Training Structured Prediction Models with Extrinsic Loss Functions

Keith Hall **Ryan McDonald** **Slav Petrov**
Google, New York
{kbhall,ryanmcd,slav}@google.com

Abstract

We present an online learning algorithm for training structured prediction models with extrinsic loss functions. This allows us to extend a standard supervised learning objective with additional loss-functions, either based on intrinsic or task-specific extrinsic measures of quality. We present experiments with sequence models on part-of-speech tagging and named entity recognition tasks, and with syntactic parsers on dependency parsing and machine translation reordering tasks.

1 Introduction

It is well known that the performance of prediction models suffers when there is a mismatch between the training and test domains. This is especially the case in Natural Language Processing, where the labeled training data is often decades old, while the models are typically applied to recently generated webtext.

In this paper we extend a standard supervised learning objective with additional loss-functions. We make no assumptions about the loss-functions. In particular, we do not need them to decompose over the model structure or factor in any specific way. Additionally, each loss-function can be associated with a distinct dataset. Our *augmented-loss* training framework (Section 2) simply iterates over the various loss-functions and training examples in round-robin fashion, performing perceptron-style updates: if the model prediction is already optimal, no update is performed; however, if there is a better labeling, then the model parameters are updated. We describe how this update can be performed when the optimal loss is unknown and sketch a convergence proof for the separable case.

The additional loss-functions can provide a signal for adapting to new domains or to specific tasks, or both. For example, it might be too expensive to annotate sentences from a new domain with part-of-speech (POS) tags or full syntactic parse trees. However, it might be feasible to obtain partial annotations, for example specifying the main verb of the sentence. We show in our experiments in Section 3 that even this single bit of information can provide a big boost in performance, closing more than half of the gap between in-domain and out-of-domain POS tagging and parsing performance. We also present an experiment where a syntactic parser is tuned specifically for the task of machine translation reordering. Using an extrinsic loss (that does not decompose over the model structure), we are able to obtain significant improvements in machine translation reodering score, as well as downstream machine translation quality.

Our approach is similar to the perceptron-based learning of phrase-translation parameters presented in [1]. However, their main goal is to incorporate additional features into the model, whereas we are interested in adapting the existing model parameters. Constraint Driven Learning [2] optimizes a loss function with the addition of constraints based on unlabeled data. The augmented-loss algorithm can be viewed as an online version of this algorithm. Direct Loss Minimization [3] is also highly related, however, we jointly optimize multiple loss functions and do not make any assumptions about the decomposability of the loss. Additionally, our algorithm can be viewed as an instance of Sample Rank [4] extended to multiple loss functions, both intrinsic and task-specific.

2 Methodology

The augmented-loss algorithm [5] is a general mechanism for incorporating multiple loss functions in online learning. We review it here in the context of the structured perceptron [6]. The structured perceptron is an online learning algorithm which takes as input: (1) a set of training examples $d_i = (x_i, y_i)$ consisting of input sentences x_i and an outputs y_i; and (2) a loss-function, $L(\hat{y}, y)$, that measures the cost of predicting output $\hat{y}$ relative to the gold standard y, typically the 0/1 loss. Learning proceeds by predicting a structured output $\hat{y}$ given the current model parameters θ: $\hat{y} = F_\theta(x) = \arg\max_{y \in \mathcal{Y}_x} \theta \cdot \Phi(y, x)$, where Φ is an application specific mapping from a structured output y for sentence x to a high dimensional feature space. If the predicted structure is incorrect ($L(\hat{y}, y) > 0$) the model is updated by rewarding features that fire in the gold-standard $\Phi(y, x)$, and discounting features that fire only in the predicted output, $\Phi(\hat{y}, x)$,

2.1 Augmented-Loss Training

Augmented-loss training (see the Appendix for pseudocode) extends the structured perceptron by allowing for (1) multiple datasets $D^1, \ldots, \mathcal{D}^M$; (2) multiple loss functions $L^1, \ldots, L^M$ which are associated with these datasets; and (3) a schedule for processing examples from each of these datasets. Note that the label sets for the different datasets will often be different, or can even be empty. The training procedure is strictly guided by the loss L^j, which can be any task-specific metric. The algorithm is effectively the same as the perceptron, the primary difference being that if L^j is an extrinsic loss, we cannot compute the standard updates since we do not necessarily know the correct output. The algorithm iterates over the training examples (and loss functions) according to the schedule.

2.2 Inline Ranker Training

We can use inline ranker training [5] when the loss L^j is not a standard supervised loss and we do not have access to the correct output. For ease of exposition, we will assume that we have access to a cost function and that the loss is computed as the difference in cost between two different outputs. This formulation is general and does not restrict our choice of loss functions, in particular it encompasses losses that do not decompose according to the model structure. If we could enumerate all outputs (e.g. in atomic classification), then we could use the cost function to determine the optimal output. For structured prediction problems, however, the output space is exponential. We therefore restrict ourselves to searching over a candidate set of output structures. In practice we use a ranked k-best list, but any type of samples from the output space could be used instead [4]. $F_\theta^{\text{k-best}}(x_i) = \{\hat{y}_i^1, \ldots, \hat{y}_i^k\}$. If the lowest-cost output in $F_\theta^{\text{k-best}}(x_i)$ is not the 1-best, then $F_\theta^{\text{k-best}}(x_i)$ is taken to be the *correct* output structure y_i, and the 1-best output is taken to be an incorrect prediction, and we take a perceptron step. If on the other hand the 1-best parse has the lowest cost, then our current model is assumed to be optimal and we move to the next training example according to the schedule.

Similarly to the regular perceptron, we only perform updates when there is an error – when the 1-best output has a higher cost than any other output in the k-best list. The intuition behind this method is that in the presence of only a cost function and a k-best list, the parameters will be updated towards the output structure that has the lowest cost, which over time will move the parameters of the model to a place with low extrinsic loss. An advantage of this approach is that the scoring function does not need to be factored, requiring no internal knowledge of the function itself. Furthermore, this approach can be applied to any structured prediction algorithm which can generate k-best lists.

2.3 Convergence

Assuming that the training set is loss-separable, one can show that augmented-loss training will converge [5]. The convergence proof is similar to the original perceptron proof and does not assume anything about the loss. In particular, every instance (x_i, y_i) could use a different loss. It is only required that the loss for a specific input-output pair is fixed throughout training. However, the proof does make the assumption that for any θ that exists during training, but before convergence, there is at least one example in the training data where the k-best list is large enough to include one output with a lower loss when $\hat{y}_1$ does not have the optimal minimal loss. In practice, this seems not to be a problem as we will see in the experiments presented in the next section.

Question Tagging	Accuracy
PTB supervised	89.77
QTB supervised	93.63
augmented-loss	91.92

Question Parsing	LAS	UAS	Root-F1
PTB supervised	67.97	73.52	47.60
QTB supervised	84.59	89.59	91.06
augmented-loss	76.27	86.42	83.41

Table 1: Augmented-loss training can be used to adapt POS taggers and parsers to new domains.

3 Experiments

In our experiments we use augmented-loss training (1) for adapting POS taggers and parsers trained on newswire to a question domain and (2) for task specific adaptation of POS taggers used in named entity recognition (NER) systems and of parsers used in a machine translation reordering systems.

3.1 Models & Datasets

The augmented-loss framework can be used with any structured prediction model that can be trained with the perceptron. We use the following models in our experiments:

Part-of-Speech Tagger: A linear chain Conditional Random Field (CRF) [7] with prefix, suffix and word cluster based features. The word clusters are generated on a large unlabeled newswire corpus.

Named Entity Tagger: A linear chain CRF that uses POS tags in addition to prefix, suffix, capitalization and word cluster features. The same clusters as in the POS tagger are used.

Syntactic Parser: An implementation of the transition-based dependency parsing framework [8] using an arc-eager transition strategy and trained using the perceptron algorithm as in [9]. Beams with varying sizes can be used to produce k-best lists.

We utilize a few different datasets in our experiments. Depending on the experiment, we will use either the full annotation or derive a weaker signal for our additional loss functions.

Treebank Data: The Penn Wall Street Journal Treebank (PTB) [10], the Brown corpus, and the Question Treebank (QTB) [11], provide labeled data for our part-of-speech tagging and parsing experiments. We convert the treebanks to dependency format using the Stanford converter [12].

Named Entity Data: The 2003 CoNLL shared task on Named Entity Recognition [13] provides an English dataset with labels for four different types of named entities.

Reordering Data: The dataset of Talbot et al. [14] provides word-level alignments between English and Japanese sentences that can be used in for adapting parsers for a machine translation task.

3.2 Semi-Supervised Domain Adaptation

One of the main applications of the augmented-loss framework is to improve the domain portability of structured prediction systems in the presence of partially labeled data. Consider, for example, the case of questions. It has been observed that part-of-speech taggers and dependency parsers tend to do quite poorly on questions due to their limited occurence in the newswire training data [15, 16]. Table 1 shows that models trained on the PTB perform much worse on the QTB test data, than models trained on the QTB, even though the PTB training set is 20 times larger.

Because of the difference in word order between declarative sentences and questions, the models often cannot even determine the main verb of questions. We therefore consider a scenario where it is possible to ask annotators to determine the main verb of a given question. While full part-of-speech and syntactic parse tree annotation requires extensive linguistic training, such a question is easy to answer for anybody speaking the language. In practice, we used the QTB training set stripped of all annotations except the label of the main verb for each sentence. Our augmented-loss function in this case is a simple binary function: 0 if the model prediction (tag sequence or parse tree) has the correct main verb and 1 if it does not. Thus, the algorithm will select the first prediction in the k-best list that has the correct main verb as a proxy to the gold standard labeling. The last row in Table 1 shows that by having the main verb annotated in each sentence and iterating between the

Named Entity Recognition	F1
supervised	84.10
augmented-loss-1	84.51
augmented-loss-2	84.87

MT Reordering	Exact	Reorder
PTB + Brown + QTB	35.29	76.49
$1.0\times$augmented-loss	39.02	78.39
$2.0\times$augmented-loss	39.58	78.67

Table 2: Augmented-loss training can be used to adapt structured models to specific tasks.

supervised objective and the augmented loss objective, half of the errors of the original model can be eliminated. It is important to point out that these improvements are not limited to simply better main verb predictions. Due to the fact that the structured prediction models make many decisions jointly, these decisions influence each other and improvements are seen across the board.

3.3 Named Entity Recognition

Most named entity recognition systems rely heavily on features derived from a part-of-speech tagger. In such a system the accuracy of the POS tagger is only indirectly relevant – only as much as it helps NER accuracy. Since the training sets of the POS tagger and NER system are disjoint, we can use augmented-loss training to adapt the POS tagger to the training domain of the NER system and to also tune it specifically for being used in an NER system. Because named entities should be labeled with the POS tag "NNP," we can define a loss function that penalizes the tagger for not obeying the named entity annotations. We experimented with two loss functions. The first one is simply the hamming loss relative to the entity annotation, while the second one also incorporates a term based on the NER F1-score. As Table 2 shows, both lead to improvements over the non-adapted baseline.

3.4 Machine Translation Reordering

Many statistical machine translation systems use a source-side reordering component, which reorders the sentence into target-side word order before translating it [17]. The reordering component typically uses the syntactic parse of the source sentence and a set of rules to perform the reordering. While better parsing accuracies tend to lead to better reordering scores, it is clear that certain parsing mistakes will matter more than others. We can use a reordering-based loss function to improve a parser used in a reordering component of a machine translation system. In our experiments we parse the input sentence with a statistical dependency parser and then apply a set of hand-written English-Japanese reordering rules [18]. A set of human generated golden reorderings for aligned target sentences is then used to compute a reordering score [14] which measures what fraction of words are in the correct order, similarly to the METEOR scoring metric.

As a baseline, we use a parser trained on the training portions of PTB, Brown, and QTB. For augmented-loss training, we add extrinsic reordering training data consisting of 10K examples of English sentences and their correct Japanese word-order, and use the negative of the reordering score as extrinsic loss. Evaluating on a set of 6338 examples of similarly created reordering data, we observe improvements as we adjust the schedule to process the extrinsic loss more frequently. The best result in Table 2 is achieved when we make two augmented-loss updates for every treebank-based loss update.

4 Conclusions

We presented experiments with the augmented-loss training algorithm [5] and showed that it can be used to incorporate multiple loss functions. This allows us to adapt structured prediction models to new domains or specific tasks. The augmented-loss framework supports both intrinsic and extrinsic losses, allowing for both combinations of objectives. This flexibility makes it possible to tune a model for a downstream task. The only requirement is a metric which can be defined over outputs of the downstream application.

References

[1] P. Liang, A. Bouchard-Cote, D. Klein, and B. Taskar. An end-to-end discriminative approach to machine translation. In *Proc. of COLING/ACL*, 2006.

[2] M.W. Chang, L. Ratinov, and D. Roth. Guiding semi-supervision with constraint-driven learning. In *Proc. of ACL*, 2007.

[3] D. McAllester, T. Hazan, and J. Keshet. Direct loss minimization for structured prediction. In *Proc. of NIPS*, 2010.

[4] M. Wick, K. Rohanimanesh, K. Bellare, A. Culotta, and A. McCallum. Samplerank: Training factor graphs with atomic gradients. In *Proc. of ICML*, 2011.

[5] K. Hall, R. McDonald, J. Katz-Brown, and M. Ringgaard. Training dependency parsers by jointly optimizing multiple objectives. In *Proc. of EMNLP*, 2011.

[6] M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. of ACL*, 2002.

[7] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*, 2001.

[8] J. Nivre. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553, 2008.

[9] Y. Zhang and S. Clark. A Tale of Two Parsers: Investigating and Combining Graph-based and Transition-based Dependency Parsing. In *Proc. of EMNLP*, pages 562–571, 2008.

[10] M. Marcus, B. Santorini, and M.A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330, 1993.

[11] J. Judge, A. Cahill, and J. Van Genabith. Question-bank: Creating a corpus of parse-annotated questions. In *Proc. of ACL*, pages 497–504, 2006.

[12] M.C. de Marneffe, B. MacCartney, and C. Manning. Generating typed dependency parses from phrase structure parses. In *Proc. of LREC*, Genoa, Italy, 2006.

[13] E. F. Tjong Kim Sang and F. De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proc of CoNLL*, pages 142–147, 2003.

[14] D. Talbot, H. Kazawa, H. Ichikawa, J. Katz-Brown, M. Seno, and F. Och. A lightweight evaluation framework for machine translation reordering. In *Proc. of the Sixth Workshop on Statistical Machine Translation*, 2011.

[15] A. Subramanya, S. Petrov, and F. Pereira. Efficient graph-based semi-supervised learning of structured tagging models. In *Proc. of EMNLP*, 2010.

[16] S. Petrov, P.C. Chang, M. Ringgaard, and H. Alshawi. Uptraining for accurate deterministic question parsing. In *Proc. of EMNLP*, pages 705–713, 2010.

[17] M. Collins, P. Koehn, and I. Kučerová. Clause restructuring for statistical machine translation. In *Proc. of ACL*, 2005.

[18] P. Xu, J. Kang, M. Ringgaard, and F. Och. Using a dependency parser to improve SMT for Subject-Object-Verb languages. In *Proc. of NAACL*, 2009.

Appendix

Algorithm 1 Augmented-Loss Perceptron

1: {Input data sets}:
2: $\mathcal{D}^1 = \{d_1^1 = (x_1^1, y_1^1) \dots d_{N^1}^1 = (x_{N^1}^1, y_{N^1}^1)\}$,
3: ...
4: $\mathcal{D}^M = \{d_1^M = (x_1^M, y_1^M) \dots d_{N^M}^M = (x_{N^M}^M, y_{N^M}^M)\}$
5: {Input loss functions: $L^1 \dots L^M$}
6: {Initialize indexes: $c_1 \dots c_M = \vec{0}$}
7: {Initialize model parameters: $\theta = \vec{0}$}
8: $i = 0$
9: **repeat**
10: **for** $j = 1 \dots M$ **do**
11: {Check whether to update L^j on iteration i}
12: **if** Sched(j, i) **then**
13: {Compute index of instance – reset if $c_j \equiv N^j$}
14: $c_j = [(c_j \equiv N^j)\ ?\ 0\ :\ c_j + 1]$
15: {Compute structured loss for instance}
16: **if** L^j is intrinsic loss **then**
17: $\hat{y} = F_\theta(x_{c_j}^j)$
18: **if** $L^j(\hat{y}, y_{c_j}^j) > 0$ **then**
19: $\theta = \theta + \Phi(y_{c_j}^j) - \Phi(\hat{y})$ {$y_{c_j}^j$ is a tree}
20: **end if**
21: **else if** L^j is an extrinsic loss **then**
22: $\{\hat{y}_1, \dots, \hat{y}_k\} = F_\theta^{\text{k-best}}(x_i)$
23: $\tau = \min_\tau C(x_{c_j}^j, \hat{y}_\tau, y_{c_j}^j)$ {τ is m in const index}
24: $L^j(\hat{y}_1, y_{c_j}^j) = C(x_{c_j}^j, \hat{y}_1, y_{c_j}^j) - C(x_{c_j}^j, \hat{y}_\tau, y_{c_j}^j)$
25: **if** $L^j(\hat{y}_1, y_{c_j}^j) > 0$ **then**
26: $\theta = \theta + \Phi(\hat{y}_\tau) - \Phi(\hat{y}_1)$
27: **end if**
28: **end if**
29: **end if**
30: **end for**
31: $i = i + 1$
32: **until** converged
33: {Return model θ}

Algorithm 1 presents pseudo-code for the augmented-loss structured perceptron algorithm. The algorithm is an extension of the structured perceptron, but where there are (1) multiple loss functions being evaluated $L^1, \dots, L^M$; (2) there are multiple datasets associated with each of these loss functions $D^1, \dots, \mathcal{D}^M$; and (3) there is a schedule for processing examples from each of these datasets, where Sched(j, i) is true if the j^{th} loss function should be updated on the i^{th} iteration of training. Note that for data point $d_i^j = (x, y)$, which is the i^{th} training instance of the j^{th} data set, that y does not necessarily have come from the output space of the intrinsic dataset. It can either be a task-specific output of interest or even null, in the case where learning will be guided strictly by the loss L^j. The training algorithm is effectively the same as the perceptron, the primary difference is that if L^j is an extrinsic loss, we cannot compute the standard updates since we do not necessarily know the correct output.

The inline reranker (appearing at line 21) uses the currently trained model parameters θ to process the external input, producing a k-best set of outputs associated with the output-space of the intrinsic data: $F_\theta^{\text{k-best}}(x_i) = \{\hat{y}_1, \dots, \hat{y}_k\}$. We can compute the cost $C(x_i, \hat{y}, y_i)$ for all $\hat{y} \in F_\theta^{\text{k-best}}(x_i)$. If the 1-best parse, $\hat{y}_1$ has the lowest cost, then there is no need to update the model parameters. Otherwise, the lowest-cost output in $F_\theta^{\text{k-best}}(x_i)$ is taken to be the *correct* output structure y_i, and the 1-best output is taken to be an incorrect prediction.