Distilling an Ensemble of Greedy Dependency Parsers into One MST Parser

Adhiguna Kuncoro[♠] **Miguel Ballesteros**[♢] **Lingpeng Kong**[♠]
Chris Dyer[♠♣] **Noah A. Smith**[♡]
[♠]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
[♢]NLP Group, Pompeu Fabra University, Barcelona, Spain
[♣]Google DeepMind, London, UK
[♡]Computer Science & Engineering, University of Washington, Seattle, WA, USA
{akuncoro,cdyer,lingpenk}@cs.cmu.edu
miguel.ballesteros@upf.edu, nasmith@cs.washington.edu

Abstract

We introduce two first-order graph-based dependency parsers achieving a new state of the art. The first is a consensus parser built from an ensemble of independently trained greedy LSTM transition-based parsers with different random initializations. We cast this approach as minimum Bayes risk decoding (under the Hamming cost) and argue that weaker consensus within the ensemble is a useful signal of difficulty or ambiguity. The second parser is a "distillation" of the ensemble into a single model. We train the distillation parser using a structured hinge loss objective with a novel cost that incorporates ensemble uncertainty estimates for each possible attachment, thereby avoiding the intractable cross-entropy computations required by applying standard distillation objectives to problems with structured outputs. The first-order distillation parser matches or surpasses the state of the art on English, Chinese, and German.

1 Introduction

Neural network dependency parsers achieve state of the art performance (Dyer et al., 2015; Weiss et al., 2015; Andor et al., 2016), but training them involves gradient descent on non-convex objectives, which is unstable with respect to initial parameter values. For some tasks, an **ensemble** of neural networks from different random initializations has been found to improve performance over individual models (Sutskever et al., 2014; Vinyals et al., 2015, *inter alia*). In §3, we apply this idea to build a first-order graph-based (FOG) ensemble parser (Sagae and Lavie, 2006) that seeks consensus among 20 randomly-initialized stack LSTM parsers (Dyer et al., 2015), achieving nearly the best-reported performance on the standard Penn Treebank Stanford dependencies task (94.51 UAS, 92.70 LAS).

We give a probabilistic interpretation to the ensemble parser (with a minor modification), viewing it as an instance of **minimum Bayes risk** inference. We propose that disagreements among the ensemble's members may be taken as a signal that an attachment decision is difficult or ambiguous.

Ensemble parsing is not a practical solution, however, since an ensemble of N parsers requires N times as much computation, plus the runtime of finding consensus. We address this issue in §5 by **distilling** the ensemble into a **single** FOG parser with discriminative training by defining a new *cost function*, inspired by the notion of "soft targets" (Hinton et al., 2015). The essential idea is to derive the cost of each possible attachment from the ensemble's division of votes, and use this cost in discriminative learning. The application of distilliation to structured prediction is, to our knowledge, new, as is the idea of empirically estimating cost functions.

The distilled model performs almost as well as the ensemble consensus and much better than (i) a strong LSTM FOG parser trained using the conventional Hamming cost function, (ii) recently published strong LSTM FOG parsers (Kiperwasser and Goldberg, 2016; Wang and Chang, 2016), and (iii) many higher-order graph-based parsers (Koo and Collins, 2010; Martins et al., 2013; Le and Zuidema, 2014). It represents a new state of the art for graph-based dependency parsing for English, Chinese, and

German. The code to reproduce our results is publicly available.[1]

2 Notation and Definitions

Let $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$ denote an n-length sentence. A dependency parse for $\boldsymbol{x}$, denoted $\boldsymbol{y}$, is a set of tuples (h, m, ℓ), where h is the index of a head, m the index of a modifier, and ℓ a dependency label (or relation type). Most dependency parsers are constrained to return $\boldsymbol{y}$ that form a directed tree.

A first-order graph-based (**FOG**; also known as "arc-factored") dependency parser exactly solves

$$\hat{\boldsymbol{y}}(\boldsymbol{x}) = \arg \max_{\boldsymbol{y} \in \mathcal{T}(\boldsymbol{x})} \underbrace{\sum_{(h,m) \in \boldsymbol{y}} s(h, m, \boldsymbol{x})}_{S(\boldsymbol{y}, \boldsymbol{x})}, \tag{1}$$

where $\mathcal{T}(\boldsymbol{x})$ is the set of directed trees over $\boldsymbol{x}$, and s is a local scoring function that considers only a single dependency arc at a time. (We suppress dependency labels; there are various ways to incorporate them, discussed later.) To define s, McDonald et al. (2005a) used hand-engineered features of the surrounding and in-between context of x_h and x_m; more recently, Kiperwasser and Goldberg (2016) used a bidirectional LSTM followed by a single hidden layer with non-linearity.

The exact solution to Eq. 1 can be found using a minimum (directed) spanning tree algorithm (McDonald et al., 2005b) or, under a projectivity constraint, a dynamic programming algorithm (Eisner, 1996), in $O(n^2)$ or $O(n^3)$ runtime, respectively. We refer to parsing with a minimum spanning tree algorithm as **MST parsing**.

An alternative that runs in linear time is **transition-based** parsing, which recasts parsing as a sequence of actions that manipulate auxiliary data structures to incrementally build a parse tree (Nivre, 2003). Such parsers can return a solution in a faster $O(n)$ asymptotic runtime. Unlike FOG parsers, transition-based parsers allow the use of scoring functions with history-based features, so that attachment decisions can interact more freely; the best performing parser at the time of this writing employ neural networks (Andor et al., 2016).

[1]`https://github.com/adhigunasurya/distillation_parser.git`

Let $h_{\boldsymbol{y}}(m)$ denote the parent of x_m in $\boldsymbol{y}$ (using a special null symbol when m is the root of the tree), and $h_{\boldsymbol{y}'}(m)$ denotes the parent of x_m in the predicted tree $\boldsymbol{y}'$. Given two dependency parses of the same sentence, $\boldsymbol{y}$ and $\boldsymbol{y}'$, the **Hamming cost** is

$$C_H(\boldsymbol{y}, \boldsymbol{y}') = \sum_{m=1}^{n} \begin{cases} 0 & \text{if } h_{\boldsymbol{y}}(m) = h_{\boldsymbol{y}'}(m) \\ 1 & \text{otherwise} \end{cases}$$

This cost underlies the standard dependency parsing evaluation scores (unlabeled and labeled attachment scores, henceforth UAS and LAS). More generally, a **cost function** C maps pairs of parses for the same sentence to non-negative values interpreted as the cost of mistaking one for the other, and a **first-order cost function** (FOC) is one that decomposes by attachments, like the Hamming cost.

Given a cost function C and a probabilistic model that defines $p(\boldsymbol{y} \mid \boldsymbol{x})$, **minimum Bayes risk** (MBR) decoding is defined by

$$\begin{aligned} \hat{\boldsymbol{y}}_{\text{MBR}}(\boldsymbol{x}) &= \arg \min_{\boldsymbol{y} \in \mathcal{T}(\boldsymbol{x})} \sum_{\boldsymbol{y}' \in \mathcal{T}(\boldsymbol{x})} p(\boldsymbol{y}' \mid \boldsymbol{x}) \cdot C(\boldsymbol{y}, \boldsymbol{y}') \\ &= \arg \min_{\boldsymbol{y} \in \mathcal{T}(\boldsymbol{x})} \mathbb{E}_{p(\boldsymbol{Y} \mid \boldsymbol{x})}[C(\boldsymbol{y}, \boldsymbol{Y})]. \end{aligned} \tag{2}$$

Under the Hamming cost, MBR parsing equates algorithmically to FOG parsing with $s(h, m, \boldsymbol{x}) = p((h, m) \in \boldsymbol{Y} \mid \boldsymbol{x})$, the posterior marginal of the attachment under p. This is shown by linearity of expectation; see also Titov and Henderson (2006).

Apart from MBR decoding, cost functions are also used for *discriminative training* of a parser. For example, suppose we seek to estimate the parameters $\boldsymbol{\theta}$ of scoring function $S_{\boldsymbol{\theta}}$. One approach is to minimize the structured hinge loss of a training dataset $\mathcal{D}$ with respect to $\boldsymbol{\theta}$:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}} \Big[&- S_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{x}) \\ &+ \max_{\boldsymbol{y}' \in \mathcal{T}(\boldsymbol{x})} \big(S_{\boldsymbol{\theta}}(\boldsymbol{y}', \boldsymbol{x}) + C(\boldsymbol{y}', \boldsymbol{y}) \big) \Big] \end{aligned} \tag{3}$$

Intuitively, this amounts to finding parameters that separate the model score of the correct parse from any wrong parse by a distance proportional to the cost of the wrong parse. With regularization, this is equivalent to the structured support vector machine

(Taskar et al., 2005; Tsochantaridis et al., 2005), and if $S_{\boldsymbol{\theta}}$ is (sub)differentiable, many algorithms are available. Variants have been used extensively in training graph-based parsers (McDonald et al., 2005b; Martins et al., 2009), which typically make use of Hamming cost, so that the inner $\max$ can be solved efficiently using FOG parsing with a slightly revised local scoring function:

$$s'(h, m, \boldsymbol{x}) = s(h, m, \boldsymbol{x}) + \begin{cases} 0 & \text{if } (h, m) \in \boldsymbol{y} \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

Plugging this into Eq. 1 is known as **cost-augmented** parsing.

3 Consensus and Minimum Bayes Risk

Despite the recent success of neural network dependency parsers, most prior works exclusively report single-model performance. Ensembling neural network models trained from different random starting points is a standard technique in a variety of problems, such as machine translation (Sutskever et al., 2014) and constituency parsing (Vinyals et al., 2015). We aim to investigate the benefit of ensembling independently trained neural network dependency parsers by applying the parser ensembling method of Sagae and Lavie (2006) to a collection of N strong neural network base parsers.

Here, each base parser is an instance of the greedy, transition-based parser of Dyer et al. (2015), known as the stack LSTM parser, trained from a different random initial estimate. Given a sentence $\boldsymbol{x}$, the consensus FOG parser (Eq. 1) defines score $s(h, m, \boldsymbol{x})$ as the number of base parsers that include the attachment (h, m), which we denote $votes(h, m)$.[2] An example of this scoring function with an ensemble of 20 models is shown in Figure 1 We assign to dependency (h, m) the label most frequently selected by the base parsers that attach m to h.

Next, note that if we let $s(h, m, \boldsymbol{x}) = votes(h, m)/N$, this has no effect on the parser (we have only scaled by a constant factor). We can therefore view s as a posterior marginal, and the ensemble parser as an MBR parser (Eq. 2).

[2]An alternative to building an ensemble of stack LSTM parsers in this way would be to average the softmax decisions at each timestep (transition), similar to Vinyals et al. (2015).

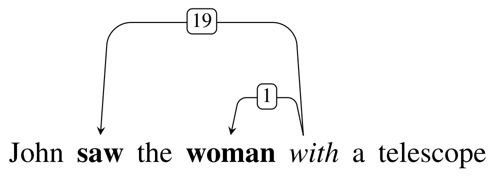

Figure 1: Our ensemble's votes (20 models) on an ambiguous PP attachment of *with*. The ensemble is nearly but not perfectly unanimous in selecting *saw* as the head.

Model	UAS	LAS	UEM
Andor et al. (2016)	**94.61**	**92.79**	-
$N = 1$ (stack LSTM)	93.10	90.90	47.60
ensemble, $N = 5$, MST	93.91	91.94	50.12
ensemble, $N = 10$, MST	94.34	92.47	52.07
ensemble, $N = 15$, MST	94.40	92.57	51.86
ensemble, $N = 20$, MST	94.51	92.70	**52.44**

Table 1: PTB-SD task: ensembles improve over a strong greedy baseline. UEM indicates unlabeled exact match.

Experiment. We consider this approach on the Stanford dependencies version 3.3.0 (De Marneffe and Manning, 2008) Penn Treebank task. As noted, the base parsers instantiate the greedy stack LSTM parser (Dyer et al., 2015).[3]

Table 1 shows that ensembles, even with small N, strongly outperform a single stack LSTM parser. Our ensembles of greedy, locally normalized parsers perform comparably to the best previously reported, due to Andor et al. (2016), which uses a beam (width 32) for training and decoding.

4 What is Ensemble Uncertainty?

While previous works have already demonstrated the merit of ensembling in dependency parsing (Sagae and Lavie, 2006; Surdeanu and Manning, 2010), usually with diverse base parsers, we consider whether the posterior marginals estimated by $\hat{p}((h, m) \in \boldsymbol{Y} \mid \boldsymbol{x}) = votes(h, m)/N$ can be interpreted. We conjecture that disagreement among base parsers about where to attach x_m (i.e., uncertainty in the posterior) is a sign of difficulty or am-

[3]We use the standard data split (02–21 for training, 22 for development, 23 for test), automatically predicted part-of-speech tags, same pretrained word embedding as Dyer et al. (2015), and recommended hyperparameters; `https://github.com/clab/lstm-parser`, each with a different random initialization; this differs from past work on ensembles, which often uses different base model architectures.

Sentence: It will **go** for **work** ranging from refinery **modification** to **changes** in the distribution **system**, *including* the way service stations **pump** fuel into cars.

x_h	posterior	new cost	Hamming
go	0.143	0.143	1
work	0.095	0.191	1
modification	0.190	0.096	1
changes	**0.286**	**0.000**	**0**
system	0.095	0.191	1
pump	0.190	0.096	1
stations	0.000	0.286	1

Table 2: An ambiguous sentence from the training set and the posteriors[4] of various possible parents for *including*. The last two columns are, respectively, the contributions to the distillation cost C_D (explained in §5.1, Eq. 5) and the standard Hamming cost C_H. The most probable head under the ensemble is *changes*, which is also the correct answer.

biguity. If this is true, then the ensemble provides information about which confusions are more or less reasonable—information we will exploit in our distilled parser (§5).

A complete linguistic study is out of scope here; instead, we provide a motivating example before empirically validating our conjecture. Table 2 shows an example where there is considerable disagreement among base parsers over the attachment of a word (*including*). We invite the reader to attempt to select the correct attachment and gauge the difficulty of doing so, before reading on.

Regardless of whether our intuition that this is an inherently difficult and perhaps ambiguous case is correct, it is uncontroversial to say that the words in the sentence not listed, which received zero votes (e.g., both instances of *the*), are obviously implausible attachments.

Our next idea is to transform ensemble uncertainty into a new estimate of cost—a replacement for the Hamming cost—and use it in discriminative training of a single FOG parser. This allows us to distill what has been learned by the ensemble into a single model.

5 Distilling the Ensemble

Despite its state of the art performance, our ensemble requires N parsing calls to decode each sentence. To reduce the computational cost, we introduce a method for "distilling" the ensemble's knowledge into a single parser, making use of a novel **cost function** to communicate this knowledge from the ensemble to the distilled model. While models that combine the outputs of other parsing models have been proposed before (Martins et al., 2008; Nivre and McDonald, 2008; Zhang and Clark, 2008, *inter alia*), these works incorporated the scores or outputs of the baseline parsers as **features** and as such require running the first-stage models at test-time. Creating a cost function from a data analysis procedure is, to our knowledge, a new idea.

The idea is attractive because cost functions are model-agnostic; they can be used with any parser amenable to discriminative training. Further, only the training procedure changes; parsing at test time does not require consulting the ensemble at all, avoiding the costly application of the N parsers to new data, unlike model combination techniques like stacking and beam search.

Distilling an ensemble of classifiers into one simpler classifer that behaves similarly is due to Bucilă et al. (2006) and Hinton et al. (2015); they were likewise motivated by a desire to create a simpler model that was cheaper to run at test time. In their work, the ensemble provides a probability distribution over labels for each input, and this predicted distribution serves as the training target for the distilled model (a sum of two cross entropies objective is used, one targeting the empirical training distribution and the other targeting the ensemble's posterior distribution). This can be contrasted with the supervision provided by the training data alone, which conventionally provides a single correct label for each instance. These are respectively called "soft" and "hard" targets.

We propose a novel adaptation of the soft target idea to the structured output case. Since a sentence

[4]In §3, we used 20 models. Since those 20 models were trained on the whole training set, they cannot be used to obtain the uncertainty estimates on the training set, where the example sentence in Table 2 comes from. Therefore we trained a new ensemble of 21 models from scratch with five-way jackknifing. The same jackknifing setting is used in the distillation parser (§6).

Sentence: John **saw** the **woman** *with* a telescope

x_h	soft	hard
John	0.0	0
saw	**0.95**	**1**
the	0.0	0
woman	0.05	0
a	0.0	0
telescope	0.0	0

Table 3: Example of soft targets (taken from our 20-model ensemble's uncertainty on the sentence) and hard targets (taken from the gold standard) for possible parents of *with*. The soft target corresponds with the posterior (second column) in Table 2, but the hard target differs from the Hamming cost (last column of Table 2) since the hard target assigns a value of 1 to the correct answer and 0 to all others (the reverse is true for Hamming cost).

has an exponential (in its length) number of parses, representing the posterior distribution over parses predicted by the ensemble is nontrivial. We solve this problem by taking a single parse from each model, representing the N-sized ensemble's parse distribution using N samples.

Second, rather than considering uncertainty at the level of complete parse trees (which would be analogous to the classification case) or larger structures, we instead consider uncertainty about *individual* attachments, and seek to "soften" the attachment targets used in training the parser. An illustration for the prepositional phrase attachment ambiguity in Fig. 1, taken from the ensemble output for the sentence, is shown in Table 3. Soft targets allow us to encode the notion that mistaking *woman* as the parent of *with* is less bad than attaching *with* to *John* or *telescope*. Hard targets alone do not capture this information.

5.1 Distillation Cost Function

The natural place to exploit this additional information when training a parser is in the cost function. When incorporated into discriminative training, the Hamming cost encodes hard targets: the correct attachment should receive a higher score than all incorrect ones, with the same margin. Our distillation cost function aims to reduce the cost of decisions that—based on the ensemble uncertainty—appear to be more difficult, or where there may be multiple plausible attachments.

Let $\pi(h, m) =$

$$1 - \hat{p}((h, m) \in \boldsymbol{Y} \mid \boldsymbol{x}) = \frac{N - votes(h, m)}{N}.$$

Our new cost function is defined by $C_D(\boldsymbol{y}, \boldsymbol{y'}) =$

$$\begin{aligned} &\textstyle\sum_{m=1}^{n} \max\left\{0, \pi(h_{\boldsymbol{y'}}(m), m) - \pi(h_{\boldsymbol{y}}(m), m)\right\} \\ &= \textstyle\sum_{m=1}^{n} \max\left\{0, \hat{p}(h_{\boldsymbol{y}}(m), m) - \hat{p}(h_{\boldsymbol{y'}}(m), m)\right\}. \end{aligned} \quad (5)$$

Recall that $\boldsymbol{y}$ denotes the correct parse, according to the training data, and $\boldsymbol{y'}$ is a candidate parse.

This function has several attractive properties:

1. When a word x_m has more than one plausible (according to the ensemble) but incorrect (according to the annotations) attachment, each one has a diminished cost (relative to Hamming cost and all implausible attachments).
2. The correct attachment (according to the gold-standard training data) always has zero cost since $h_{\boldsymbol{y}}(m) = h_{\boldsymbol{y'}}(m)$ and Eq. 5 cancels out.
3. When the ensemble is confident, cost for its choice(s) is lower than it would be under Hamming cost—even when the ensemble is wrong. This means that we are largely training the distilled parser to simulate the ensemble, including mistakes and correct predictions. This encourages the model to replicate the state of the art ensemble performance.
4. Further, when the ensemble is perfectly confident and correct, every incorrect attachment has a cost of 1, just as in Hamming cost.
5. The cost of any attachment is bounded above by the proportion of votes assigned to the correct attachment.

One way to understand this cost function is to imagine that it gives the parser more ways to achieve a zero-cost[5] attachment. The first is to correctly attach a word to its correct parent. The second is to predict a parent that the ensemble prefers to the correct parent, i.e., $\pi(h_{\boldsymbol{y'}}(m), m) < \pi(h_{\boldsymbol{y}}(m), m)$. Any other decision will incur a non-zero cost that is

[5]It is important to note the difference between cost (Eq. 5) and loss (Eq. 3).

proportional to the implausibility of the attachment, according to the ensemble. Hence the model is supervised both by the hard targets in the training data annotations and the soft targets from the ensemble.

While it may seem counter-intuitive to place zero cost on an incorrect attachment, recall that the *cost* is merely a margin that must separate the scores of parses containing correct and incorrect arcs. In contrast, the *loss* (in our case, the structured hinge loss) is the "penalty" the learner tries to minimize while training the graph-based parser, which depends on both the *cost* and *model score* as defined in Equation 3. When an incorrect arc is preferred by the ensemble over the gold arc (hence assigned a cost/margin of 0), the model will still incur a loss if $s(h_{\boldsymbol{y}}(m), m, \boldsymbol{x}) < s(h_{\boldsymbol{y}'}(m), m, \boldsymbol{x})$. In other words, the score of *any* incorrect arc (including those strongly preferred by the ensemble) cannot be higher than the score of the gold arc.

The learner only incurs 0 loss if $s(h_{\boldsymbol{y}}(m), m, \boldsymbol{x}) \geq s(h_{\boldsymbol{y}'}(m), m, \boldsymbol{x})$. This means that the gold score and the predicted score can have a margin of 0 (i.e., have the same score and incur no loss) when the ensemble is highly confident of that prediction, but the score of the correct parse cannot be lower regardless of how confident the ensemble is (hence the objective does not encourage incorrect trees at the expense of gold ones).

In the example in Table 2, we show the (additive) contribution to the distillation cost by each attachment decision (column labeled "new cost"). Note that more plausible attachments according to the ensemble have a lower cost than less plausible ones (e.g., the cost for *modification* is less than *system*, though both are incorrect). While in the last line *stations* received no votes in the ensemble (implausible attachment), its contribution to the cost is bounded by the proportion of votes for correct attachment. The intuition is that, when the ensemble is not certain of the correct answer, it should not assign a large cost to implausible attachments. In contrast, Hamming cost would assign a cost of 1 (column labeled "Hamming") in all incorrect cases.

5.2 Distilled Parser

Our distilled parser is trained discriminatively with the structured hinge loss (Eq. 3). This is a natural choice because it makes the cost function explicit and central to learning.[6] Further, because our ensemble's posterior gives us information about each attachment individually, the cost function we construct can be first-order, which simplifies training with exact inference.

This approach to training a model is well-studied for a FOG parser, but not for a transition-based parser, which is comprised of a collection of classifiers trained to choose good sequences of transitions—not to score whole trees for good attachment accuracy. Transition-based approaches are therefore unsuitable for our proposed distillation cost function, even though they are asymptotically faster. We proceed with a FOG parser (with Eisner's algorithm for English and Chinese, and MST for German since it contains a considerable number of non-projective trees) as the distilled model.

Concretely, we use a bidirectional LSTM followed by a hidden layer of non-linearity to calculate the scoring function $s(h, m, \boldsymbol{x})$, following Kiperwasser and Goldberg (2016) with minor modifications. The bidirectional LSTM maps each word x_i to a vector $\bar{\mathbf{x}}_i$ that embeds the word in context (i.e., $\boldsymbol{x}_{1:i-1}$ and $\boldsymbol{x}_{i+1:n}$). Local attachment scores are given by:

$$s(h, m, \boldsymbol{x}) = \mathbf{v}^\top \tanh\left(\mathbf{W}[\bar{\mathbf{x}}_h; \bar{\mathbf{x}}_m] + \mathbf{b}\right) \quad (6)$$

where the model parameters are $\mathbf{v}$, $\mathbf{W}$, and $\mathbf{b}$, plus the bidirectional LSTM parameters. We will refer to this parsing model as **neural FOG**.

Our model architecture is nearly identical to that of Kiperwasser and Goldberg (2016), with two primary differences. The first difference is that we fix the pretrained word embeddings and compose them with learned embeddings and POS tag embeddings (Dyer et al., 2015), allowing the model to simultaneously leverage pretrained vectors and learn a task-specific representation.[7] Unlike Kiperwasser and Goldberg (2016), we did not observe any degradation by incorporating the pretrained vectors. Second,

[6]Alternatives that do not use cost functions include probabilistic parsers, whether locally normalized like the stack LSTM parser used within our ensemble, or globally normalized, as in Andor et al. (2016); cost functions can be incorporated in such cases with minimum risk training (Smith and Eisner, 2006) or softmax margin (Gimpel and Smith, 2010).

[7]To our understanding, Kiperwasser and Goldberg (2016) initialized with pretrained vectors and backpropagated during training.

we apply a per-epoch learning rate decay of 0.05 to the Adam optimizer. While the Adam optimizer automatically adjusts the global learning rate according to past gradient magnitudes, we find that this additional per-epoch decay consistently improves performance across all settings and languages.

6 Experiments

We ran experiments on the English PTB-SD version 3.3.0, Penn Chinese Treebank (Xue et al., 2002), and German CoNLL 2009 (Hajič et al., 2009) tasks.

Experimental settings. We used the standard splits for all languages. Like Chen and Manning (2014) and Dyer et al. (2015), we use predicted tags with the Stanford tagger (Toutanova et al., 2003) for English and gold tags for Chinese. For German we use the predicted tags provided by the CoNLL 2009 shared task organizers. All models were augmented with pretrained structured-skipgram (Ling et al., 2015) embeddings; for English we used the Gigaword corpus and 100 dimensions, for Chinese Gigaword and 80, and for German WMT 2010 monolingual data and 64.

Hyperparameters. The hyperparameters for neural FOG are summarized in Table 4. For the Adam optimizer we use the default settings in the CNN neural network library.[8] Since the ensemble is used to obtain the uncertainty on the training set, it is imperative that the stack LSTMs do not overfit the training set. To address this issue, we performed five-way jackknifing of the training data for each stack LSTM model to obtain the training data uncertainty under the ensemble. To obtain the ensemble uncertainty on each language, we use 21 base models for English (see footnote 4), 17 for Chinese, and 11 for German.

Speed. One potential drawback of using a quadratic or cubic time parser to distill an ensemble of linear-time transition-based models is speed. Our FOG model is implemented using the same CNN library as the stack LSTM transition-based parser. On the same single-thread CPU hardware, the distilled MST parser[9] parses 20 sentences per second without any pruning, while a single stack LSTM model is only three times faster at 60 sentences per second. Running an ensemble of 20 stack LSTMs is at least 20 times slower (without multi-threading), not including consensus parsing. In the end, the distilled parser is more than ten times faster than the ensemble pipeline.

[8]https://github.com/clab/cnn.git

[9]The runtime of the Hamming-cost bidirectional LSTM FOG parser is the same as the distilled parser.

Bi-LSTM dimension	100
Bi-LSTM layers	2
POS tag embedding	12
Learned word embedding	32
Hidden Layer Units	100
Labeler Hiden Layer Units	100
Optimizer	Adam
Learning rate decay	0.05

Table 4: Hyperparameters for the distilled FOG parser. Both the model architecture and the hyperparameters are nearly identical with Kiperwasser and Goldberg (2016). We apply a per-epoch learning rate decay to the Adam optimizer, which consistently improves performance across all datasets.

Accuracy. All scores are shown in Table 5. First, consider the neural FOG parser trained with Hamming cost (C_H in the second-to-last row). This is a very strong benchmark, outperforming many higher-order graph-based and neural network models on all three datasets. Nonetheless, training the same model with distillation cost gives consistent improvements for all languages. For English, we see that this model comes close to the slower ensemble it was trained to simulate. For Chinese, it achieves the best published scores, for German the best published UAS scores, and just after Bohnet and Nivre (2012) for LAS.

Effects of Pre-trained Word Embedding. As an ablation study, we ran experiments on English without pre-trained word embedding, both with the Hamming and distillation costs. The model trained with Hamming cost achieved 93.1 UAS and 90.9 LAS, compared to 93.6 UAS and 91.1 LAS for the model with distillation cost. This result further showcases the consistent improvements from using the distillation cost across different settings and languages.

We conclude that "soft targets" derived from ensemble uncertainty offer useful guidance, through the distillation cost function and discriminative training of a graph-based parser. Here we consid-

System	Method	P?	PTB-SD		CTB		German CoNLL'09	
			UAS	LAS	UAS	LAS	UAS	LAS
Zhang and Nivre (2011)	Transition (beam)		-	-	86.0	84.4	-	-
Bohnet and Nivre (2012)[†]	Transition (beam)		-	-	87.3	85.9	91.37	89.38
Chen and Manning (2014)	Transition (greedy)	✓	91.8	89.6	83.9	82.4	-	-
Dyer et al. (2015)	Transition (greedy)	✓	93.1	90.9	87.2	85.7	-	-
Weiss et al. (2015)	Transition (beam)	✓	94.0	92.0	-	-	-	-
Yazdani and Henderson (2015)	Transition (beam)		-	-	-	-	89.6	86.0
Ballesteros et al. (2015)	Transition (greedy)		91.63	89.44	85.30	83.72	88.83	86.10
Ballesteros et al. (2016)	Transition (greedy)	✓	93.56	91.42	87.65	86.21	-	-
Kiperwasser and Goldberg (2016)	Transition (greedy)	✓	93.9	91.9	87.6	86.1	-	-
Andor et al. (2016)	Transition (beam)	✓	**94.61**	**92.79**	-	-	90.91	89.15
Ma and Zhao (2012)	Graph (4th order)		-	-	87.74	-	-	-
Martins et al. (2013)	Graph (3rd order)		93.1	-	-	-	-	-
Le and Zuidema (2014)	Reranking/blend	✓	93.8	91.5	-	-	-	-
Zhu et al. (2015)	Reranking/blend	✓	-	-	85.7	-	-	-
Kiperwasser and Goldberg (2016)	Graph (1st order)		93.1	91.0	86.6	85.1	-	-
Wang and Chang (2016)	Graph (1st order)	✓	94.08	91.82	87.55	86.23	-	-
This work: ensemble, $N = 20$, MST	Transition (greedy)	✓	94.51	92.70	**89.80**	**88.56**	**91.86**	**89.98**
This work: neural FOG, C_H	Graph (1st order)	✓	93.76	91.60	87.32	85.82	91.22	88.82
This work: neural FOG, C_D (distilled)	Graph (1st order)	✓	94.26	92.06	88.87	87.30	91.60	89.24

Table 5: Dependency parsing performance on English, Chinese, and German tasks. The "P?" column indicates the use of pretrained word embeddings. Reranking/blend indicates that the reranker score is interpolated with the base model's score. Note that previous works might use different predicted tags for English. We report accuracy without punctuation for English and Chinese, and with punctuation for German, using the standard evaluation script in each case. We only consider systems that do not use additional training data. The best overall results are indicated with bold (this was achieved by the ensemble of greedy stack LSTMs in Chinese and German), while the best non-ensemble model is denoted with an underline. The † sign indicates the use of predicted tags for Chinese in the original publication, although we report accuracy using gold Chinese tags based on private correspondence with the authors.

ered a FOG parser, though future work might investigate any parser amenable to training to minimize a cost-aware loss like the structured hinge.

7 Related Work

Our work on ensembling dependency parsers is based on Sagae and Lavie (2006) and Surdeanu and Manning (2010); an additional contribution of this work is to show that the normalized ensemble votes correspond to MBR parsing. Petrov (2010) proposed a similar model combination with random initializations for phrase-structure parsing, using products of constituent marginals. The local optima in his base model's training objective arise from latent variables instead of neural networks (in our case).

Model distillation was proposed by Bucilă et al. (2006), who used a single neural network to simulate a large ensemble of classifiers. More recently, Ba and Caruana (2014) showed that a single shallow neural network can closely replicate the performance of an ensemble of deep neural networks in phoneme recognition and object detection. Our work is closer to Hinton et al. (2015), in the sense that we do not simply *compress* the ensemble and hit the "soft target," but also the "hard target" at the same time[10]. These previous works only used model compression and distillation for classification; we extend the work to a structured prediction problem (dependency parsing).

Täckström et al. (2013) similarly used an ensemble of other parsers to guide the prediction of a seed model, though in a different context of "ambiguity-aware" ensemble training to re-lexicalize a transfer model for a target language. We similarly use an ensemble of models as a supervision for a sin-

[10]Our cost is zero when the correct arc is predicted, regardless of what the soft target thinks, something a compression model without gold supervision cannot do.

gle model. By incorporating the ensemble uncertainty estimates in the cost function, our approach is cheaper, not requiring any marginalization during training. An additional difference is that we learn from the gold labels ("hard targets") rather than only ensemble estimates on unlabeled data.

Kim and Rush (2016) proposed a distillation model at the sequence level, with application in sequence-to-sequence neural machine translation. There are two primary differences with this work. First, we use a global model to distill the ensemble, instead of a sequential one. Second, Kim and Rush (2016) aim to distill a larger model into a smaller one, while we propose to distill an ensemble instead of a single model.

8 Conclusions

We demonstrate that an ensemble of 20 greedy stack LSTMs (Dyer et al., 2015) can achieve state of the art accuracy on English dependency parsing. This approach corresponds to minimum Bayes risk decoding, and we conjecture that the arc attachment posterior marginals quantify a notion of uncertainty that may indicate difficulty or ambiguity. Since running an ensemble is computationally expensive, we proposed discriminative training of a graph-based model with a novel cost function that *distills* the ensemble uncertainty. Deriving a cost function from a statistical model and extending distillation to structured prediction are new contributions. This distilled model, trained to simulate the slower ensemble parser, improves over the state of the art on Chinese and German.

Acknowledgments

We thank Swabha Swayamdipta, Sam Thomson, Jesse Dodge, Dallas Card, Yuichiro Sawai, Graham Neubig, and the anonymous reviewers for useful feedback. We also thank Juntao Yu and Bernd Bohnet for re-running the parser of Bohnet and Nivre (2012) on Chinese with gold tags. This work was sponsored in part by the Defense Advanced Research Projects Agency (DARPA) Information Innovation Office (I2O) under the Low Resource Languages for Emergent Incidents (LORELEI) program issued by DARPA/I2O under Contract No. HR0011-15-C-0114; it was also supported in part by Contract No. W911NF-15-1-0543 with the DARPA and the Army Research Office (ARO). Approved for public release, distribution unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Miguel Ballesteros was supported by the European Commission under the contract numbers FP7-ICT-610411 (project MULTISENSOR) and H2020-RIA-645012 (project KRISTINA).

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proc. of ACL*.

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *Proc. of NIPS*.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proc. of EMNLP*.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack-LSTM parser. In *Proc. of EMNLP*.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proc. of EMNLP-CoNLL*.

Cristian Bucilă, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proc. of KDD*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of EMNLP*.

Marie-Catherine De Marneffe and Christopher D. Manning. 2008. Stanford typed dependencies manual.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING*.

Kevin Gimpel and Noah A Smith. 2010. Softmax-margin CRFs: Training log-linear models with cost functions. In *Proc. of NAACL*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian

Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 Shared Task: Syntactic and semantic dependencies in multiple languages. In *Proc. of CONLL 2009 Shared Task*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proc. of EMNLP*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proc. of ACL*.

Phong Le and Willem Zuidema. 2014. The inside-outside recursive neural network model for dependency parsing. In *Proc. of EMNLP*.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proc. of NAACL*.

Xuezhe Ma and Hai Zhao. 2012. Fourth-order dependency parsing. In *Proc. of COLING*.

André F. T. Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proc. of EMNLP*.

André F. T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Polyhedral outer approximations with application to natural language parsing. In *Proc. of ICML*.

André F. T. Martins, Mariana S.C. Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proc. of ACL*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proc. of ACL*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of EMNLP*.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proc. of ACL*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. of IWPT*.

Slav Petrov. 2010. Products of random latent variable grammars. In *Proc.of NAACL*.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proc. of NAACL*.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proc. of ACL*.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Proc. of NAACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of NIPS*.

Oscar Täckström, Ryan T. McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proc. of NAACL*.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proc. of ICML*.

Ivan Titov and James Henderson. 2006. Bayes risk minimization in natural language parsing. Technical report, University of Geneva.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of NAACL*.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *JMLR*.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proc. of NIPS*.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional LSTM. In *Proc. of ACL*.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proc. of ACL*.

Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. 2002. Building a large-scale annotated chinese corpus. In *Proc. of COLING*.

Majid Yazdani and James Henderson. 2015. Incremental recurrent neural network dependency parser with search-based discriminative training. In *Proc. of CoNLL*.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proc. of EMNLP*.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proc. of ACL*.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. 2015. A re-ranking model for dependency parser with recursive convolutional neural network. In *Proc. of ACL-IJCNLP*.