An Online Algorithm for Learning over Constrained Latent Representations using Multiple Views

Abstract

We introduce an online framework for discriminative learning problems over hidden structures. In many Natural Language Processing tasks it is helpful to take advantage of latent structural properties of the data. However, since these structures are hidden, solution must learn both the latent structure and the classifier for the task at hand. Previous work on leveraging latent representations for discriminative learners has used batch algorithms that require multiple passes though the entire training data. Instead, we propose an online algorithm that efficiently jointly learns the latent structures and the classifier. We further extend this to include multiple views on the latent structures with different representations, and we explore a variety of ways to incorporate these multiple views in learning the classifier and their interaction. We evaluate our algorithms in comparison to the most closely related batch algorithm, LCLR, and show that our online algorithm significantly outperforms it on a grammaticality task.

1 Introduction

Natural language data is implicitly richly structured, and making use of that structure can be valuable in a wide variety of NLP tasks, from machine translation to paraphrase generation to question answering. However, finding these latent structures is a complex task of its own right, enough to warrant extensive study. Early work used a two-phase pipeline process, in which the output of a structure prediction algorithm acts as fixed input features to train a classifier. Chang et al. (2009), Das and Smith (2009), Goldwasser and Roth (2008), and Mccallum and Bellare (2005) have shown that this approach is problematic in that it can propagate error from the fixed latent structures to the classifier, and that it can fail to take full advantage of the labeled data for the final task in generating the latent structures. Moreover, we may wish to allow feedback between the latent learner and the final task learner in optimizing the latent representation of the data for the final task to be performed. More recent work has come to approach latent structure learning for NLP from the perspective of finding the best latent representation for a particular task, rather than what is necessarily the "correct" underlying representation. Work in this vein has focused on jointly learning the latent structures together with the task-specific classifier they inform Cherry and Quirk (2008), Chang et al. (2010).

Chang et al. (2010) in particular set forth a powerful framework for solving classification problems using constraints over latent structures, referred to as Learning over Constrained Latent Representations (henceforth, LCLR). Inspired by LCLR, we have extended this framework for discriminative joint learning over latent structures to a novel online algorithm. We evaluate the algorithm in comparison to the LCLR batch method on a grammaticality test using a discriminative model that learns over shallow parse (chunk) structures. We show that our online method has standard convergence guarantees for a max-margin learner, but that it attains higher accuracy. Furthermore, in practice we find that it requires fewer passes over the data.

Further, we explore the use of allowing multiple views on the latent structures using different representations in the classifier. This is inspired by Shen and Sarkar (2005), who found that using a majority voting approach on multiple representations of the latent structures on a chunking task outperformed both a single representation as well as voting between multiple learning models. We investigate several different methods of combining the views and we show that the multiple-view approach to latent structure learning yields modest improvements over the single-view classifier.

2 The Grammaticality Task

To evaluate our algorithms, we use a language modeling task as an example application. Language modeling is an important task in many NLP applications, in which is it usually responsible for making sure that generated text is fluent. The predominant n-gram form of language model (LM) bases the likelihood of generated items upon pre-

viously seen word histories. A well-known limitation of n-gram LMs is that they are informed only by the previously seen word-string histories of a fixed maximum length. Thus, while they are sensitive to local disfluencies, they ignore longer distance dependencies between more distant parts of the sentence.

Consider the following example sentences generated by a 3-gram language model:

- chemical waste and pollution control (amendment) bill , all are equal , and , above all else .
- kindergartens are now .

These fragments are composed of viable trigrams, but a human reader could easily judge them to be ungrammatical. However, if our language model were to use latent information such as a syntactic parse of the sentence, our model could recognize their lack of grammaticality based on syntactic cues.

Discriminative models, which have been successfully applied to many natural language tasks, can take into account arbitrary features over a given example, and thus may be able to avoid the shortcomings of n-gram LMs in judging the grammaticality of a piece of text. In the case of language modeling, however, there is no obvious choice of categories between which the model should discriminate. Cherry and Quirk (2008) show that by following the pseudo-negative examples approach of Okanohara and Tsujii (2007), they can build a syntactic discriminative LM that learns to distinguish between samples from a corpus generated by human speakers and samples generated by an n-gram model. In this framework, the samples generated from the n-gram model serve as negative examples. While both positive and negative examples will contain probable n-grams, the negative examples can be expected to contain locally fluent constituents but lack global fluency or structural coherence. Thus the discriminative LM learns to assign high scores to the more grammatical structures that look human-generated and low scores to those that do not.

Our approach is similar to Cherry and Quirk (2008), but they use probabilistic context-free grammar (PCFG) parses as latent structure, use a latent SVM as the learning model, and handle negative examples differently. We use a latent passive-aggressive (PA) learning algorithm rather than the latent SVM used by Cherry and Quirk (2008). Cherry and Quirk (2008) use a PCFG parse as a latent variable for each example. A alternate representation of sentence structure is chunking, which can be seen as a shallow parse. A chunking is represented by tagging each word in the sentence to indicate phrase membership and boundaries, but there are various ways to choose tags to do so.

We train our model on real sentences versus pseudo-negative sentences sampled from an n-gram model. Our model simultaneously learns to apply multiple sets of chunk tags to produce chunkings representing sentence structure and to prefer the shallow parse features of the human sentences to those sampled from an n-gram LM. Since the shallow parse features of the sampled examples are penalized in the learning algorithm, the chunker-based classifier learns chunking models that prefer not necessarily the linguistically best chunking for each example, but rather the chunking that will assign the widest margin between the positive (grammatical) and negative (ungrammatical) examples.

3 Latent Structure Classifier

Our generalized classifier is trained by simultaneously searching for the highest scoring latent structure while classifying data instances. Rather than training the best latent structure to adhere to a linguistically motivated gold-standard, we want to exploit the underlying structure implicit in each example to produce the best classification of the data. Here we use the discriminative latent learning framework due to Chang et al. (2010) together with the passive-aggressive (PA) updating strategy due to Crammer and Singer (2001).

3.1 PA Learning

The latent structure classifier training uses a decision function that searches for the best structure $z_i^* \in Z(x_i)$ for each training sentence x_i with a space of possible structures $Z(x_i)$ according to feature weights $\mathbf{w}$, i.e.:

$$f_w(x_i) = \arg\max_{z_i} \mathbf{w} \cdot \phi(x_i, z_i) \qquad (1)$$

where $\phi(x_i, z_i)$ is a feature vector over the sentence-parse pair. The sign of the prediction $y_i^* \mathbf{w} \cdot \phi(x_i, z_i^*)$ determines the classification of the sentence x_i.

Using passive-aggressive max-margin training (Crammer and Singer, 2001), we incorporate

this decision function into our global objective, searching for the $\mathbf{w}$ that minimizes

$$\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{X} \ell(\mathbf{w} \cdot (f(x_i), y_i)), \qquad (2)$$

where ℓ is a loss function. Setting ℓ to be hinge loss, at each iteration and for each example x_i we find and update according to a new weight vector $\mathbf{w}'$ that minimizes:

$$\frac{1}{2}\|\mathbf{w} - \mathbf{w}'\|^2 + \tau(1 - y_i(\mathbf{w}' \cdot \phi(x_i, z_i^*)), \quad (3)$$

where w is the previous weight vector, z_i^* is the structure found by Eqn. (1), $y_i \in \{-1, 1\}$ is the true label (ungrammatical or grammatical) for the example, and $\tau \geq 0$ is a Lagrange multiplier proportional to the example loss. The second term penalizes classification examples proportionally to the degree to which they violate the margin (see Alg. 1).

3.2 Optimization Method

Because Eqn. (3) contains an inner max over z_i^*, it is not convex for the positive examples, since it is the maximum of a convex function (zero) and a concave function $(1 - y_i(\mathbf{w}' \cdot \phi(x_i, z_i^*))$. In hinge loss, when we drive the inner function to higher values it minimizes the outer problem for negative examples, but maximizes it for the positive ones. So, as in LCLR, we hold the latent structures fixed for the positive examples but can perform inference to solve the inner minimization problem for the negative examples.

3.3 Online Training

Our online training method is shown as algorithm 1. It applies the structured prediction and PA update of section 3 on a per-example basis in a variant of the cutting plane algorithm discussed in Joachims and Yu (2009). Since for the positive examples the latent structures are fixed per-iteration, it does a single search and update step for each example at each iteration. For negative examples it repeats the prediction and PA update for each example until the model correctly predicts the label (i.e. until $y_i^* = y_i$). Since the negative examples should never be associated with good latent structures under the model, we wish to penalize all possible structures for the negative examples. However, because this is intractable to compute, we use the approximation of the single-best structure for each negative example. We re-decode the

```
1  initialize w_0
2  for t = 0, ..., T − 1 do
3      for each training example x_i in X do
4          repeat
5              find z_i* = arg max_{z_i} w_t · φ(x_i, z_i)
6              let y_i* = w_t · φ(x_i, z_i*)
7              let loss l_t = max{0, 1 − y_i y_i*)}
8              let multiplier τ_t = l_t / ||φ(x_i, z_i*)||^2
9              update
               w_{t+1} := w_t + τ_t y_i φ(x_i, z_i*)
10         until y_i > 0 or (y_i* = y_i if y_i < 0);
11 return w_T
```

Algorithm 1: Online PA algorithm for binary classification with latent structures.

negative examples until the highest scoring structure is correctly labeled as negative. This approximation is analogous to the handling of inference over negative examples in the batch algorithm described in Chang et al. (2010). In the batch version, however, updates for all negative examples are performed at once and all are re-decoded until no new structures are found for any single negative example. We do not retain any information about negative examples between iterations as do Cherry and Quirk (2008) or a cache of negative structures per-example as in LCLR.

Alg. 1 is a soft-margin classifier and we can write down the objective function in terms of a slack variable ξ and assuming hinge loss:

$$\mathbf{w}_{t+1} \quad = \quad \arg\min_{\mathbf{w}} \|\mathbf{w} - \mathbf{w}_t\|^2 + \sum_i \xi_i^2$$

s.t. $\forall_i \max(0, 1 - y_i(\max_{z \in Z(x_i)}(\mathbf{w} \cdot \phi(x_i, z)))) \leq \xi_i$.

The update in line 9 of Alg. 1 is justified using Lagrange multipliers to optimize the objective per iteration as shown in (Crammer and Singer, 2001). This is a semi-convex problem which becomes clear if the above objective function is expanded into two special cases: one for positive and one for negative examples. The first max in the above objective means that the objective for positive examples is convex when we fix the value of z while the objective for negative examples is convex even if we search for the maximum z. This is what we do repeatedly for negative examples in Alg. 1 while we make a step in the objective using the $\arg\max$ value of z for all positive examples.

For any weight vector $\mathbf{w}'$, ℓ_t^* is the loss for $\mathbf{w}'$

at round t which we write as

$$\ell_t^* = \max(0, 1 - y_t(\max_{z \in Z(x_t)} (\mathbf{w}' \cdot \phi(x_t, z))))$$

which we compare to the weight vector produced by Alg. 1 in round t which is

$$\ell_t = \max(0, 1 - y_t(\max_{z \in Z(x_t)} (\mathbf{w}_t \cdot \phi(x_t, z))))$$

.

By splitting up the positive and negative examples we adapt the proof from (Crammer and Singer, 2001) to show that at round t the cumulative squared loss of Alg. 1 on a particular sequence of examples is bounded from above:

$$\sum_{t=1}^{T} \ell_t^2 = \left(\|\mathbf{w}'\| + 2\sqrt{\sum_{t=1}^{T} (\ell_t^*)^2} \right)^2$$

The proof of the above convergence bound on the loss can be derived by combining the split view of positive and negative examples described in (Felzenszwalb et al., 2010) and Theorem 3 from (Crammer and Singer, 2001).

3.4 Multiple Views on Latent Representations

To incorporate multiple views on the latent representations, we perform inference separately for each view for each training example.

In our single-view model, the latent structures are provided by a chunker which splits a sentence into phrasal constituents. Shen and Sarkar (2005) find that using multiple chunking representations is advantageous for the chunking task itself. Moreover, they demonstrate that the careful selection of latent structure can yield more helpful features for a task-specific classifier. We thus generate separate latent structures for each of their five chunking representations (which are mostly from (Sang and Veenstra, 1999)) at line 5 of Alg. 1.

The addition of multiple views on latent representations retains the same semi-convexity properties of our single-view algorithm. Each view r is a convex function generating some latent structure over training example x_i. Since the combination of the views is just the sum over each function r, the result is another convex function, so the objective function remains the same.

Each of the views use a different representation of the chunk structures, which we will briefly describe here. For more detailed information, please see Sang and Veenstra (1999). Each representation uses a set of tags to label each token in a sentence as belonging to a non-overlapping chunk type. We refer to the chunking schemas as IOB1, IOB2, IOE1, IOE2, and O+C. The total set of available tags for each of the representations are B- (current token begins a chunk), I- (current token is inside a chunk), E- (current token ends a chunk), S- (current token is in a chunk by itself), and O (current token is outside of any chunk). All chunks except O append the part-of-speech tag of the token as a suffix. IOB1, IOB2, IOE1, and IOE2 are variant of inside/outside representations. The IOB representations distinguish only between B, I, and O. IOB1 differs from IOB2 in that it assigns the tag B only if the token that follows is inside a chunk. IOE1 and IOE2 only use E, I, and O, and they differ in that IOE2 assigns the E tag regardless of whether the token that follows is inside a chunk. Table 1 shows the different chunking schemas on an example sentence.

Token	IOB1	IOB2	IOE1	IOE2	O+C
In	O	O	O	O	O
early	I	B	I	I	B
trading	I	I	I	E	E
in	O	O	O	O	O
Hong	I	B	I	I	B
Kong	I	I	E	E	E
Monday	B	B	I	E	S
,	O	O	O	O	O
gold	I	B	I	E	S
was	O	O	O	O	O
quoted	O	O	O	O	O
at	O	O	O	O	O
$	I	B	I	I	B
366.50	I	I	E	E	E
an	B	B	I	I	B
ounce	I	I	I	E	E
.	O	O	O	O	O

Table 1: The five different chunking representations for the example sentence "In early trading in Hong Kong Monday , gold was quoted at $ 366.50 an ounce ."

Each of these chunking schemas can be conceived as a different kind of expert. The inside/outside schemas vary over their sensitivity to finding chunks that span multiple tokens. The IOB variants will be better at detecting where a chunk begins, whereas the IOE variants will do better at detecting the chunk's end. O+C allows for a more fine-grained representation of the chunk types.

It is straightforward to use dynamic program-

1 initialize $\mathbf{w}_0$
2 **for** $t = 0, ..., T-1$ **do**
3 | **for** *each training example* x_i *in* X **do**
4 | | **repeat**
5 | | | **for** *view* $r \in R$ **do**
6 | | | | find $z^*_{i_r} = \arg\max_{z_i} \mathbf{w}_t \cdot \phi(x_i, z_i)$
7 | | | let $\phi(x_i, z^*_i) =$ COMBINE$(\{\phi(x_i, z^*_{i_r}) : r\})$
8 | | | let $y^*_i = \mathbf{w}_t \cdot \phi(x_i, z^*_i)$
9 | | | let loss $l_t = \max\{0, 1 - y_i y^*_i)\}$
10 | | | let multiplier $\tau_t = \frac{l_t}{\|\phi(x_i, z^*_i)\|^2}$
11 | | | update $\mathbf{w}_{t+1} := \mathbf{w}_t + \tau_t y_i \phi(x_i, z^*_i)$
12 | | **until** $y_i > 0$ *or* $(y^*_i = y_i$ *if* $y_i < 0)$;
13 **return** w

Algorithm 2: Online PA algorithm for binary classification with multiple views latent structures.

ming to find the best chunking for each representation. The features of $\phi(x, z)$ are 1-, 2-, 3-grams of words and POS tags paired with the chunk tags thus found, as well as bigrams of chunk tags. Such features are taken across the chunkings for each representation. We use entirely separate chunk tags for each representation. E.g., although each representation uses an "O" tag to indicate a word outside of any phrase, we consider the "O" for each representation to be distinct. Like the single-view version, this model can optionally be initialized by using weights obtained by training for the standard phrase-identification chunking task. The algorithm is given in algorithm 2. In the following sections we describe some different approaches toward using the multiple representations generated by the different views by modifying the combine step on line 7.

3.4.1 Method 1 - Adding Representations

In our first method, we simply extract the features used in each of the different representations and combine the features for each example x_i. The resulting z^*_i is a tuple of chunking structures, and $\phi(x_i, z_i)$ is a feature vector across all representations, which is then used in the weight vector update. Since the features generated by each representation's inference are distinct to that representation, they are shared in updating the model, but not in inference. Therefore the features contributed by each example are distinct, and $\phi(x_i, z_i)$ can be considered the sum of separate feature vectors for each representation r: $\sum_r \phi(x, z_{i_r})$.

3.4.2 Method 2 - Majority Voting

In the majority voting method, we combine the output of the views by converting the structures

1 COMBINE$(x_i, \{z^*_{i_r} : r\})$:
2 **for** *all* $r \in R$ **do**
3 | convert $z^*_{i_r}$ to common representation Z_{i_r}
4 get consensus structure $\hat{Z}_i$ by voting between $\{Z^*_{i_r} : r\}$
5 **for** *all* $r \in R$ **do**
6 | convert $\hat{Z}_i$ into view-specific representation $\hat{z}^*_{i_r}$
7 **return** $\phi(x_i, z_i) = \sum_r \phi(x_i, \hat{z}_{i_r})$

Algorithm 3: Majority voting method of combining multiple views.

1 COMBINE$(x_i, \{z^*_{i_r} : r\})$:
2 **for** *all* $r \in R$ **do**
3 | **for** *each representation* $s \neq r$ **do**
4 | | convert z_{s_i} to common representation $Z_{i_{rs}}$
5 | get consensus structure $\hat{Z}_{i_r}$ by voting between all $\{Z_{i_{rs}} : s \neq r\}$
6 | convert $\hat{Z}_{i_r}$ into view-specific representation $\hat{z}^*_{i_r}$
7 **return** $\phi(x_i, z_i) = \sum_r \phi(x_i, \hat{z}_{i_r})$

Algorithm 4: Co-training based method of combining multiple views.

found under each representation to a single common representation, from which we then pick the consensus structure and update the model towards that. However, the consensus structure is first converted back to each of the individual representations for the weight vector update step so that we update weights pertaining to features for each of the representational forms.

3.4.3 Method 3 - Co-training

The final method we examined for combining the multiple views was inspired by co-training (Blum and Mitchell, 1998). In co-training, the predictions of a weak classifier are bolstered by information supplied by alternative views on the data. Intuitively, we surmise that allowing the alternative views to guide each single view may act as a form of regularization while still retaining the information from the different experts. Our method is similar to co-training in that we allow each view to be informed by the output of the other view. However, it does not obey the strict condition on co-training that the feature sets from each view should be conditionally independent from each other.

In this method, for each training example, we again begin by converting each of the views' best latent structure to a common representation. Then, for each single view $r_i \in r = 1..R$, the best structure for r_i is selected by voting from the other $R - 1$ views. That consensus structure is converted into r's representation and added to the feature vector for all views for that example and the weights are updated accordingly.

4 Experiments

We trained several settings of our chunker-classifier as well as the batch baseline classifier, as we detail in section 4.1. We then evaluated the resulting trained models on the grammaticality task.

4.1 Training

For the chunkers we used the CONLL 2000 tagset (consisting of 23 constituent chunk tags), modified for the five chunking representations of (Shen and Sarkar, 2005).

For training data we used the English side of the HK Chinese-English parallel corpus. For positive examples, we used 50,000 sentences taken directly from the English side. For negative examples we used the pseudo-negative approach of Okanohara and Tsujii (2007): we trained a standard 3-gram language model on the 50,000 sentences plus 450,000 additional sentences from the same corpus. From this we sampled 50,000 sentences to create the negative training data set.

This choice of training data is similar to Cherry and Quirk (2008). However, they do not include the positive examples in the sentences used to train the n-gram LM for pseudo-negative generation, and they do not filter the result to use only words from the positive data. We made these choices so as to make the positive and negative training examples more similar in terms of the words used. If the positive and negative examples have substantially different vocabulary, it artifically makes the task much simpler since it makes it possible for the classifier to avoid using the latent features; a basic 1-gram model can distinguish between positives and negatives based on vocabulary alone with high accuracy, but this situation does not approximate the kind of fluency decisions that we are interested in.

The chunker-classifier can either be started with a zero weight vector or with weights from training on the chunking task itself; we tried both initializations. For the latter option we used weights from supervised discriminative training against gold-standard chunking. This supervised training used the same features as the chunker-classifier with the CONLL 2000 chunking tagset, which Shen and Sarkar (2005) refer to as the IOB2 representation. To transfer the weights to the classifier, we scaled them to the range of weight values observed after training the zero-initialized chunker-classifier, approximately $[-0.1, 0.1]$ with a single representation and $[-0.01, 0.01]$ with all five representations. This gives non-zero initial weights for some of the features corresponding to the IOB2 representation, but not for any of the other representations.

initialize $\mathbf{w}$
for $t = 0, ..., T-1$ **do**
  repeat
    for *each training example* x_i *in* X **do**
      find $z_i^* = \arg\max_{z_i} \mathbf{w} \cdot \phi(x_i, z_i)$
    for *each training example* x_i *in* X **do**
      repeat
        let $y_i^* = \mathbf{w} \cdot \phi(x_i, z_i^*)$
        let loss $l = \max\{0, 1 - y_i y_i^*)\}$
        let multiplier $\tau = \frac{l}{\|\phi(x_i, z_i^*)\|^2}$
        update $\mathbf{w} := \mathbf{w} + \tau y_i \phi(x_i, z_i^*)$
      until $y_i > 0$ *or* ($y_i^* = y_i$ *if* $y_i < 0$);
  until *convergence*;
return $\mathbf{w}_T$

Algorithm 5: Batch-variant PA algorithm for binary classification with latent structures.

4.2 Batch Baseline

We implemented two batch baselines. The first is a strict implementation of the LCLR algorithm as it appears in Chang et al. (2010), with per-outer-iteration example caching (LCLR). The only difference is that we use a Passive-Aggressive large-margin classifier instead of an SVM. In (Chang et al., 2010), the SVM was trained using a coordinate descent algorithm on the dual (Hsieh et al., 2008), while we train our PA algorithm as we specify earlier. This allows us to consistently use the same learning algorithm to compare between batch and online learning with latent variables. However, we found that in practice, this algorithm severely overfits to our application task. The inner loop that repeatedly performs batch updates to the negative examples swayed the model such that while training error approached zero, it disproportionately classified the test data sets as negative. So, we also implemented a variant that skips the inference step in the inner loop. This variant treated the latent structures found in the inference step of the outer loop as fixed, but relabeled and updated accordingly until convergence, where it would resume the next outer iteration. Since this variant was competitive with the online algorithms, we present its results as LCLR-variant in Sections 4.3 and 4.4. The algorithm is given in Alg. 5.

4.3 Evaluation

One way to evaluate discriminative LMs is on the intrinsic classification of distinguishing real gram-

Model	Classification Accuracy
LCLR	90.27
LCLR-variant	94.55
online 1-view	98.75
online multi-1	98.70
online multi-2	98.78
online multi-3	98.85

Table 2: Classification accuracy (percent) after 40 outer iterations. Multi-1 refers to the additive method of combining representations; Multi-2 refers to majority voting, and Multi-3 refers to co-training.

matical sentences from generated (and presumably ungrammatical) pseudo-negative sentences.

As test data for this task we used the Xinhua data from the English Gigaword corpus. We used the first 3000 sentences as positive examples. For negative examples we trained a 3-gram LM on the first 500,000 examples (including those used for positive data), with no filtering. We used this 3-gram LM to generate five separate 3000 example negative data sets. To account for random variation due to using pseudo-negatives results are reported as a mean over the positive data paired with each negative set.

The results are summarized in Table 2.

4.4 Analysis

As shown in Table 2, all online algorithms outperform the batch versions. We also see a slight increase in classification accuracy with some of the multi-view approaches, in particular the co-training based approach. Improved performance of methods 2 and 3 for combining over method 1 might be expected. Since the motivation for using multiple representations is that the model might be guided towards those features that are most helpful to the classification problem, intuitively we can suppose that this is more likely to occur when the views are being driven towards agreement, which is not enforced under method 1. The co-training based method in particular is well-positioned to take advantage of having multiple views on the latent structure. This approach encourages agreement between the views, but by retaining multiple consensus structures, it can capitalize on the different kinds of expertise of the various views, which may be lost in a simple majority voting approach.

The sentence shown in Table 3 is taken from the positive examples in the testset; it illustrates the advantage that came from having multiple views

B	B	(B	I)	B	B
hong	*kong*	*is*	*a*	*conspicuous*	*example*
B	B	B	B	(B	I)

Table 3: Output chunking on a sample phrase *"hong kong is a conspicuous example"* from the single-view model (on top) and the co-train based multi-view model (at the bottom). The chunking shown is IOB2 for both models.

on the latent structures. This example was correctly classified by the co-train based model but incorrectly classified by the single-view model. As the latent structure prediction was optimized to distinguish between real and synthetic data (rather than to match a gold-standard chunking), neither chunking reflects the linguistically accurate one. However, we see that there were different groupings between the two models: the single view groups "is a" together in a chunk constituent, whereas the multi-view groups together "conspicuous example". While "is" and "a" may frequently co-occur, "conspicuous" and "example" may together form a more coherent constituent.

Figure 1 shows the per-iteration test accuracy of the different models. The batch versions are much slower to improve their testset accuracies. Though LCLR-variant attains more competitive accuracy with the online version than LCLR, it seems to be overfitting past the 35th iteration. The online versions, on the other hand, learn quickly and maintain high accuracies. While all online models eventually reach high accuracies, the voting and co-train multiple-views models reach a high accuracy much faster than the simpler models. In particular, the co-training based method reaches an accuracy over 98% by the fifth iteration; for the single-view and the simple multi-view combine methods, it takes at least twice as many iterations. This indicates that co-training based method using mutiple views on the latent representations gives an advantage when training on a budget. We surmise that this is because the enforced consenses between the multiple views finds latent structures that are helpful to the classification task much more quickly.

In addition to reaching high accuracy on test data quickly, we also find that in practice, the online algorithm requires fewer updates total in training than the batch version. In the online algorithm, each negative example is re-decoded until it is correctly labeled, whereas in LCLR, all negative examples must be re-decoded each time for any single incorrectly labeled example. We found that

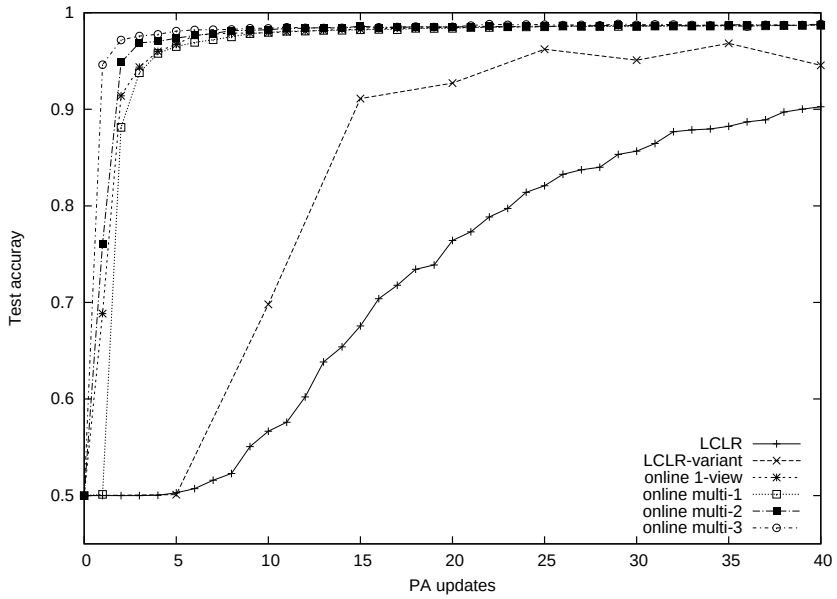

Figure 1: Per-iteration test accuracies up to 40 (outer) iterations for the classifiers with weight-vector initialization.

for both models, incorrectly labeled examples usually only needed to be re-decoded once to receive a correct label. Because there are generally only a small number of "difficult" examples that need to be re-decoded in each iteration, this means that the batch algorithm is forced to repeat the inference step and perform increased computation on the order of the size of the negative training set.

5 Related Work

Our work is most similar to Chang et al. (2010), which proposes a batch algorithm for learning over latent representations. We expand upon their framework by developing an efficient online algorithm and exploring learning over multiple views on the latent representations. However, there are several works that have explored a similar task to the one we report on. Max-margin LMs for speech recognition focus on the word prediction task (Gao et al., 2005; Roark et al., 2007; Singh-Miller and Collins, 2007). This focus is also shared by other syntactic LMs (Chelba and Jelinek, 1998; Xu et al., 2002; Schwartz et al., 2011) which use syntax but rely on supervised data to train their parsers and then decode for LMs from left to right but also bottom-up (Charniak, 2001). Charniak et al. (2003) and Shen et al. (2010) use parsing based LMs for machine translation but LM integration into the decoder limits the number of features used compared to a full-sentence discriminative LM. Our focus is on fully exploiting the latent variables and training whole-sentence discriminative LMs. Our chunker model is related to the semi-Markov model described by Okanohara and Tsujii (2007), but ours can take advantage of latent structures. Our work is related to Cherry and Quirk (2008) but differs in ways previously described.

6 Conclusion and Future Work

In future work, we plan to apply our algorithms to more tasks to corroborate its effectiveness in other areas. We would also like to undertake more thorough analysis of the properties of online learning algorithms over latent structures. For multiple views in latent structure learning, we have explored separate inference for each representations' view; in the future, we will examine the use of joint inference across multiple latent representations.

We have presented a general online algorithm for learning over constrained latent representations, and we have shown that it attains higher accuracy on a binary grammaticality test. We have also explored the use of multiple views on latent representations via concatenation, majority voting, and co-training. This type of training can be applied to various NLP tasks that stand to benefit from latent structure information.

References

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, COLT' 98, pages 92–100, New York, NY, USA. ACM.

Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Yuancheng Tu. 2009. Unsupervised constraint driven learning for transliteration discovery. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 299–307, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Vivek Srikumar. 2010. Discriminative learning over constrained latent representations. In *HLT-NAACL*, pages 429–437.

Eugene Charniak, Kevin Knight, and Kenji Yamada. 2003. Syntax-based Language Models for Machine Translation. In *Proc. of MT Summit IX*.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proc. of ACL 2001*, pages 124–131, Toulouse, France, July. Association for Computational Linguistics.

Ciprian Chelba and Frederick Jelinek. 1998. Exploiting syntactic structure for language modeling. In *Proc. of ACL 1998*, pages 225–231, Montreal, Quebec, Canada, August. Association for Computational Linguistics.

Colin Cherry and Chris Quirk. 2008. Discriminative, syntactic language modeling through latent SVMs. In *Proc. of AMTA 2008*.

Koby Crammer and Yoram Singer. 2001. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, January.

Dipanjan Das and Noah A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 468–476, Stroudsburg, PA, USA. Association for Computational Linguistics.

Pedro Felzenszwalb, Ross Girshick, and David McAllester. 2010. Cascade object detection with deformable part models. In *Proc. of CVPR 2010*, pages 2241–2248.

Jianfeng Gao, Hao Yu, Wei Yuan, and Peng Xu. 2005. Minimum sample risk methods for language modeling. In *Proc. of ACL*, pages 209–216, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Dan Goldwasser and Dan Roth. 2008. Transliteration as constrained optimization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 353–362, Stroudsburg, PA, USA. Association for Computational Linguistics.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. 2008. A dual coordinate descent method for large-scale linear svm. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *ICML*, volume 307 of *ACM International Conference Proceeding Series*, pages 408–415. ACM.

Thorsten Joachims and Chun-Nam John Yu. 2009. Sparse kernel svms via cutting-plane training. *Machine Learning*, 76(2-3):179–193.

Andrew Mccallum and Kedar Bellare. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. In *In Conference on Uncertainty in AI (UAI*.

Daisuke Okanohara and Jun'ichi Tsujii. 2007. A discriminative language model with pseudo-negative samples. In *Proc. of ACL 2007*, pages 73–80, Prague, Czech Republic, June. Association for Computational Linguistics.

Brian Roark, Murat Saraclar, and Michael Collins. 2007. Discriminative n-gram language modeling. *Computer Speech and Language*, 21(2):373–392.

Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, EACL '99, pages 173–179, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lane Schwartz, Chris Callison-Burch, William Schuler, and Stephen Wu. 2011. Incremental syntactic language models for phrase-based translation. In *Proc. of ACL-HLT 2011*, pages 620–631, Portland, Oregon, USA, June. Association for Computational Linguistics.

Hong Shen and Anoop Sarkar. 2005. Voting between multiple data representations for text chunking. In *Canadian Conference on AI*, pages 389–400.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2010. String-to-dependency statistical machine translation. *Comput. Linguist.*, 36(4):649–671.

Natasha Singh-Miller and Michael Collins. 2007. Trigger-based Language Modeling using a Loss-sensitive Perceptron Algorithm. In *Proc. of ICASSP 2007*.

Peng Xu, Ciprian Chelba, and Frederick Jelinek. 2002. A study on richer syntactic dependencies for structured language modeling. In *Proc. of ACL 2002*, pages 191–198, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.