving ambiguities, present in stemming, part-of-speech (POS) tagging, word sense disambiguation (WSD), anaphora resolution, etc. In tackling this problem, it is commonly divided into discrete steps, such as POS tagging, followed by parsing, then by WSD. While the merit of this task subdivision is beyond the scope of this paper, it has been demonstrated that the information generated at each step is valuable on its own. Furthermore, while the steps can be integrated, such as parsers performing POS tagging, it comes at the expense of

exponential complexity. The approach presented in this paper is to maintain the division of these steps, but as information is produced by later steps, take advantage of this new knowledge to further refine earlier steps. Circularity is not an issue since each step can function independently of information from later steps. However, the hypothesis is that when it does become available, these additional context can be integrated by the upstream steps to improve accuracy, without incurring exorbitant time complexity.

To test this hypothesis, we chose part-of-speech tagging as the evaluation task, with recurrent information provided by the parsing and WSD steps shown in Figure 1. The recurrent process is setup to simulate a tagger that is part of a three step NLP system, where with each further step more contextual information becomes available and is fed back to the tagger. Shown in Table 1, each sentence is processed five times by the tagger, and with each pass the context is expanded. The tagger is composed of a constellation of probabilistic classifiers, i.e., word experts, and computes the probabilities of the associated POS tags based on the available contextual information. The new search algorithm is presented, which can efficiently and effectively find the instantiations that maximize the probability across the whole sentence, in contrast to a greedy search procedure such as a beam search that could result in sub-optimal results.

We demonstrate that by integrating recurrent information from downstream processes, coupled with an improved search procedure, this recurrent POS tagger's accuracy is improved over the traditional, non-recurrent models.

### 1.1 Problem Formulation

Part of speech tagging is treated in this system as a classification task, where the  $n^{th}$  POS tag of a word  $i$  ( $W_i$ ) is classified as the correct one ( $T_i = n$ ), given the word  $W_i$  and usually some surrounding

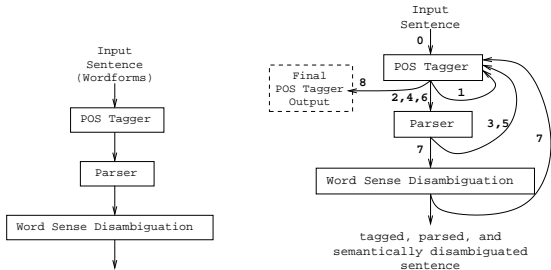


Figure 1: Overview of the recurrent system on the right, and the traditional configuration on the left.

Step	Task	Context words	Attributes	Output
1	tagging	previous words	M	POS tags
2	tagging	4 WW	M	POS tags
3	parsing	4 WW + SFs	POS	SFs
4	tagging	4 WW + SFs	M + S	POS tags
5	parsing	4 WW + SFs + target	POS	SFs + target
6	tagging	4 WW + SFs + target	M + S	POS
7	WSD	"	"	semantic tags
8	tagging	"	M + S + E	final POS tags
9	parsing	"	POS + E	final structure
10	WSD	"	POS + S	final semantic tags

Table 1: The steps in the proposed recurrent process. (WW is the word window, SF are slot fillers, and M, S and E are morphological, structural and semantic attributes, respectively.)

context and this is referred to as  $\tilde{T} = T_{best}(S) = \arg \max P(T|S)$ , where  $S$  is the input sentence, and  $T$  is the sequence of POS tags assigned to each word. Therefore, we wish to find instantiations of  $T$  such that  $P(T|S)$  is maximized.

This process can be modeled as a Hidden Markov Model, which has been shown to be very effective at POS tagging, such as the second order HMM model by Thede and Harper (1999). However, their contextual information is limited to nearby words. Another probabilistic approach is the Maximum Entropy (ME) modeling (Ratnaparkhi, 1998), which is also one of the best POS taggers. However, a beam search strategy is used to estimate the globally maximal POS tagging. There are also non-probabilistic models, such as Brill’s transformation based learning tagger (Brill, 1995) and Roth’s SNOW POS tagger (Roth and Zelenko, 1998). While these models perform well and could potentially benefit from recurrent information, their non-probabilistic nature make it difficult to integrate them with other probabilistic NLP components.

The approach taken in this system is the word-experts model, where each word computes the prob-

ability  $P(T_i|S)$ , and is then maximized across the whole sentence. However, since using a full sentential context  $S$  is impractical, the context is reduced to not only the surrounding words, but also words with long distance dependencies. We evaluated two types of classifiers, Bayesian networks and ME models, under the framework of word-experts with recurrent feedback.

## 2 Contextual Features

The contextual features used in this model consist of not only the POS tags from the surrounding words, but also structural and semantic attributes of the word window and structurally related words. As the structure and the semantics of a sentence become known, these features should provide more contextual information.

Determining the values of these features is done in two steps: 1) locating eight contextual words, and for each word 2) quantifying ten of its attributes. The first four context words are simply the four word window around the current word, and the latter four are the structurally related words that are not necessarily nearby due to long distance dependencies.

To determine the structurally related words and the structural attributes efficiently and consistently, a new structural representation is introduced. This new representation, called Core and Modifiers (CAM), is a simplification of the parse tree, meant to improve the identification and extraction of structurally related words and features.

### 2.1 CAM Representation

A CAM is composed of a core and its modifiers. The core consists of three “slots”, which are filled by slot fillers (SFs). The modifiers of the three slot fillers (MSs) are numbered according to which slot they modify. For example, the sentence “John ate the Italian pasta with meat-balls” in the CAM representation is:

$$\left[ \begin{array}{l} \text{SF1} \text{ 'John'} \\ \text{SF2} \text{ 'ate'} \\ \text{SF3} \text{ 'pasta'} \\ \left[ \begin{array}{l} \text{MS3} \text{ 'the'} \\ \text{MS3} \text{ 'Italian'} \end{array} \right] \\ \left[ \begin{array}{l} \text{MS3} \left[ \begin{array}{l} \text{SF1} \text{ 'with'} \\ \text{SF2} \text{ 'with'} \\ \text{SF3} \text{ 'meat-balls'} \end{array} \right] \end{array} \right] \end{array} \right]$$

One can see that CAM is a simplified representation of sentential structures, designed to capture the main constituents in the cores and limit the modifiers to three slots. Once a sentence is converted to the CAM representation from its parse tree, determining the structural information becomes a simple and consistent lookup. The structurally related words are defined to be the main constituents of the core, i.e.,

Type	Values/Range
Morphological features:	
1) Word form	0-45 (Penn)
2) POS	punc, adj, adv, cc, prep, noun, verb, rel, misc
3) Simplified POS	noun, verb, adjective, adverb, other
4) POS class	none, -s, -ed, -ing, -er, -est, etc.
5) Suffix	
Structural features:	
6) Word slot #	1-6
7) CAM slot #	1-6
8) CAM fill status	0x0 - 0x2F
Semantic features:	
9) Semantic class	0-44 (lexnames file)
10) Synset ID	noun 0-74487, verb 0-12753, adj 0-18522, adv 0-3631

Table 2: The different features used in this system and their range of values.

SF1 through 3, and if a word is a modifier, its target, such as “pasta” as the target for “Italian” from the previous example. Once the contextual words are established, ten features are determined for each word, and their range of values are shown in Table 2.

### 3 Word Experts

The classifiers used to compute  $P(T_i|C_i)$ , where  $C_i$  is the context for the word  $i$ , are built using both ME models and Bayesian networks to compare their effectiveness under the recurrent framework.

#### 3.1 Maximum Entropy Modeling

Maximum entropy models are in the form of  $p^*(S) = \pi \prod_{j=1}^k \alpha_j^{f_j(c,o)}$ ,  $0 < \alpha_j < \infty$ , where  $f_j(c,o)$  is one of the  $k$  binary-valued feature functions,  $\alpha_j$ 's are the parameters adjusted to model the observed statistics, and  $\pi$  is a normalizing factor. A ME model estimates  $p^*(S)$  by adjusting the  $k$  model parameters  $\alpha_j$ ,  $1 < j < k$ , subject to the constraints imposed by the  $k$  feature functions. This can be accomplished by using the Generalized Iterative Scaling (GIS) algorithm, which adjusts each  $\alpha_j$  based on its value from the previous iteration and is guaranteed to converge. For more detailed discussion of ME modeling and GIS, see Ratnaparkhi (1998) and Berger et al. (1996).

#### 3.2 Bayesian Network Models

The Bayesian networks (BNs) used in this system are automatically induced from training data to act as the word experts. To build a Bayesian network classifier, one first determines its structure, or the DAG  $G$ , and then quantifies the conditional probability tables at each node. To automatically learn the structure from data, we use Tian’s (2000) improved BnB.K3 algorithm, which we will refer to as the K4 algorithm. K4 is a systematic search algorithm that will find a graph  $G$  that is guaranteed to have the lowest Minimum Description Length (MDL)

(see (Tian, 2000)) measure, which is a scoring function of how well a graph  $G$  represents the training data, given a node ordering.

The K4 algorithm determines the parents  $PA_i$  from the set of variables  $X_1..X_{i-1}$  for each variable  $X_i$  by systematically searching, but rarely exhaustively owing to efficient pruning, from all combinations of  $X_1..X_{i-1}$  that minimizes the MDL score. Once all of the parents for each node are determined, K4 produces the Bayesian network structure. To quantify the network, the parameters are estimated using maximum likelihood estimation (MLE) using statistics from training data.

#### 3.3 Unknown-Word Expert

Since the lexicon is open in this system, i.e., not all words and their possible POS tags are known *a priori*, a classifier is needed for previously unseen words, or an unknown-word expert. For the ME model, it is built by training on all of the training data using non-recurrent features. That is, the ME unknown-word expert is the same classifier used in Ratnaparkhi’s POS tagger and is thus non-recurrent. A recurrent, ME unknown-word expert is currently unfeasible due to resource constraints. For the BN model, the same feature selection process is used, but due to efficiency reasons, a Naive Bayes network is used instead of inducing a Bayesian network. The parameters are then quantified using MLE.

### 4 Search Algorithm

With the individual word experts constructed, the process of tagging a sentence consists of presenting the classifiers with relevant contextual information, computing the probability  $P(T_i|C_i)$ , and determining the instantiations of  $T_i$  that would maximize  $P(T|S)$  across the whole sentence  $S$ . This search space is potentially exponential, in the worst case being  $O(T^N)$ , where  $N$  is the sentence length, and  $T$  is the number of POS tags at 46. Fortunately, since the individual probability  $P(T_i|C_i)$  is dependent on the surrounding  $2n$  words, referred to as dependent words ( $DW$ ), and is thus independent from the rest of the sentence, the time complexity is reduced to  $O(N \times T^{2n+1})$ . However, in our recurrent model, the context is expanded to the four structurally related words, increasing the complexity to  $O(N \times T^9)$ .

This is the primary reason for choosing the word experts model, to minimize  $T$ , which is the number of POS tags per word being evaluated. The average number of POS tags specific to each word is 2.35, much fewer than 46. However, commonly used words are often more ambiguous, with the worst case having  $T = 10$ . Therefore, a further step is taken to im-

prove efficiency by reducing  $T$  as  $DW$  increases between each pass. That is, when the context is small during earlier passes, all of the POS tags are evaluated. As the context expands between each pass, POS tags with low probabilities are discarded, thus reducing  $T$ , effectively keeping the time complexity linear with a manageable constant.

Let  $S = w_1 \dots w_N$  be the input sentence, and  $T_i$  be the POS tags being evaluated for the word  $i$ , the following search algorithm produces  $\arg \max P(T|S)$ , the instantiations of  $T$  that maximize the probability across the sentence:

1.  $\forall i$  initialize  $T_i \leftarrow$  all tags for word  $w_i$
2. for each pass  $p$ ,
  - (a) for each word  $w_i$ ,  $1 \leq i \leq N$ ,
    - let  $DW_i = \text{Dependent Words}(w_i, p)$ , determine the dependent words based on the current pass.
    - let  $S = \{T_{DW_i}\}$ , the set containing all permutations of the tags  $T$  for  $DW_i$ .
    - for each  $s \in S$ ,
      - $PDT_i[s] \leftarrow P(T_i|s, C_i)$ , classify given the current context and save the probability to the  $s^{th}$  entry of the probability distribution table (PDT) for word  $i$ .
  - (b)  $\tilde{T}_p \leftarrow \text{MAP}(PDT)$ , generate the best instantiations of  $T$  for pass  $p$  based on the PDTs.
  - (c)  $T_i \leftarrow \text{eliminate}(\text{sort}(T_i), \theta_p)$ , reduce the number tags for word  $i$  based on cutoff  $\theta_p$  for pass  $p$ .

The algorithm can be summarized in three steps: 1) classifying each word based on the current context, 2) performing the Maximum A Posteriori (MAP) query and eliminating unlikely candidates, and 3) expanding the context to include long distance dependencies. We will refer to our search algorithm as the CME (Classify, MAP and eliminate, Expand) algorithm. Although this search algorithm is non-systematic, i.e., it cannot guarantee the probability is optimal, we show in the results section that with accurate classifiers, the correct POS tags are ranked highly and are rarely pruned. By balancing time complexity with accuracy via pruning, this search algorithm is efficient while maintaining high accuracy.

## 5 Experiments

To test our recurrent model, the Wall Street Journal (WSJ) portion of the UPenn Treebank is divided into three sections: 1-19 for training, 23-24 for development, and 20-22 for testing. Also, to test the effects of semantic information on POS tagging, the portion of the Brown corpus that is both in the Treebank and SemCor corpora is used as training data.

To model the recurrent feedback process, five passes are used to simulate the multiple stages depicted in Table 1. In the first pass, the previous two

	# of words	# of sentences
Training Data	1,287,652	56,865
WSJ 20-22	108,805	4,537
SENSEVAL-2	5,814	237

Table 3: Statistics on the experimental setup.

Search algorithm	Overall accuracy
Beam	93.67%
CME	94.65%

Table 4: Accuracy comparison between the search algorithms.

words are used to generate the initial tagging, while in the second the four-word window is used. During pass 3, the context from SF1, SF2 and SF3 is used, while for pass 4 the target word TG is added. Lastly, in pass 5 semantic information from word sense disambiguation is added.

Additionally, the test data set from the SENSEVAL-2 Workshop is used to test the effects of both syntactic and semantic recurrent feedback, since the files are both parsed as well as semantically tagged with WordNet 1.7 senses. Some basic statistics of the training and test data is shown in Table 3.

## 6 Results & Discussion

To test the effectiveness of the CME search algorithm, the ME unknown-word expert classifier, which is equivalent to Ratnaparkhi’s POS tagger, is built and trained on the training data for 100 iterations. It is then compared using Ratnaparkhi’s beam search algorithm, with parameter  $K = 5$ , and the CME algorithm. The results are shown in Table 4, and since every word is tagged, precision equals recall. The CME algorithm in this case reduces the error rate by 15.5%, an improvement over one of the state-of-the-art POS tagging algorithms. Furthermore, since only one pass is needed (no recurrent steps), CME in this case guarantees that the probability is maximized over the whole sentence, given the classification probabilities of each word.

### 6.1 Recurrent POS Tagging

Having established the search algorithm, we investigate the impact recurrent information has on POS tagging. Using the same training data but converted to the CAM representation, a BN classifier and a ME model are constructed for each word using all of the features shown in Table 2. Two test sets, the WSJ sections 20-22 and the SENSEVAL-2 test data, are evaluated using both models. The accuracy results are shown in Table 5 and 6, and since the WSJ

Pass	WSJ 20-22		SENSEVAL-2	
	BN	ME	BN	ME
1	94.82%	90.45%	94.89%	94.10%
2	95.21%	93.15%	95.91%	96.41%
3	95.20%	94.29%	96.13%	97.35%
4	<b>95.40%</b>	94.96%	96.39%	<b>97.75%</b>
5	N/A	N/A	96.39%	97.68%
MXPOST	96.72%		97.06%	

Table 5: Overall accuracy of POS taggers with recurrent feedback, with parameter  $\theta_p = \{\infty, 5, 3, -\}$ . The MXPOST model is trained on the same data with 100 iterations.

Pass	WSJ 20-22		SENSEVAL-2	
	BN	ME	BN	ME
1	74.75%	63.12%	62.50%	67.50%
2	76.24%	63.48%	70.00%	67.50%
3	76.16%	63.66%	75.00%	67.50%
4	76.73%	63.73%	72.50%	67.50%
5	N/A	N/A	72.50%	67.50%

Table 6: POS tagging accuracy of unknown words with recurrent feedback.

test set contains no semantic information, the fifth pass is not shown.

As seen in this set of experiments, both classifier models benefit from additional context, whether it’s the expanded word window between pass 1 and 2, or the recurrent structural information in pass 3 and 4. Unfortunately, semantic information did not improve the accuracy, probably due to the limited training data. Nevertheless, as we proposed in our hypothesis, recurrent information does provide useful statistics and context to improve the classification accuracy. For the unknown words, their tagging benefited from having the CME algorithm determine the instantiation that’s maximized across the sentence, improving as the surrounding words’ accuracy between passes improves.

When comparing between the BN and ME models, the most notable aspect is the BN model’s performance under partial evidence, i.e., in the earlier passes. This advantage is not as important in POS tagging, but more so in the integrated system described in the Future Work section, where per word fewer POS tags need to be considered by the parser and WSD since the BN models can better determine the correct tags in the earlier passes. Since parsers are more complex, the time savings from the reduction in POS tags can be significant. Furthermore, on the larger WSJ test set, the BN model performs better than the ME models, demonstrating that Bayesian networks can perform equally to

Pass	WSJ 20-22		SENSEVAL-2	
	BNr	BNnr	BNr	BNnr
1	94.82%	94.61%	94.89%	95.72%
2	95.21%	94.79%	95.91%	96.54%

Table 7: Comparison between recurrent BN models (BNr) with non-recurrent BN models (BNnr) to verify actual gains from feedback information.

one of the best algorithms for NLP.

To ensure that the improvements from our recurrent models did not result from degrading the performance of non-recurrent passes, we built classifiers with only non-recurrent features, i.e., the four-words window. That is, since the recurrent models are given partial evidence during the initial passes, it is possible that they perform poorly until full evidence is presented at the final pass, thus appearing to improve with feedback information. Therefore, we compare the recurrent networks with non-recurrent networks built using the same process, but only with features specific to that pass. The results are shown in Table 7.

One might expect that non-recurrent models should always perform better in this experiment, since all evidence is known and thus providing a complete context. We believe that this is due to *overfitting* to the training data, whereas under partial evidence, sometimes the recurrent models are better able to infer the correct classification. Nevertheless, the non-recurrent models did not outperform the recurrent ones shown in Table 5 on the larger WSJ test data, indicating that the improved performance did not come at the expense of degrading the earlier passes. Therefore, we conclude that recurrent feedback does indeed provide additional statistics to improve non-recurrent models.

## 6.2 CME Revisited

Lastly, we compare the CME’s performance with the empirical upper-bound, determined by providing each Bayesian network classifier with perfect evidence, i.e., the correct POS tags of the dependent words, and choosing the class with the maximum probability. Recall that CME is not systematic, where in the elimination step between passes, unlikely candidates are thrown out, possibly the correct ones. Shown in Table 8, CME is able to stay quite close to the upper-bound, by eliminating mostly the incorrect candidates at each pass. Although it is expected to become less accurate with each pass, as more correct candidates are eliminated, CME is able to remain highly accurate while keeping the time complexity in check. Therefore, efforts can be fo-

P	WSJ 20-22			SENSEVAL-2		
	EUB	CME	$\Delta$	EUB	CME	$\Delta$
1	93.31%	93.19%	-0.1%	92.81%	92.71%	-0.1%
2	94.06%	93.89%	-0.2%	95.48%	94.21%	-1.3%
3	94.16%	93.87%	-0.3%	95.69%	94.51%	-1.2%
4	94.66%	94.16%	-0.5%	95.80%	94.94%	-0.9%
5	N/A	N/A	N/A	95.80%	94.94%	-0.9%

Table 8: Comparison between the empirical upper-bound (EUB), determined by providing the classifiers with perfect evidence, to our CME search algorithm.

cused on improving the accuracy of the individual word experts, since CME will compute  $\tilde{T}$ , the best instantiations across the whole sentence, efficiently, accurately and deterministically.

## 7 Conclusion & Future Work

Resolving natural language ambiguities, we believe, is an integrated process, where POS tagging, parsing, word sense disambiguation, discourse analysis, etc., are all inter-dependent. Each is composed of its own constraints and ambiguities, and instead of compartmentalizing them and solving them separately, we feel that a better approach is to find the most probable explanation that satisfies the most number of constraints.

We demonstrate here that the first step of this process, POS tagging, does benefit from information from downstream processes. The two probabilistic models, Bayesian networks and maximum entropy models, are induced from training data automatically, and the algorithms are deterministic and require no parameter tuning. While the trade off for this recurrent modeling is the added complexity, using the word-experts model to keep the branching factor low, coupled with the CME search algorithm, the time complexity is well-bounded yet with minimal penalty on accuracy.

While our models are achieving respectable results, more work is needed to out-perform the current best. For the ME models, we aim to improve the current implementation to be on par with MXPOST, and when combined with CME, to advance the state-of-the-art accuracy. For the Bayesian network models, we plan to investigate altering the context size and the structure learning algorithm.

An equally important issue is that of parameter learning and over-fitting to the training data. Our initial investigations into parameter learning algorithms resulted in a degradation in accuracy (results not shown), most likely due to over-fitting. We plan to investigate this issue further by setting up rigorous training and evaluation procedures.

Although currently only structural information

improved POS tagging accuracy, we believe that semantic information can further improve accuracy once more training data becomes available, or parameter smoothing techniques such as semantic distance are used.

One benefit of the word-experts model is that, if available, human knowledge can be used to “debug” error-prone words. This is especially applicable to the BN models, where their graphical nature and their CPTs are intuitive to understand. By inspecting the models, humans can alter their structures, add more features, or manually tune the parameters, to improve the word experts’ accuracy. The ME models can also benefit from human assistance, by introducing new feature functions that might be particularly useful to a word expert.

Our goal is to build one integrated system with POS tagger, parser and word sense disambiguation components, as shown in Table 1. Using the same principles demonstrated here, i.e., recurrent feedback, word experts, and the CME algorithm, we are currently developing a parser based on the CAM representation and a WSD system. When completed, the system will be able to produce the most probable interpretation of an input sentence that is maximized across POS tag assignment, structural construct, and word sense distinction.

## References

- Adam Berger, Stephen Della Pietra, and Vincent Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22-1.
- Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21:722–727.
- Adwait Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.
- Dan Roth and Dmitry Zelenko. 1998. Part of speech tagging using a network of linear separators. In *COLING-ACL 98, The 17th International Conference on Computational Linguistics*, pages 1136–1142.
- Scott M. Thede and Mary P. Harper. 1999. A second-order hidden markov model for part-of-speech tagging. In *Proceedings of ACL-99*.
- Jin Tian. 2000. A branch-and-bound algorithm for mdl learning bayesian networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, pages 580–588.