Distilling Knowledge for Search-based Structured Prediction

Yijia Liu, Wanxiang Che*, Huaipeng Zhao, Bing Qin, Ting Liu
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{yjliu,car,hpzhao,qinb,tliu}@ir.hit.edu.cn

Abstract

Many natural language processing tasks can be modeled into structured prediction and solved as a search problem. In this paper, we distill an ensemble of multiple models trained with different initialization into a single model. In addition to learning to match the ensemble's probability output on the reference states, we also use the ensemble to explore the search space and learn from the encountered states in the exploration. Experimental results on two typical search-based structured prediction tasks – transition-based dependency parsing and neural machine translation show that distillation can effectively improve the single model's performance and the final model achieves improvements of 1.32 in LAS and 2.65 in BLEU score on these two tasks respectively over strong baselines and it outperforms the greedy structured prediction models in previous literatures.

1 Introduction

Search-based structured prediction models the generation of natural language structure (part-of-speech tags, syntax tree, translations, semantic graphs, etc.) as a search problem (Collins and Roark, 2004; Liang et al., 2006; Zhang and Clark, 2008; Huang et al., 2012; Sutskever et al., 2014; Goodman et al., 2016). It has drawn a lot of research attention in recent years thanks to its competitive performance on both accuracy and running time. A stochastic policy that controls the whole search process is usually learned by imitating a *reference policy*. The imitation is usually addressed as training a classifier to predict the reference policy's search action on the encountered states when performing the reference policy. Such imitation process can sometimes be problematic. One problem is the ambiguities of the reference policy, in which multiple actions lead to the optimal structure but usually, only one is chosen as training instance (Goldberg and Nivre, 2012). Another problem is the discrepancy between training and testing, in which during the test phase, the learned policy enters non-optimal states whose search action is never learned (Ross and Bagnell, 2010; Ross et al., 2011). All these problems harm the generalization ability of search-based structured prediction and lead to poor performance.

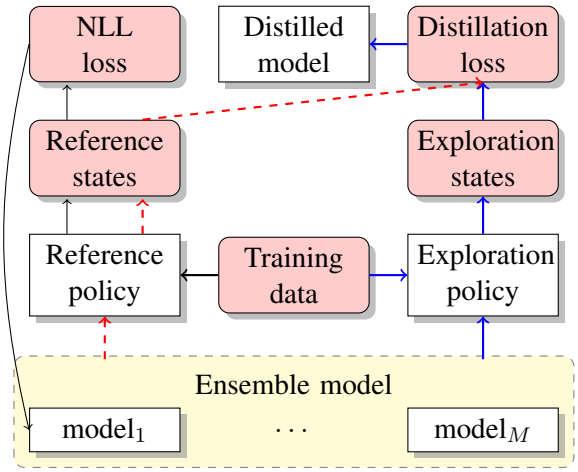

Figure 1: Workflow of our knowledge distillation for search-based structured prediction. The yellow bracket represents the ensemble of multiple models trained with different initialization. The dashed red line shows our *distillation from reference* (§3.2). The solid blue line shows our *distillation from exploration* (§3.3).

Previous works tackle these problems from two directions. To overcome the ambiguities in data, techniques like *ensemble* are often adopted (Di-

* Email corresponding.

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 1393–1402
Melbourne, Australia, July 15 - 20, 2018.

	Dependency parsing	Neural machine translation
s_t	(σ, β, A), where σ is a stack, β is a buffer, and A is the partially generated tree	$(\$, y_1, y_2, ..., y_t)$, where \$ is the start symbol.
$\mathcal{A}$	{SHIFT, LEFT, RIGHT}	pick one word w from the target side vocabulary $\mathcal{W}$.
$\mathcal{S}_0$	$\{([\,], [1, .., n], \emptyset)\}$	$\{(\$)\}$
$\mathcal{S}_T$	$\{([\text{ROOT}], [\,], A)\}$	$\{(\$, y_1, y_2, ..., y_m)\}$
$\mathcal{T}(s, a)$	• SHIFT: $(\sigma, j\|\beta) \rightarrow (\sigma\|j, \beta)$ • LEFT: $(\sigma\|i\ j, \beta) \rightarrow (\sigma\|j, \beta) \quad A \leftarrow A \cup \{i \leftarrow j\}$ • RIGHT: $(\sigma\|i\ j, \beta) \rightarrow (\sigma\|i, \beta) \quad A \leftarrow A \cup \{i \rightarrow j\}$	$(\$, y_1, y_2, ..., y_t) \rightarrow (\$, y_1, y_2, ..., y_t, y_{t+1} = w)$

Table 1: The search-based structured prediction view of transition-based dependency parsing (Nivre, 2008) and neural machine translation (Sutskever et al., 2014).

etterich, 2000). To mitigate the discrepancy, exploration is encouraged during the training process (Ross and Bagnell, 2010; Ross et al., 2011; Goldberg and Nivre, 2012; Bengio et al., 2015; Goodman et al., 2016). In this paper, we propose to consider these two problems in an integrated *knowledge distillation* manner (Hinton et al., 2015). We distill a single model from the ensemble of several baselines trained with different initialization by matching the ensemble's output distribution on the reference states. We also let the ensemble randomly explore the search space and learn the single model to mimic ensemble's distribution on the encountered exploration states. Combing the distillation from reference and exploration further improves our single model's performance. The workflow of our method is shown in Figure 1.

We conduct experiments on two typical search-based structured prediction tasks: transition-based dependency parsing and neural machine translation. The results of both these two experiments show the effectiveness of our knowledge distillation method by outperforming strong baselines. In the parsing experiments, an improvement of 1.32 in LAS is achieved and in the machine translation experiments, such improvement is 2.65 in BLEU. Our model also outperforms the greedy models in previous works.

Major contributions of this paper include:

- We study the knowledge distillation in search-based structured prediction and propose to distill the knowledge of an ensemble into a single model by learning to match its distribution on both the reference states (§3.2) and exploration states encountered when using the ensemble to explore the search space (§3.3). A further combination of these two methods is also proposed to improve the performance (§3.4).
- We conduct experiments on two search-based structured prediction problems: transition-based dependency parsing and neural machine translation. In both these two problems, the distilled model significantly improves over strong baselines and outperforms other greedy structured prediction (§4.2). Comprehensive analysis empirically shows the feasibility of our distillation method (§4.3).

2 Background

2.1 Search-based Structured Prediction

Structured prediction maps an input $\mathbf{x} = (x_1, x_2, ..., x_n)$ to its structural output $\mathbf{y} = (y_1, y_2, ..., y_m)$, where each component of $\mathbf{y}$ has some internal dependencies. Search-based structured prediction (Collins and Roark, 2004; Daumé III et al., 2005; Daumé III et al., 2009; Ross and Bagnell, 2010; Ross et al., 2011; Doppa et al., 2014; Vlachos and Clark, 2014; Chang et al., 2015) models the generation of the structure as a search problem and it can be formalized as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}(s, a), \mathcal{S}_0, \mathcal{S}_T)$, in which $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{T}$ is a function that maps $\mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, $\mathcal{S}_0$ is a set of initial states, and $\mathcal{S}_T$ is a set of terminal states. Starting from an initial state $s_0 \in \mathcal{S}_0$, the structured prediction model repeatably chooses an action $a_t \in \mathcal{A}$ by following a *policy* $\pi(s)$ and applies a_t to s_t and enter a new state s_{t+1} as $s_{t+1} \leftarrow \mathcal{T}(s_t, a_t)$, until a final state $s_T \in \mathcal{S}_T$ is achieved. Several natural language structured prediction problems can be modeled under the search-based framework including dependency parsing (Nivre, 2008) and neural machine translation (Liang et al., 2006; Sutskever et al., 2014). Table 1 shows the search-based structured prediction view of these two problems.

In the data-driven settings, $\pi(s)$ controls the whole search process and is usually parameterized by a classifier $p(a \mid s)$ which outputs the proba-

Algorithm 1: Generic learning algorithm for search-based structured prediction.

Input: training data: $\{\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^{N}$; the reference policy: $\pi_{\mathcal{R}}(s, \mathbf{y})$.
Output: classifier $p(a|s)$.

1 $D \leftarrow \emptyset$;
2 **for** $n \leftarrow 1...N$ **do**
3 $t \leftarrow 0$;
4 $s_t \leftarrow s_0(\mathbf{x}^{(n)})$;
5 **while** $s_t \notin \mathcal{S}_T$ **do**
6 $a_t \leftarrow \pi_{\mathcal{R}}(s_t, \mathbf{y}^{(n)})$;
7 $D \leftarrow D \cup \{s_t\}$;
8 $s_{t+1} \leftarrow \mathcal{T}(s_t, a_t)$;
9 $t \leftarrow t + 1$;
10 **end**
11 **end**
12 optimize $\mathcal{L}_{NLL}$;

bility of choosing an action a on the given state s. The commonly adopted greedy policy can be formalized as choosing the most probable action with $\pi(s) = \text{argmax}_a\, p(a \mid s)$ at test stage. To learn an optimal classifier, search-based structured prediction requires constructing a reference policy $\pi_{\mathcal{R}}(s, \mathbf{y})$, which takes an input state s, gold structure $\mathbf{y}$ and outputs its reference action a, and training $p(a \mid s)$ to imitate the reference policy. Algorithm 1 shows the common practices in training $p(a \mid s)$, which involves: first, using $\pi_{\mathcal{R}}(s, \mathbf{y})$ to generate a sequence of reference states and actions on the training data (line 1 to line 11 in Algorithm 1); second, using the states and actions on the reference sequences as examples to train $p(a \mid s)$ with negative log-likelihood (NLL) loss (line 12 in Algorithm 1),

$$\mathcal{L}_{NLL} = \sum_{s \in D} \sum_{a} -\mathbb{1}\{a = \pi_{\mathcal{R}}\} \cdot \log p(a \mid s)$$

where D is a set of training data.

The reference policy is sometimes sub-optimal and ambiguous which means on one state, there can be more than one action that leads to the optimal prediction. In transition-based dependency parsing, Goldberg and Nivre (2012) showed that one dependency tree can be reached by several search sequences using Nivre (2008)'s *arc-standard* algorithm. In machine translation, the ambiguity problem also exists because one source language sentence usually has multiple semantically correct translations but only one reference translation is presented. Similar problems have also been observed in semantic parsing (Goodman et al., 2016). According to Frénay and Verleysen (2014), the widely used NLL loss is vulnerable to ambiguous data which make it worse for search-based structured prediction.

Besides the ambiguity problem, training and testing discrepancy is another problem that lags the search-based structured prediction performance. Since the training process imitates the reference policy, all the states in the training data are optimal which means they are guaranteed to reach the optimal structure. But during the test phase, the model can predict non-optimal states whose search action is never learned. The greedy search which is prone to error propagation also worsens this problem.

2.2 Knowledge Distillation

A cumbersome model, which could be an ensemble of several models or a single model with larger number of parameters, usually provides better generalization ability. *Knowledge distillation* (Buciluǎ et al., 2006; Ba and Caruana, 2014; Hinton et al., 2015) is a class of methods for transferring the generalization ability of the cumbersome *teacher model* into a small *student model*. Instead of optimizing NLL loss, knowledge distillation uses the distribution $q(y \mid x)$ outputted by the teacher model as "soft target" and optimizes the knowledge distillation loss,

$$\mathcal{L}_{KD} = \sum_{x \in D} \sum_{y} -q(y \mid x) \cdot \log p(y \mid x).$$

In search-based structured prediction scenario, x corresponds to the state s and y corresponds to the action a. Through optimizing the distillation loss, knowledge of the teacher model is learned by the student model $p(y \mid x)$. When correct label is presented, NLL loss can be combined with the distillation loss via simple interpolation as

$$\mathcal{L} = \alpha \mathcal{L}_{KD} + (1 - \alpha)\mathcal{L}_{NLL} \quad (1)$$

3 Knowledge Distillation for Search-based Structured Prediction

3.1 Ensemble

As Hinton et al. (2015) pointed out, although the real objective of a machine learning algorithm is to generalize well to new data, models are usually trained to optimize the performance on training data, which bias the model to the training data.

In search-based structured prediction, such biases can result from either the ambiguities in the training data or the discrepancy between training and testing. It would be more problematic to train $p(a \mid s)$ using the loss which is in-robust to ambiguities and only considering the optimal states.

The effect of ensemble on ambiguous data has been studied in Dietterich (2000). They empirically showed that ensemble can overcome the ambiguities in the training data. Daumé III et al. (2005) also use weighted ensemble of parameters from different iterations as their final structure prediction model. In this paper, we consider to use ensemble technique to improve the generalization ability of our search-based structured prediction model following these works. In practice, we train M search-based structured prediction models with different initialized weights and ensemble them by the average of their output distribution as $q(a \mid s) = \frac{1}{M} \sum_m q_m(a \mid s)$. In Section 4.3.1, we empirically show that the ensemble has the ability to choose a good search action in the optimal-yet-ambiguous states and the non-optimal states.

3.2 Distillation from Reference

As we can see in Section 4, ensemble indeed improves the performance of baseline models. However, real world deployment is usually constrained by computation and memory resources. Ensemble requires running the structured prediction models for multiple times, and that makes it less applicable in real-world problem. To take the advantage of the ensemble model while avoid running the models multiple times, we use the knowledge distillation technique to distill a single model from the ensemble. We started from changing the NLL learning objective in Algorithm 1 into the distillation loss (Equation 1) as shown in Algorithm 2. Since such method learns the model on the states produced by the reference policy, we name it as *distillation from reference*. Blocks connected by in dashed red lines in Figure 1 show the workflow of our *distillation from reference*.

3.3 Distillation from Exploration

In the scenario of search-based structured prediction, transferring the teacher model's generalization ability into a student model not only includes matching the teacher model's soft targets on the reference search sequence, but also imitating the search decisions made by the teacher model. One way to accomplish the imitation can be sampling

Algorithm 2: Knowledge distillation for search-based structured prediction.

Input: training data: $\{\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^{N}$; the reference policy: $\pi_{\mathcal{R}}(s, \mathbf{y})$; the exploration policy: $\pi_{\mathcal{E}}(s)$ which samples an action from the annealed ensemble $q(a \mid s)^{\frac{1}{T}}$

Output: classifier $p(a \mid s)$.

```
D ← ∅;
for n ← 1...N do
    t ← 0;
    s_t ← s_0(x^(n));
    while s_t ∉ S_T do
        if distilling from reference then
            a_t ← π_R(s_t, y^(n));
        else
            a_t ← π_E(s_t);
        end
        D ← D ∪ {s_t};
        s_{t+1} ← T(s_t, a_t);
        t ← t + 1;
    end
end
if distilling from reference then
    optimize αL_KD + (1 − α)L_NLL;
else
    optimize L_KD;
end
```

search sequence from the ensemble and learn from the soft target on the sampled states. More concretely, we change $\pi_{\mathcal{R}}(s, \mathbf{y})$ into a policy $\pi_{\mathcal{E}}(s)$ which samples an action a from $q(a \mid s)^{\frac{1}{T}}$, where T is the temperature that controls the sharpness of the distribution (Hinton et al., 2015). The algorithm is shown in Algorithm 2. Since such distillation generate training instances from exploration, we name it as *distillation from exploration*. Blocks connected by in solid blue lines in Figure 1 show the workflow of our *distillation from exploration*.

On the sampled states, reference decision from $\pi_{\mathcal{R}}$ is usually non-trivial to achieve, which makes learning from NLL loss infeasible. In Section 4, we empirically show that fully distilling from the soft target, i.e. setting $\alpha = 1$ in Equation 1, achieves comparable performance with that both from distillation and NLL.

3.4 Distillation from Both

Distillation from reference can encourage the model to predict the action made by the reference policy and distillation from exploration learns the model on arbitrary states. They transfer the generalization ability of the ensemble from different aspects. Hopefully combining them can further improve the performance. In this paper, we combine distillation from reference and exploration with the following manner: we use $\pi_{\mathcal{R}}$ and $\pi_{\mathcal{E}}$ to generate a set of training states. Then, we learn $p(a \mid s)$ on the generated states. If one state was generated by the reference policy, we minimize the interpretation of distillation and NLL loss. Otherwise, we minimize the distillation loss only.

4 Experiments

We perform experiments on two tasks: transition-based dependency parsing and neural machine translation. Both these two tasks are converted to search-based structured prediction as Section 2.1.

For the transition-based parsing, we use the stack-lstm parsing model proposed by Dyer et al. (2015) to parameterize the classifier.[1] For the neural machine translation, we parameterize the classifier as an LSTM encoder-decoder model by following Luong et al. (2015).[2] We encourage the reader of this paper to refer corresponding papers for more details.

4.1 Settings

4.1.1 Transition-based Dependency Parsing

We perform experiments on Penn Treebank (PTB) dataset with standard data split (Section 2-21 for training, Section 22 for development, and Section 23 for testing). Stanford dependencies are converted from the original constituent trees using Stanford CoreNLP 3.3.0[3] by following Dyer et al. (2015). Automatic part-of-speech tags are assigned by 10-way jackknifing whose accuracy is 97.5%. Labeled attachment score (LAS) excluding punctuation are used in evaluation. For the other hyper-parameters, we use the same settings as Dyer et al. (2015). The best iteration and α is determined on the development set.

[1]The code for parsing experiments is available at: https://github.com/Oneplus/twpipe.

[2]We based our NMT experiments on OpenNMT (Klein et al., 2017). The code for NMT experiments is available at: https://github.com/Oneplus/OpenNMT-py.

[3]stanfordnlp.github.io/CoreNLP/history.html

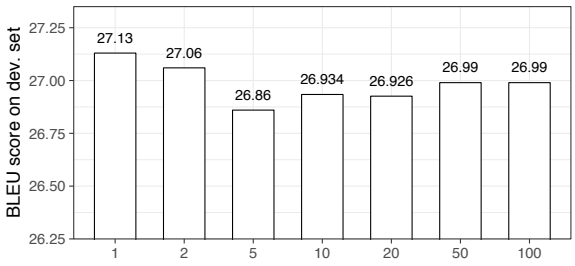

Figure 2: The effect of using different Ks when approximating distillation loss with K-most probable actions in the machine translation experiments.

Reimers and Gurevych (2017) and others have pointed out that neural network training is nondeterministic and depends on the seed for the random number generator. To control for this effect, they suggest to report the average of M differently-seeded runs. In all our dependency parsing, we set $n = 20$.

4.1.2 Neural Machine Translation

We conduct our experiments on a small machine translation dataset, which is the German-to-English portion of the IWSLT 2014 machine translation evaluation campaign. The dataset contains around 153K training sentence pairs, 7K development sentence pairs, and 7K testing sentence pairs. We use the same preprocessing as Ranzato et al. (2015), which leads to a German vocabulary of about 30K entries and an English vocabulary of 25K entries. One-layer LSTM for both encoder and decoder with 256 hidden units are used by following Wiseman and Rush (2016). BLEU (Papineni et al., 2002) was used to evaluate the translator's performance.[4] Like in the dependency parsing experiments, we run $M = 10$ differently-seeded runs and report the averaged score.

Optimizing the distillation loss in Equation 1 requires enumerating over the action space. It is expensive for machine translation since the size of the action space (vocabulary) is considerably large (25K in our experiments). In this paper, we use the K-most probable actions (translations on target side) on one state to approximate the whole probability distribution of $q(a \mid s)$ as $\sum_a q(a \mid s) \cdot \log p(a \mid s) \approx \sum_k^K q(\hat{a}_k \mid s) \cdot \log p(\hat{a}_k \mid s)$, where $\hat{a}_k$ is the k-th probable action. We fix α to

[4]We use multi-bleu.perl to evaluate our model's performance

	LAS
Baseline	90.83
Ensemble (20)	92.73
Distill (reference, α=1.0)	91.99
Distill (exploration, T=1.0)	92.00
Distill (both)	92.14
Ballesteros et al. (2016) (dyn. oracle)	91.42
Andor et al. (2016) (local, B=1)	91.02
Buckman et al. (2016) (local, B=8)	91.19
Andor et al. (2016) (local, B=32)	91.70
Andor et al. (2016) (global, B=32)	92.79
Dozat and Manning (2016)	94.08
Kuncoro et al. (2016)	92.06
Kuncoro et al. (2017)	94.60

Table 2: The dependency parsing results. Significance test (Nilsson and Nivre, 2008) shows the improvement of our *Distill (both)* over *Baseline* is statistically significant with $p < 0.01$.

	BLEU
Baseline	22.79
Ensemble (10)	26.26
Distill (reference, α=0.8)	24.76
Distill (exploration, T=0.1)	24.64
Distill (both)	25.44
MIXER	20.73
BSO (local, B=1)	22.53
BSO (global, B=1)	23.83

Table 3: The machine translation results. MIXER denotes that of Ranzato et al. (2015), BSO denotes that of Wiseman and Rush (2016). Significance test (Koehn, 2004) shows the improvement of our *Distill (both)* over *Baseline* is statistically significant with $p < 0.01$.

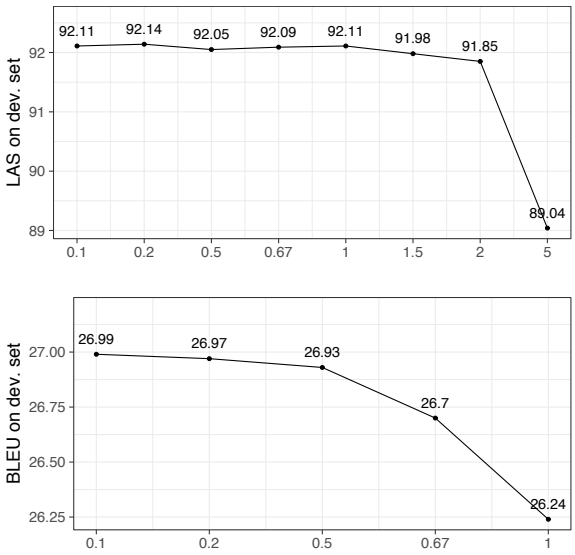

Figure 3: The effect of T on PTB (above) and IWSLT 2014 (below) development set.

1 and vary K and evaluate the distillation model's performance. These results are shown in Figure 2 where there is no significant difference between different Ks and in speed consideration, we set K to 1 in the following experiments.

4.2 Results

4.2.1 Transition-based Dependency Parsing

Table 2 shows our PTB experimental results. From this result, we can see that the ensemble model outperforms the baseline model by 1.90 in LAS. For our distillation from reference, when setting $\alpha = 1.0$, best performance on development set is achieved and the test LAS is 91.99.

We tune the temperature T during exploration and the results are shown in Figure 3. Sharpen the distribution during the sampling process generally performs better on development set. Our distillation from exploration model gets almost the same performance as that from reference, but simply combing these two sets of data outperform both models by achieving an LAS of 92.14.

We also compare our parser with the other parsers in Table 2. The second group shows the greedy transition-based parsers in previous literatures. Andor et al. (2016) presented an alternative state representation and explored both greedy and beam search decoding. (Ballesteros et al., 2016) explores training the greedy parser with dynamic oracle. Our distillation parser outperforms all these greedy counterparts. The third group shows parsers trained on different techniques including decoding with beam search (Buckman et al., 2016; Andor et al., 2016), training transition-based parser with beam search (Andor et al., 2016), graph-based parsing (Dozat and Manning, 2016), distilling a graph-based parser from the output of 20 parsers (Kuncoro et al., 2016), and converting constituent parsing results to dependencies (Kuncoro et al., 2017). Our distillation parser still outperforms its transition-based counterparts but lags the others. We attribute the gap between our parser with the other parsers to the difference in parsing algorithms.

4.2.2 Neural Machine Translation

Table 3 shows the experimental results on IWSLT 2014 dataset. Similar to the PTB parsing results, the ensemble 10 translators outperforms the baseline translator by 3.47 in BLEU score. Distilling from the ensemble by following the reference leads to a single translator of 24.76 BLEU score.

Like in the parsing experiments, sharpen the distribution when exploring the search space is more helpful to the model's performance but the differences when $T \leq 0.2$ is not significant as shown in Figure 3. We set $T = 0.1$ in our distillation from exploration experiments since it achieves the best development score. Table 3 shows the exploration result of a BLEU score of 24.64 and it slightly lags the best reference model. Distilling from both the reference and exploration improves the single model's performance by a large margin and achieves a BLEU score of 25.44.

We also compare our model with other translation models including the one trained with reinforcement learning (Ranzato et al., 2015) and that using beam search in training (Wiseman and Rush, 2016). Our distillation translator outperforms these models.

Both the parsing and machine translation experiments confirm that it's feasible to distill a reasonable search-based structured prediction model by just exploring the search space. Combining the reference and exploration further improves the model's performance and outperforms its greedy structured prediction counterparts.

4.3 Analysis

In Section 4.2, improvements from distilling the ensemble have been witnessed in both the transition-based dependency parsing and neural machine translation experiments. However, questions like "Why the ensemble works better? Is it feasible to fully learn from the distillation loss without NLL? Is learning from distillation loss stable?" are yet to be answered. In this section, we first study the ensemble's behavior on "problematic" states to show its generalization ability. Then, we empirically study the feasibility of fully learning from the distillation loss by studying the effect of α in the distillation from reference setting. Finally, we show that learning from distillation loss is less sensitive to initialization and achieves a more stable model.

	optimal-yet-ambiguous	non-optimal
Baseline	68.59	89.59
Ensemble	74.19	90.90
Distill (both)	81.15	91.38

Table 4: The ranking performance of parsers' output distributions evaluated in MAP on "problematic" states.

4.3.1 Ensemble on "Problematic" States

As mentioned in previous sections, "problematic" states which is either ambiguous or non-optimal harm structured prediciton's performance. Ensemble shows to improve the performance in Section 4.2, which indicates it does better on these states. To empirically testify this, we use dependency parsing as a testbed and study the ensemble's output distribution using the dynamic oracle.

The dynamic oracle (Goldberg and Nivre, 2012; Goldberg et al., 2014) can be used to efficiently determine, given any state s, which transition action leads to the best achievable parse from s; if some errors may have already made, what is the best the parser can do, going forward? This allows us to analyze the accuracy of each parser's individual decisions, in the "problematic" states. In this paper, we evaluate the output distributions of the baseline and ensemble parser against the *reference actions* suggested by the dynamic oracle. Since dynamic oracle yields more than one reference actions due to ambiguities and previous mistakes and the output distribution can be treated as their scoring, we evaluate them as a ranking problem. Intuitively, when multiple reference actions exist, a good parser should push probability mass to these actions. We draw problematic states by sampling from our baseline parser. The comparison in Table 4 shows that the ensemble model significantly outperforms the baseline on ambiguous and non-optimal states. This observation indicates the ensemble's output distribution is more "informative", thus generalizes well on problematic states and achieves better performance. We also observe that the distillation model perform better than both the baseline and ensemble. We attribute this to the fact that the distillation model is learned from exploration.

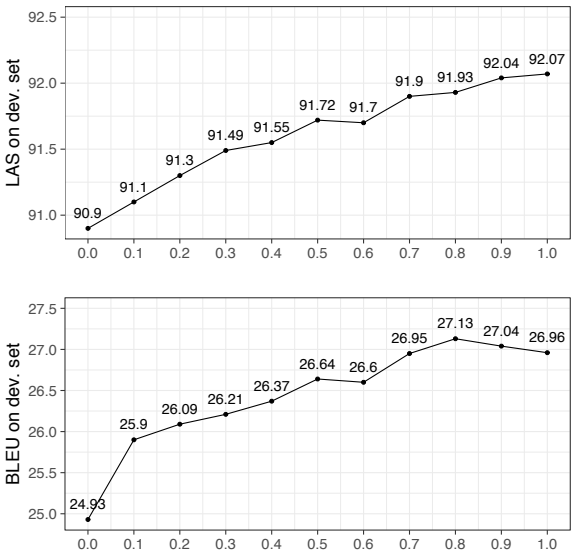

Figure 4: The effect of α on PTB (above) and IWSLT 2014 (below) development set.

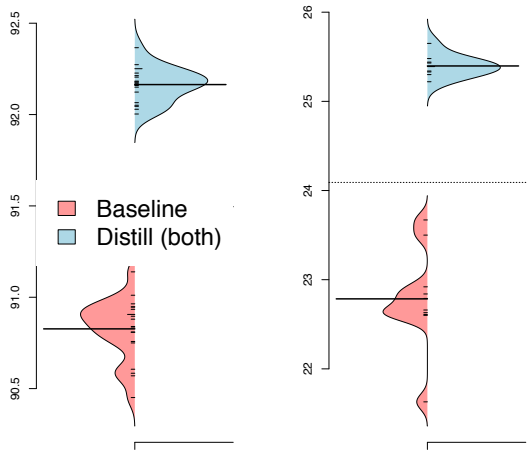

Figure 5: The distributions of scores for the baseline model and our *distillation from both* on PTB test (left) and IWSLT 2014 test (right) on differently-seeded runs.

system	seeds	min	max	σ
PTB test				
Baseline	20	90.45	91.14	0.17
Distill (both)	20	92.00	92.37	0.09
IWSLT 2014 test				
Baseline	10	21.63	23.67	0.55
Distill (both)	10	24.22	25.65	0.12

Table 5: The minimal, maximum, and standard derivation values on differently-seeded runs.

4.3.2 Effect of α

Over our distillation from reference model, we study the effect of α in Equation 1. We vary α from 0 to 1 by a step of 0.1 in both the transition-based dependency parsing and neural machine translation experiments and plot the model's performance on development sets in Figure 4. Similar trends are witnessed in both these two experiments that model that's configured with larger α generally performs better than that with smaller α. For the dependency parsing problem, the best development performance is achieved when we set $\alpha = 1$, and for the machine translation, the best α is 0.8. There is only 0.2 point of difference between the best α model and the one with α equals to 1. Such observation indicates that when distilling from the reference policy paying more attention to the distillation loss rather than the NLL is more beneficial. It also indicates that fully learning from the distillation loss outputted by the ensemble is reasonable because models configured with $\alpha = 1$ generally achieves good performance.

4.3.3 Learning Stability

Besides the improved performance, knowledge distillation also leads to more stable learning. The performance score distributions of differently-seed runs are depicted as violin plot in Figure 5. Table 5 also reveals the smaller standard derivations are achieved by our distillation methods. As Keskar et al. (2016) pointed out that the generalization gap is not due to *overfit*, but due to the network converge to *sharp minimizer* which generalizes worse, we attribute the more stable training from our distillation model as the distillation loss presents less *sharp minimizers*.

5 Related Work

Several works have been proposed to applying knowledge distillation to NLP problems. Kim and Rush (2016) presented a distillation model which focus on distilling the structured loss from a large model into a small one which works on sequence-level. In contrast to their work, we pay more attention to action-level distillation and propose to do better action-level distillation by both from reference and exploration.

Freitag et al. (2017) used an ensemble of 6-translators to generate training reference. Exploration was tried in their work with beam-search. We differ their work by training the single model

to match the distribution of the ensemble.

Using ensemble in exploration was also studied in reinforcement learning community (Osband et al., 2016). In addition to distilling the ensemble on the labeled training data, a line of semi-supervised learning works show that it's effective to transfer knowledge of cumbersome model into a simple one on the unlabeled data (Liang et al., 2008; Li et al., 2014). Their extensions to knowledge distillation call for further study.

Kuncoro et al. (2016) proposed to compile the knowledge from an ensemble of 20 transition-based parsers into a voting and distill the knowledge by introducing the voting results as a regularizer in learning a graph-based parser. Different from their work, we directly do the distillation on the classifier of the transition-based parser.

Besides the attempts for directly using the knowledge distillation technique, Stahlberg and Byrne (2017) propose to first build the ensemble of several machine translators into one network by unfolding and then use SVD to shrink its parameters, which can be treated as another kind of knowledge distillation.

6 Conclusion

In this paper, we study knowledge distillation for search-based structured prediction and propose to distill an ensemble into a single model both from reference and exploration states. Experiments on transition-based dependency parsing and machine translation show that our distillation method significantly improves the single model's performance. Comparison analysis gives empirically guarantee for our distillation method.

Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by the National Key Basic Research Program of China via grant 2014CB340503 and the National Natural Science Foundation of China (NSFC) via grant 61632011 and 61772153.

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 *NIPS 27*, pages 2654–2662.

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

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS 28*, pages 1171–1179.

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

Jacob Buckman, Miguel Ballesteros, and Chris Dyer. 2016. Transition-based dependency parsing with heuristic backtracking. In *Proc. of EMNLP*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015. Learning to search better than your teacher. In *Proc. of ICML*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proc. of ACL*.

Hal Daumé III, John Langford, and Daniel Marcu. 2005. Search-based structured prediction as classification. In *NIPS Workshop on ASLTSP*.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75(3).

Thomas G. Dietterich. 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157.

Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. 2014. Hc-search: A learning framework for search-based structured prediction. *J. Artif. Intell. Res. (JAIR)*, 50.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

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*.

Markus Freitag, Yaser Al-Onaizan, and Baskaran Sankaran. 2017. Ensemble distillation for neural machine translation. *CoRR*, abs/1702.01802.

Benoît Frénay and Michel Verleysen. 2014. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25:845–869.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proc. of COLING*.

Yoav Goldberg, Francesco Sartorio, and Giorgio Satta. 2014. A tabular method for dynamic oracles in transition-based parsing. *TACL*, 2.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing. In *Proc. of ACL*.

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

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proc. of NAACL*.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836.

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

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proc. of ACL 2017, System Demonstrations*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP 2004*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proc. of EACL*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one MST parser. In *Proc. of EMNLP*.

Zhenghua Li, Min Zhang, and Wenliang Chen. 2014. Ambiguity-aware ensemble training for semi-supervised dependency parsing. In *Proc. of ACL*.

P. Liang, H. Daumé, and D. Klein. 2008. Structure compilation: trading structure for features. pages 592–599.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. of ACL*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*.

Jens Nilsson and Joakim Nivre. 2008. Malteval: an evaluation and visualization tool for dependency parsing. In *Proc. of LREC*. Http://www.lrec-conf.org/proceedings/lrec2008/.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4).

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep exploration via bootstrapped dqn. In *NIPS 29*, pages 4026–4034.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proc. of EMNLP*.

Stephane Ross and Drew Bagnell. 2010. Efficient reductions for imitation learning. In *Proc. of AISTATS*, volume 9.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proc. of AISTATS*, volume 15.

Felix Stahlberg and Bill Byrne. 2017. Unfolding and shrinking neural machine translation ensembles. In *Proc. of EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS 27*, pages 3104–3112.

Andreas Vlachos and Stephen Clark. 2014. A new corpus and imitation learning framework for context-dependent semantic parsing. *Transactions of the Association for Computational Linguistics*, 2:547–559.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proc. of EMNLP*.

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