Part of Speech Tagging Using a Network of Linear Separators

Dan Roth and **Dmitry Zelenko**
Department of Computer Science
University of Illinois at Urbana-Champaign
1304 W Springfield Ave., Urbana, IL 61801
{danr,zelenko}@cs.uiuc.edu

Abstract

We present an architecture and an on-line learning algorithm and apply it to the problem of part-of-speech tagging. The architecture presented, *SNOW*, is a network of linear separators in the feature space, utilizing the Winnow update algorithm.

Multiplicative weight-update algorithms such as Winnow have been shown to have exceptionally good behavior when applied to very high dimensional problems, and especially when the target concepts depend on only a small subset of the features in the feature space. In this paper we describe an architecture that utilizes this mistake-driven algorithm for multi-class prediction – selecting the part of speech of a word. The experimental analysis presented here provides more evidence to that these algorithms are suitable for natural language problems.

The algorithm used is an on-line algorithm: every example is used by the algorithm only once, and is then discarded. This has significance in terms of efficiency, as well as quick adaptation to new contexts.

We present an extensive experimental study of our algorithm under various conditions; in particular, it is shown that the algorithm performs comparably to the best known algorithms for POS.

1 Introduction

Learning problems in the natural language domain often map the text to a space whose dimensions are the measured features of the text, e.g., its words. Two characteristic properties of this domain are that its dimensionality is very high and that both the learned concepts and the instances reside very sparsely in the feature space. In this paper we present a learning algorithm and an architecture with properties suitable for this domain.

The *SNOW* algorithm presented here builds on recently introduced theories of multiplicative weight-updating learning algorithms for linear functions. Multiplicative weight-updating algorithms such as Winnow (Littlestone, 1988) and Weighted Majority (Littlestone and Warmuth, 1994) have been studied extensively in the COLT literature. Theoretical analysis has shown that they have exceptionally good behavior in the presence of irrelevant attributes, noise, and even a target function changing in time (Littlestone, 1988; Littlestone and Warmuth, 1994; Herbster and Warmuth, 1995).

Only recently have people started to test these claimed abilities in applications. We address these claims empirically by applying *SNOW* to one of the fundamental disambiguation problems in natural language: part-of speech tagging.

Part of Speech tagging (POS) is the problem of assigning each word in a sentence the part of speech that it assumes in that sentence. The importance of the problem stems from the fact that POS is one of the first stages in the process performed by various natural language related processes such as speech, information extraction and others.

The architecture presented here, *SNOW*, is a Sparse Network Of Linear separators which utilizes the Winnow learning algorithm. A target node in the network corresponds to a candidate in the disambiguation task; all subnetworks learn autonomously from the same data in an online fashion, and at run time, they compete for assigning the correct meaning. A similar architecture which includes an additional layer is described in (Golding and Roth, 1998).

The POS problem suggests a special challenge to this approach. First, the problem is a multi-class prediction problem. Second, determining the POS of a word in a sentence may depend on the POS of its neighbors in the sentence, but these are not known with any certainty. In the *SNOW* architecture, we address these problems by learning at the same time and from the

same input, a network of many classifiers. Each sub-network is devoted to a single POS tag and learns to separate its POS tag from all others. At run time, all classifiers are applied simultaneously and compete for deciding the POS of this word.

We present an extensive set of experiments in which we study some of the properties that *SNOW* exhibits on this problem, as well as compare it to other algorithms. In our first experiment, for example, we study the *quality* of the learned classifiers by, artificially, supplying each classifier with the *correct* POS tags of its neighbors. We show that under these conditions our classifier is almost perfect. This observation motivates an improvement in the algorithm which aims at trying to gradually improve the input supplied to the classifier.

We then perform a preliminary study of learning the POS tagger in an unsupervised fashion. We show that we can reduce the requirements from the training corpus to some degree, but do not get good results, so far, when it is trained in a completely unsupervised fashion.

Unlike most of the algorithms tried on this and other disambiguation tasks, *SNOW* is an online learning algorithm. That is, during training, every example is used once to update the learned hypothesis, and is then discarded. While on-line learning algorithms may be at disadvantage because they see each example only once, the algorithms are able to adapt to testing examples by receiving feedback after prediction. We evaluate this claim for the POS task, and discover that indeed, allowing feedback while testing, significantly improves the performance of *SNOW* on this task.

Finally, we compare our approach to a state-of-the-art tagger, based on Brill's transformation based approach; we show that *SNOW*-based taggers already achieve results that are comparable to it, and outperform it, when we allow online update.

Our work also raises a few methodological questions with regard to the way we measure the performance of algorithms for solving this problem, and improvements that can be made by better defining the goals of the tagger.

The paper is organized as follows. We start by presenting the *SNOW* approach. We then describe our test task, POS tagging, and the way we model it, and in Section 5 we describe our experimental studies. We conclude by discussing the significance of the approach to future research on natural language inferences.

In the discussion below, s is an input example, x_i's denote the features of the example, and c, t refer to parts of speech from a set C of possible POS tags.

2 The *SNOW* Approach

The *SNOW* (Sparse Network Of Linear separators) architecture is a network of threshold gates. Nodes in the first layer of the network represent the input features; target nodes (i.e., the correct values of the classifier) are represented by nodes in the second layer. Links from the first to the second layer have weights; each target node is thus defined as a (linear) function of the lower level nodes.

For example, in POS, target nodes correspond to different part-of-speech tags. Each target node can be thought of as an autonomous network, although they all feed from the same input. The network is *sparse* in that a target node need not be connected to all nodes in the input layer. For example, it is not connected to input nodes (features) that were never active with it in the same sentence, or it may decide, during training, to disconnect itself from some of the irrelevant input nodes, if they were not active often enough.

Learning in *SNOW* proceeds in an online fashion. Every example is treated autonomously by each target subnetworks. It is viewed as a positive example by a few of these and a negative example by the others. In the applications described in this paper, every labeled example is treated as positive by the target node corresponding to its label, and as negative by all others. Thus, every example is used once by all the nodes to refine their definition in terms of the others and is then discarded. At prediction time, given an input sentence $s = (x_1, x_2, \ldots x_m)$, (i.e., activating a subset of the input nodes) the information propagates through all the competing subnetworks; and the one which produces the highest activity gets to determine the prediction.

A local learning algorithm, Littlestone's Winnow algorithm (Littlestone, 1988), is used at each target node to learn its dependence on

other nodes. Winnow has three parameters: a threshold θ, and two update parameters, a *promotion* parameter $\alpha > 1$ and a *demotion* parameter $0 < \beta < 1$. Let $\mathcal{A} = \{i_1, \ldots, i_m\}$ be the set of active features that are linked to (a specific) target node.

The algorithm predicts 1 (positive) iff $\sum_{i \in \mathcal{A}} w_i > \theta$, where w_i is the weight on the edge connecting the ith feature to the target node. The algorithm updates its current hypothesis (i.e., weights) only when a mistake is made. If the algorithm predicts 0 and the received label is 1 the update is (promotion) $\forall i \in \mathcal{A}, w_i \leftarrow \alpha \cdot w_i$. If the algorithm predicts 1 and the received label is 0 the update is (demotion) $\forall i \in \mathcal{A}, w_i \leftarrow \beta \cdot w_i$. For a study of the advantages of Winnow, see (Littlestone, 1988; Kivinen and Warmuth, 1995).

3 The POS Problem

Part of speech tagging is the problem of identifying parts of speech of words in a presented text. Since words are ambiguous in terms of their part of speech, the correct part of speech is usually identified from the context the word appears in. Consider for example the sentence `The can will rust`. Both `can` and `rust` can accept `modal-verb, noun` and `verb` as possible POS tags (and a few more); `rust` can be tagged both as `noun` and `verb`. This leads to many possible POS tagging of the sentence one of which, `determiner, noun, modal-verb, verb`, respectively, is correct. The problem has numerous application in information retrieval, machine translation, speech recognition, and appears to be an important intermediate stage in many natural language understanding related inferences.

In recent years, a number of approaches have been tried for solving the problem. The most notable methods are based on Hidden Markov Models(HMM)(Kupiec, 1992; Schütze, 1995), transformation rules(Brill, 1995; Brill, 1997), and multi-layer neural networks(Schmid, 1994).

HMM taggers use manually tagged training data to compute statistics on features. For example, they can estimate lexical probabilities $Prob(word|tag)$ and contextual probabilities $Prob(tag|previous\ n\ tags)$. On the testing stage, the taggers conduct a search in the space of POS tags to arrive at the most probable POS labeling with respect to the computed statistics. That is, given a sentence, the taggers assign in the sentence a sequence of tags that maximize the product of lexical and contextual probabilities over all words in the sentence.

Transformation based learning(TBL)(Brill, 1995) is a machine learning approach for rule learning. The learning procedure is a mistake-driven algorithm that produces a set of rules. The hypothesis of TBL is an ordered list of transformations. A *transformation* is a rule with an antecedent t and a consequent $c \in C$. The antecedent t is a condition on the input sentence. For example, a condition might be `the preceding word tag is` t. That is, applying the condition to a sentence s defines a feature $t(s) \in \mathcal{F}$. Phrased differently, the application of the condition to a given sentence s, checks whether the corresponding feature is active in this sentence. The condition holds if and only if the feature is active in the sentence.

The TBL hypothesis is evaluated as follows: given a sentence s, an initial labeling is assigned to it. Then, each rule is applied, in order, to the sentence. If the condition of the rule applies, the current label is replaced by the label in the consequent. This process goes on until the last rule in the list is evaluated. The last labeling is the output of the hypothesis.

In its most general setting, the TBL hypothesis is not a classifier (Brill, 1995). The reason is that, in general, the truth value of the condition of the ith rule may change while evaluating one of the preceding rules. For example, in part of speech tagging, labeling a word with a part of speech changes the conditions of the following word that depend on that part of speech(e.g., the preceding word tag is t).

TBL uses a manually-tagged corpus for learning the ordered list of transformations. The learning proceeds in stages, where on each stage a transformation is chosen to minimize the number of mislabeled words in the presented corpus. The transformation is then applied, and the process is repeated until no more mislabeling minimization can be achieved.

For example, in POS, the consequence of a transformation labels a word with a part of speech. (Brill, 1995) uses lexicon for initial annotation of the training corpus, where each word in the lexicon has a set POS tags seen for the

word in the training corpus. Then a search in the space of transformations is conducted to determine a transformation that most reduces the number of wrong tags for the words in the corpus. The application of the transformation to the initially labeled produces another labeling of the corpus with a smaller number of mistakes. Iterating this procedure leads to learning an ordered list of transformation which can be used as a POS tagger.

There have been attempts to apply neural networks to POS tagging(e.g.,(Schmid, 1994)). The work explored multi-layer network architectures along with the back-propagation algorithm on the training stage. The input nodes of the network usually correspond to the tags of the words surrounding the word being tagged. The performance of the algorithms is comparable to that of HMM methods.

In this paper, we address the POS problem with no unknown words (the *closed world assumption*) from the standpoint of *SNOW*. That is, we represent a POS tagger as a network of linear separators and use Winnow for learning weights of the network. The *SNOW* approach has been successfully applied to other problems of natural language processing(Golding and Roth, 1998; Krymolowski and Roth, 1998; Roth, 1998). However, this problem offers additional challenges to the *SNOW* architecture and algorithms. First, we are trying to learn a multi-class predictor, where the number of classes is unusually large(about 50) for such learning problems. Second, evaluating hypothesis in testing is done in a presence of attribute noise. The reason is that input features of the network are computed with respect to parts of speech of words, which are initially assigned from a lexicon.

We address the first problem by restricting the parts of speech a tag for a word is selected from. Second problem is alleviated by performing several labeling cycles on the testing corpus.

4 The Tagger Network

The tagger network consists of a collection of linear separators, each corresponds to a distinct part of speech[1]. The input nodes of the network correspond to the features. The features are computed for a fixed word in a sentence. We use the following set of features[2]:
(1) The preceding word is tagged c.
(2) The following word is tagged c.
(3) The word two before is tagged c.
(4) The word two after is tagged c.
(5) The preceding word is tagged c and the following word is tagged t.
(6) The preceding word is tagged c and the word two before is tagged t.
(7) The following word is tagged c and the word two after is tagged t.
(8) The current word is w.
(9) The most probable part of speech for the current word is c.

The most probable part of speech for a word is taken from a lexicon. The lexicon is a list of words with a set of possible POS tags associated with each word. The lexicon can be computed from available labeled corpus data, or it can represent the a-priori information about words in the language.

Training of the *SNOW* tagger network proceeds as follows. Each word in a sentence produces an example. Given a sentence, features are computed with respect to each word thereby producing a positive examples for the part of speech the word is labeled with, and the negative examples for the other parts of speech. The positive and negative examples are presented to the corresponding subnetworks, which update their weights according to Winnow.

In testing, this process is repeated, producing a test example for each word in the sentence. In this case, however, the POS tags of the neighboring words are not known and, therefore, the majority of the features cannot be evaluated. We discuss later various ways to handle this situation. The default one is to use the baseline tags - the most common POS for this word in the training lexicon. Clearly this is not accurate, and the classification can be viewed as done in the presence of *attribute noise*.

Once an example is produced, it is then presented to the networks. Each of the subnetworks is evaluated and we select the one with the highest level of activation among the separators corresponding to the possible tags for the current word. After every prediction, the tag output by the *SNOW* tagger for a word is used for labeling the word in the test data. There-

[1] The 50 parts are taken from the WSJ corpus

[2] The features 1-8 are part of (Brill, 1995) features

fore, the features of the following words will depend on the output tags of the preceding words.

5 Experimental Results

The data for all the experiments was extracted from the Penn Treebank WSJ corpus. The training and test corpus consist of 600000 and 150000, respectively. The first set of experiment uses only the *SNOW* system and evaluate its performance under various conditions. In the second set *SNOW* is compared with a naive Bayes algorithm and with Brill's TBL, all trained and tested on the same data. We also compare with *Baseline* which simply assigns each word in the test corpus its most common POS in the lexicon. Baseline performance on our test corpus is 94.1%.

A lexicon is computed from both the training and the test corpus. The lexicon has 81227 distinct words, with an average of 2.2 possible POS tags per word in the lexicon.

5.1 Investigating *SNOW*

We first explore the ability of the network to adapt to new data. While online algorithms are at a disadvantage - each example is processed only once before being discarded - they have the advantage of (in principle) being able to quickly adapt to new data. This is done within *SNOW* by allowing it to update its weights in test mode. That is, after prediction, the network receives a label for a word, and then uses the label for updating its weights.

In test mode, however, the *true* tag is not available to the system. Instead, we used as the feedback label the corresponding baseline tag taken from the lexicon. In this way, the algorithm never uses more information than is available to batch algorithms tested on the same data. The intuition is that, since the baseline itself for this task is fairly high, this information will allow the tagger to better tolerate new trends in the data and steer the predictors in the right direction. This is the default system that we call *SNOW* in the discussion that follows.

Another policy with on-line algorithms is to supply it with the *true* feedback, when it makes a mistake in testing. This policy (termed adp-*SNOW*) is especially useful when the test data comes from a different source than the training data, and will allow the algorithm to adapt to the new context. For example, a language acquisition system with a tagger trained on a general corpus can quickly adapt to a specific domain, if allowed to use this policy, at least occasionally. What we found surprising is that in this case supplying the true feedback did not improve the performance of *SNOW* significantly. Both on-line methods though, perform significantly better than if we disallow on-line update, as we did for noadp-*SNOW*. The results, presented in table 1, exhibit the advantage of using an on-line algorithm.

noadp-*SNOW*	*SNOW*	adp-*SNOW*
96.5	97.13	97.2

Table 1: **Effect of adaptation:** Performance of the tagger network with no adaptation(noadp-*SNOW*), baseline adaptation(*SNOW*), and true adaptation(adp-*SNOW*).

One difficulty in applying the *SNOW* approach to the POS problem is the problem of attribute noise alluded to before. Namely, the classifiers receive a noisy set of features as input due to the attribute dependence on (unknown) tags of neighboring words. We address this by studying quality of the classifier, when it is guaranteed to get (almost) correct input.

Table 2 summarizes the effects of this noise on the performance. Under *SNOW* we give the results under normal conditions, when the the features of the each example are determined based on the baseline tags. Under *SNOW*+cr we determine the features based on the *correct* tags, as read from the tagged corpus. One can see that this results in a significant improvement, indicating that the classifier learned by *SNOW* is almost perfect. In normal conditions, though, it is affected by the attribute noise.

Baseline	*SNOW*+cr	*SNOW*
94.1	98.8	97.13

Table 2: **Quality of classifier:** The *SNOW* tagger was tested with correct initial tags (*SNOW*+cr) and, as usual, with baseline based initial tags.

Next, we experimented with the sensitivity of *SNOW* to several options of labeling the training data. Usually both features and labels of the training examples are computed in terms of

correct parts of speech for words in the training corpus. We call the labeling *Semi-supervised* when we only require the features of the training examples to be computed in terms of the most probable pos for words in the training corpus, but the labels still correspond to the correct parts of speech. The labeling is *Unsupervised* when both features and labels of the training examples are computed in terms of most probable POS of words in the training corpus.

Baseline	*SNOW*+uns	*SNOW*+ss	*SNOW*
94.1	94.3	97.13	97.13

Table 3: **Effect of supervision**. Performance of *SNOW* with *unsupervised* (*SNOW*+uns), *semi-supervised* (*SNOW*+ss) and normal mode of *supervised* training.

It is not surprising that the performance of the tagger learned in an *semi-supervised* fashion is the same as that of the one trained from the correct corpus. Intuitively, since in the test stage the input to the classifier uses the baseline classifier, in this case there is a better fit between the data supplied in training (with a correct feedback!) and the one used in testing.

5.2 Comparative Study

We compared performance of the *SNOW* tagger with one of the best POS taggers, based on Brill's TBL, and with a naive Bayes (e.g.,(Duda and Hart, 1973) based tagger. We used the same training and test sets. The results are summarized in table 4.

Baseline	NB	TBL	*SNOW*	adp-*SNOW*
94.1	96	97.15	97.13	97.2

Table 4: **Comparison of tagging performance.**

In can be seen that the TBL tagger and *SNOW* perform essentially the same. However, given that *SNOW* is an online algorithm, we have tested it also in its (true feedback) adaptive mode, where it is shown to outperform them. It is interesting to note that a simple minded NB method also performs quite well.

Another important point of comparison is that the NB tagger and the *SNOW* taggers are trained with the features described in section 4. TBL, on the other hand, uses a much larger set of features. Moreover, the learning and tagging mechanism in TBL relies on the inter-dependence between the produced labels and the features. However, (Ramshaw and Marcus, 1996) demonstrate that the inter-dependence impacts only 12% of the predictions. Since the classifier used in TBL without inter-dependence can be represented as a linear separator(Roth, 1998), it is perhaps not surprising that *SNOW* performs as well as TBL. Also, the success of the adaptive *SNOW* taggers shows that we can alleviate the lack of the inter-dependence by adaptation to the testing corpus. It also highlights importance of relationship between a tagger and a corpus.

5.3 Alternative Performance Metrics

Out of 150000 words in the test corpus used about 65000 were non-ambiguous. That is, they can assume only one POS. Incorporating these in the performance measure is somewhat misleading since it does not provide a good measure of the classifier performance.

Baseline	NB	TBL	*SNOW*	adp-*SNOW*
90.1	93	95	95	95.2

Table 5: **Performance for ambiguous words.**

Sometimes we may be interested in determining POS *classes* of words rather than simply parts of speech. For example, some natural language applications may require identifying that a word is a noun without specifying the exact noun tag for the word(singular, plular, proper, etc.). In this case, we want to measure performance with respect to POS classes. That is, if the predicted part of speech for a word is in the same class with the correct tag for the word, then the prediction is termed correct.

Out of 50 POS tags we created 12 POS classes: punctuation marks, determiners, preposition and conjunctions, existentials "there", foreign words, cardinal numbers and list markers, adjectives, modals, verbs, adverbs, particles, pronouns, nouns, possessive endings, interjections. The performance results for the classes are shown in table 5.3.

In analyzing the results, one can see that many of the mistakes of the tagger are "within" classes. We are currently exploring a few issues that may allow us to use class information, within *SNOW*, to improve tagging accuracy. In

Baseline	NB	TBL	*SNOW*	adp-*SNOW*
96.2	97	97.95	97.95	98

Table 6: **Performance for POS classes.**

particular, we can incorporate POS classes into our *SNOW* tagger network. We can create another level of output nodes. Each of the nodes will correspond to a POS class and will be connected to the output nodes of the POS tags in the class. The update mechanism of network will then be made dependent on both class and tag prediction for a word.

6 Conclusion

A Winnow-based network of linear separators was shown to be very effective when applied to POS tagging. We described the *SNOW* architecture and how to use it for POS tagging and found that although the algorithm is an on-line algorithm, with the advantages this carries, its performance is comparable to the best taggers available.

This work opens a variety of questions. Some are related to further studying this approach, based on multiplicative update algorithms, and using it for other natural language problems.

More fundamental, we believe, are those that are concerned with the general learning paradigm the *SNOW* architecture proposes.

A large number of different kinds of ambiguities are to be resolved simultaneously in performing any higher level natural language inference (Cardie, 1996). Naturally, these processes, acting on the same input and using the same "memory", will interact. In *SNOW*, a collection of classifiers are used; all are learned from the same data, and share the same "memory". In the study of *SNOW* we embark on the study of some of the fundamental issues that are involved in putting together a large number of classifiers and investigating the interactions among them, with the hope of making progress towards using these in performing higher level inferences.

References

E. Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–565.

E. Brill. 1997. Unsupervised learning of disambiguation rules for part of speech tagging. In *Natural Language Processing Using Very Large Corpora*. Kluwer Academic Press.

C. Cardie, 1996. *Embedded Machine Learning Systems for natural language processing: A general framework*, pages 315–328. Springer.

R. O. Duda and P. E. Hart. 1973. *Pattern Classification and Scene Analysis*. Wiley.

A. R. Golding and D. Roth. 1998. A winnow based approach to context-sensitive spelling correction. *Machine Learning*. Special issue on Machine Learning and Natural Language; . Preliminary version appeared in ICML-96.

M. Herbster and M. Warmuth. 1995. Tracking the best expert. In *Proc. 12th International Conference on Machine Learning*, pages 286–294. Morgan Kaufmann.

J. Kivinen and M. K. Warmuth. 1995. Exponentiated gradient versus gradient descent for linear predictors. In *Proc. of STOC*. Tech Report UCSC-CRL-94-16.

Y. Krymolowski and D. Roth. 1998. Incorporating knowledge in natural language learning: A case study. COLING-ACL Workshop.

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

N. Littlestone and M. K. Warmuth. 1994. The weighted majority algorithm. *Information and Computation*, 108(2):212–261.

N. Littlestone. 1988. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318.

L. A. Ramshaw and M. P. Marcus. 1996. Exploring the nature of transformation-based learning. In J. Klavans and P. Resnik, editors, *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*. MIT Press.

D. Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proc. National Conference on Artificial Intelligence*.

H. Schmid. 1994. Part-of-speech tagging with neural networks. In *COLING-94*.

H. Schütze. 1995. Distributional part-of-speech tagging. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics*.