Training a Parser for Machine Translation Reordering

Jason Katz-Brown Slav Petrov Ryan McDonald Franz Och
David Talbot Hiroshi Ichikawa Masakazu Seno Hideto Kazawa
Google
{jasonkb|slav|ryanmcd|och|talbot|ichikawa|seno|kazawa}@google.com

Abstract

We propose a simple training regime that can improve the extrinsic performance of a parser, given only a corpus of sentences and a way to automatically evaluate the extrinsic quality of a candidate parse. We apply our method to train parsers that excel when used as part of a reordering component in a statistical machine translation system. We use a corpus of weakly-labeled reference reorderings to guide parser training. Our best parsers contribute significant improvements in subjective translation quality while their intrinsic attachment scores typically regress.

1 Introduction

The field of syntactic parsing has received a great deal of attention and progress since the creation of the Penn Treebank (Marcus et al., 1993; Collins, 1997; Charniak, 2000; McDonald et al., 2005; Petrov et al., 2006; Nivre, 2008). A common—and valid—criticism, however, is that parsers typically get evaluated only on Section 23 of the Wall Street Journal portion of the Penn Treebank. This is problematic for many reasons. As previously observed, this test set comes from a very narrow domain that does not necessarily reflect parser performance on text coming from more varied domains (Gildea, 2001), especially web text (Foster, 2010). There is also evidence that after so much repeated testing, parsers are indirectly over-fitting to this set (Petrov and Klein, 2007). Furthermore, parsing was never meant as a stand-alone task, but is rather a means to an end, towards the goal of building systems that can process natural language input.

This is not to say that parsers are not used in larger systems. All to the contrary, as parsing technology has become more mature, parsers have become efficient and accurate enough to be useful in many natural language processing systems, most notably in machine translation (Yamada and Knight, 2001; Galley et al., 2004; Xu et al., 2009). While it has been repeatedly shown that using a parser can bring net gains on downstream application quality, it is often unclear how much intrinsic parsing accuracy actually matters.

In this paper we try to shed some light on this issue by comparing different parsers in the context of machine translation (MT). We present experiments on translation from English to three Subject-Object-Verb (SOV) languages,[1] because those require extensive syntactic reordering to produce grammatical translations. We evaluate parse quality on a number of extrinsic metrics, including word reordering accuracy, BLEU score and a human evaluation of final translation quality. We show that while there is a good correlation between those extrinsic metrics, parsing quality as measured on the Penn Treebank is not a good indicator of the final downstream application quality. Since the word reordering metric can be computed efficiently offline (i.e. without the use of the final MT system), we then propose to tune parsers specifically for that metric, with the goal of improving the performance of the overall system.

To this end we propose a simple training regime

[1] We experiment with Japanese, Korean and Turkish, but there is nothing language specific in our approach.

which we refer to as *targeted self-training* (Section 2). Similar to self-training, a baseline model is used to produce predictions on an unlabeled data set. However, rather than directly training on the output of the baseline model, we generate a list of hypotheses and use an external signal to select the best candidate. The selected parse trees are added to the training data and the model is then retrained. The experiments in Section 5 show that this simple procedure noticeably improves our parsers for the task at hand, resulting in significant improvements in downstream translation quality, as measured in a human evaluation on web text.

This idea is similar in vein to McClosky. et al. (2006) and Petrov et al. (2010), except that we use an extrinsic quality metric instead of a second parsing model for making the selection. It is also similar to Burkett and Klein (2008) and Burkett et al. (2010), but again avoiding the added complexity introduced by the use of additional (bilingual) models for candidate selection.

It should be noted that our extrinsic metric is computed from data that has been manually annotated with reference word reorderings. Details of the reordering metric and the annotated data we used are given in Sections 3 and 4. While this annotation requires some effort, such annotations are much easier to obtain than full parse trees. In our experiments in Section 6 we show that we can obtain similar improvements on downstream translation quality by targeted self-training with weakly labeled data (in form of word reorderings), as with training on the fully labeled data (with full syntactic parse trees).

2 Targeted Self-Training

Our technique for retraining a baseline parser is an extension of self-training. In standard parser self-training, one uses the baseline parsing model to parse a corpus of sentences, and then adds the 1-best output of the baseline parser to the training data. To target the self-training, we introduce an additional step, given as Algorithm 1. Instead of taking the 1-best parse, we produce a ranked n-best list of predictions and select the parser which gives the best score according to an external evaluation function. That is, instead of relying on the intrinsic model score, we use an extrinsic score to select the parse towards which to update. In the case of a tie, we prefer the parse ranked most highly in the n-best list.

Algorithm 1 Select parse that maximizes an extrinsic metric.

Input: baseline parser B
Input: sentence S
Input: function COMPUTEEXTRINSIC(parse P)
Output: a parse for the input sentence

```
P^n = {P_1, ..., P_n}   ← n-best parses of S by B
maxScore = 0
bestParse = ∅
for k = 1 to n do
    extrinsicScore = COMPUTEEXTRINSIC(P_k)
    if extrinsicScore > maxScore then
        maxScore = extrinsicScore
        bestParse = P_k
    end if
end for
return bestParse
```

The motivation of this selection step is that good performance on the downstream external task, measured by the extrinsic metric, should be predictive of an intrinsically good parse. At the very least, even if the selected parse is not syntactically correct, or even if it goes against the original treebanking guidelines, it results in a higher extrinsic score and should therefore be preferred.

One could imagine extending this framework by repeatedly running self-training on successively improving parsers in an EM-style algorithm. A recent work by Hall et al. (2011) on training a parser with multiple objective functions investigates a similar idea in the context of online learning.

In this paper we focus our attention on machine translation as the final application, but one could envision applying our techniques to other applications such as information extraction or question answering. In particular, we explore one application of targeted self-training, where computing the extrinsic metric involves plugging the parse into an MT system's reordering component and computing the accuracy of the reordering compared to a reference word order. We now direct our attention to the details of this application.

3 The MT Reordering Task

Determining appropriate target language word order for a translation is a fundamental problem in MT. When translating between languages with significantly different word order such as English and Japanese, it has been shown that metrics which explicitly account for word-order are much better correlated with human judgments of translation quality than those that give more weight to word choice, like BLEU (Lavie and Denkowski, 2009; Isozaki et al., 2010a; Birch and Osborne, 2010). This demonstrates the importance of getting reordering right.

3.1 Reordering as a separately evaluable component

One way to break down the problem of translating between languages with different word order is to handle reordering and translation separately: first reorder source-language sentences into target-language word order in a preprocessing step, and then translate the reordered sentences. It has been shown that good results can be achieved by reordering each input sentence using a series of tree transformations on its parse tree. The rules for tree transformation can be manually written (Collins et al., 2005; Wang, 2007; Xu et al., 2009) or automatically learned (Xia and McCord, 2004; Habash, 2007; Genzel, 2010).

Doing reordering as a preprocessing step, separately from translation, makes it easy to evaluate reordering performance independently from the MT system. Accordingly, Talbot et al. (2011) present a framework for evaluating the quality of reordering separately from the lexical choice involved in translation. They propose a simple *reordering metric* based on METEOR's reordering penalty (Lavie and Denkowski, 2009). This metric is computed solely on the source language side. To compute it, one takes the candidate reordering of the input sentence and partitions it into a set $\mathcal{C}$ of contiguous spans whose content appears contiguously in the same order in the reference. The reordering score is then computed as

$$\rho(\mathbf{e}_{\text{sys}}, \mathbf{e}_{\text{ref}}) = 1 - \frac{|\mathcal{C}| - 1}{|\mathbf{e}| - 1}.$$

This metric assigns a score between 0 and 1 where 1 indicates that the candidate reordering is identical to the reference and 0 indicates that no two words that are contiguous in the candidate reordering are contiguous in the reference. For example, if a reference reordering is `A B C D E`, candidate reordering `A B E C D` would get score $1-(3-1)/(5-1) = 0.5$.

Talbot et al. (2011) show that this reordering score is strongly correlated with human judgment of translation quality. Furthermore, they propose to evaluate the reordering quality of an MT system by computing its reordering score on a test set consisting of source language sentences and their reference reorderings. In this paper, we take the same approach for evaluation, and in addition, we use corpora of source language sentences and their reference reorderings for training the system, not just testing it. We describe in more detail how the reference reordering data was prepared in Section 4.1.

3.2 Reordering quality as predictor of parse quality

Figure 1 gives concrete examples of good and bad reorderings of an English sentence into Japanese word order. It shows that a bad parse leads to a bad reordering (lacking inversion of verb "wear" and object "sunscreen") and a low reordering score. Could we flip this causality around, and perhaps try to identify a good parse tree based on its reordering score? With the experiments in this paper, we show that indeed a high reordering score is predictive of the underlying parse tree that was used to generate the reordering being a good parse (or, at least, being good enough for our purpose).

In the case of translating English to Japanese or another SOV language, there is a large amount of reordering required, but with a relatively small number of reordering rules one can cover a large proportion of reordering phenomena. Isozaki et al. (2010b), for instance, were able to get impressive English→Japanese results with only a single reordering rule, given a suitable definition of a *head*. Hence, the reordering task depends crucially on a correct syntactic analysis and is extremely sensitive to parser errors.

4 Experimental Setup

4.1 Treebank data

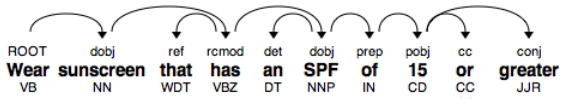
In our experiments the baseline training corpus is the Wall Street Journal (WSJ) section of the Penn Treebank (Marcus et al., 1993) using standard training/development/testing splits. We converted the treebank to match the tokenization expected by our MT system. In particular, we split tokens containing hyphens into multiple tokens and, somewhat simplistically, gave the original token's part-of-speech tag to all newly created tokens. In Section 6 we make also use of the Question Treebank (QTB) (Judge et al., 2006), as a source of syntactically annotated out-of-domain data. Though we experiment with both dependency parsers and phrase structure parsers, our MT system assumes dependency parses as input. We use the Stanford converter (de Marneffe et al., 2006) to convert phrase structure parse trees to dependency parse trees (for both treebank trees and predicted trees).

4.2 Reference reordering data

We aim to build an MT system that can accurately translate typical English text that one finds on the Internet to SOV langauges. To this end, we randomly sampled 13595 English sentences from the web and created Japanese-word-order reference reorderings for them. We split the sentences arbitrarily into a 6268-sentence Web-Train corpus and a 7327-sentence Web-Test corpus.

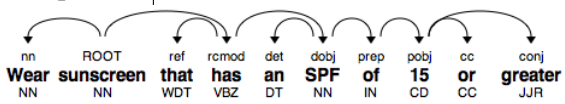
To make the reference alignments we used the technique suggested by Talbot et al. (2011): ask annotators to translate each English sentence to Japanese extremely literally and annotate which English words align to which Japanese words. Golden reference reorderings can be made programmatically from these annotations. Creating a large set of reference reorderings is straightforward because annotators need little special background or training, as long as they can speak both the source and target languages. We chose Japanese as the target language through which to create the English reference reorderings because we had access to bilingual annotators fluent in English and Japanese.

Good parse



Reordered:
15 or greater of an SPF has that sunscreen [*Wear*]
Reordering score: 1.0 (matches reference)

Bad parse



Reordered:
15 or greater of an SPF has that [*Wear*] *sunscreen*
Reordering score: 0.78 ("Wear" is out of place)

Figure 1: Examples of good and bad parses and corresponding reorderings for translation from English to Japanese. The good parse correctly identifies "Wear" as the main verb and moves it to the end of the sentence; the bad parse analyses "Wear sunscreen" as a noun phrase and does not reorder it. This example was one of the wins in the human evaluation of Section 5.2.

4.3 Parsers

The core dependency parser we use is an implementation of a transition-based dependency parser using an arc-eager transition strategy (Nivre, 2008). The parser is trained using the averaged perceptron algorithm with an early update strategy as described in Zhang and Clark (2008). The parser uses the following features: word identity of the first two words on the buffer, the top word on the stack and the head of the top word on the stack (if available); part-of-speech identities of the first four words on the buffer and top two words on the stack; dependency arc label identities for the top word on the stack, the left and rightmost modifier of the top word on the stack, and the leftmost modifier of the first word in the buffer. We also include conjunctions over all non-lexical features.

We also give results for the latent variable parser (a.k.a. BerkeleyParser) of Petrov et al. (2006). We convert the constituency trees output by the BerkeleyParser to labeled dependency trees using the same procedure that is applied to the treebanks.

While the BerkeleyParser views part-of-speech (POS) tagging as an integral part of parsing, our dependency parser requires the input to be tagged

with a separate POS tagger. We use the TnT tagger (Brants, 2000) in our experiments, because of its efficiency and ease of use. Tagger and parser are always trained on the same data.

For all parsers, we lowercase the input at train and test time. We found that this improves performance in parsing web text. In addition to general uppercase/lowercase noisiness of the web text negatively impacting scores, we found that the baseline case-sensitive parsers are especially bad at parsing imperative sentences, as discussed in Section 5.3.2.

4.4 Reordering rules

In this paper we focus on English to Japanese, Korean, and Turkish translation. We use a superset of the reordering rules proposed by Xu et al. (2009), which flatten a dependency tree into SOV word order that is suitable for all three languages. The rules define a precedence order for the dependents of each part of speech. For example, a slightly simplified version of the precedence order of child labels for a verbal head HEADVERB is: advcl, nsubj, prep, [other children], dobj, prt, aux, neg, HEADVERB, mark, ref, compl.

Alternatively, we could have used an automatic reordering-rule learning framework like that of Genzel (2010). Because the reordering accuracy metric can be computed for any source/target language pair, this would have made our approach language completely independent and applicable to any language pair. We chose to use manually written rules to eliminate the variance induced by the automatic reordering-rule learning framework.

4.5 MT system

We carried out all our translation experiments on a state-of-the-art phrase-based statistical MT system. During both training and testing, the system reorders source-language sentences in a preprocessing step using the above-mentioned rules. During decoding, we used an allowed jump width of 4 words. In addition to the regular distance distortion model, we incorporate a maximum entropy based lexicalized phrase reordering model (Zens and Ney, 2006) as a feature used in decoding.

Overall for decoding, we use between 20 to 30 features, whose weights are optimized using MERT (Och, 2003). All experiments for a given language pair use the same set of MERT weights tuned on a system using a separate parser (that is neither the baseline nor the experiment parser). This potentially underestimates the improvements that can be obtained, but also eliminates MERT as a possible source of improvement, allowing us to trace back improvements in translation quality directly to parser changes.[2]

For parallel training data, we use a custom collection of parallel documents. They come from various sources with a substantial portion coming from the web after using simple heuristics to identify potential document pairs. For all language pairs, we trained on approximately 300 million source words each.

5 Experiments Reordering Web Text

We experimented with parsers trained in three different ways:

1. Baseline: trained only on WSJ-Train.
2. Standard self-training: trained on WSJ-Train and 1-best parse of the Web-Train set by baseline parser.
3. Targeted self-training: trained on WSJ-Train and, for each sentence in Web-Train, the parse from the baseline parser's 512-best list that when reordered gives the highest reordering score.[3]

5.1 Standard self-training vs targeted self-training

Table 1 shows that targeted self-training on Web-Train significantly improves Web-Test reordering score more than standard self-training for both the shift-reduce parser and for the BerkeleyParser. The reordering score is generally divorced from the attachment scores measured on the WSJ-Test treebank: for the shift-reduce parser, Web-Test reordering score and WSJ-Test labeled attachment score

[2] We also ran MERT on all systems and the pattern of improvement is consistent, but sometimes the improvement is bigger or smaller after MERT. For instance, the BLEU delta for Japanese is +0.0030 with MERT on both sides as opposed to +0.0025 with no MERT.

[3] We saw consistent but diminishing improvements as we increased the size of the n-best list.

Parser	Web-Test reordering	WSJ-Test LAS
Shift-reduce WSJ baseline	0.757	85.31%
+ self-training 1x	0.760	85.26%
+ self-training 10x	0.756	84.14%
+ targeted self-training 1x	**0.770**	85.19%
+ targeted self-training 10x	**0.777**	84.48%
Berkeley WSJ baseline	0.780	88.66%
+ self-training 1x	0.785	89.21%
+ targeted self-training 1x	**0.790**	89.32%

Table 1: English→Japanese reordering scores on Web-Test for standard self-training and targeted self-training on Web-Train. Label "10x" indicates that the self-training data was weighted 10x relative to the WSJ training data. Bolded reordering scores are different from WSJ-only baseline with 95% confidence but are not significantly different from each other within the same group.

English to	BLEU		Human evaluation (scores range 0 to 6)		
	WSJ-only	Targeted	WSJ-only	Targeted	Sig. difference?
Japanese	0.1777	0.1802	2.56	2.69	yes (at 95% level)
Korean	0.3229	0.3259	2.61	2.70	yes (at 90% level)
Turkish	0.1344	0.1370	2.10	2.20	yes (at 95% level)

Table 2: BLEU scores and human evaluation results for translation between three language pairs, varying only the parser between systems. "WSJ-only" corresponds to the baseline WSJ-only shift-reduce parser; "Targeted" corresponds to the Web-Train targeted self-training 10x shift-reduce parser.

(LAS) are anti-correlated, but for BerkeleyParser they are correlated. Interestingly, weighting the self-training data more seems to have a negative effect on both metrics.[4]

One explanation for the drops in LAS is that some parts of the parse tree are important for downstream reordering quality while others are not (or only to a lesser extent). Some distinctions between labels become less important; for example, arcs labeled "amod" and "advmod" are transformed identically by the reordering rules. Some semantic distinctions also become less important; for example, any sane interpretation of "red hot car" would be reordered the same, that is, not at all.

5.2 Translation quality improvement

To put the improvement of the MT system in terms of BLEU score (Papineni et al., 2002), a widely used metric for automatic MT evaluation, we took 5000 sentences from Web-Test and had humans generate reference translations into Japanese, Korean, and Turkish. We then trained MT systems varying only the parser used for reordering in training and decoding. Table 2 shows that targeted self-training data increases BLEU score for translation into all three languages.

In addition to BLEU increase, a side-by-side human evaluation on 500 sentences (sampled from the 5000 used to compute BLEU scores) showed a statistically significant improvement for all three languages (see again Table 2). For each sentence, we asked annotators to simultaneously score both translations from 0 to 6, with guidelines that 6="Perfect", 4="Most Meaning/Grammar", 2="Some Meaning/Grammar", 0="Nonsense". We computed confidence intervals for the average score difference using bootstrap resampling; a difference is significant if the two-sided confidence interval does not include 0.

5.3 Analysis

As the divergence between the labeled attachment score on the WSJ-Test data and the reordering score on the WSJ-Test data indicates, parsing web text

[4]We did not attempt this experiment for the BerkeleyParser since training was too slow.

Parser	Click as N	Click as V	Imperative rate
case-sensitive shift-reduce WSJ-only	74	0	6.3%
case-sensitive shift-reduce + Web-Train targeted self-training	75	0	10.5%
case-insensitive shift-reduce WSJ-only	75	0	10.3%
case-insensitive shift-reduce + Web-Train targeted self-training	75	0	11.6%
Berkeley WSJ-only	35	35	11.9%
Berkeley + Web-Train targeted self-training	13	58	12.5%
(WSJ-Train)	1	0	0.7%

Table 3: Counts on Web-Test of "click" tagged as a noun and verb and percentage of sentences parsed imperatively.

poses very different challenges compared to parsing newswire. We show how our method improves parsing performance and reordering performance on two examples: the trendy word "click" and imperative sentences.

5.3.1 Click

The word "click" appears only once in the training portion of the WSJ (as a noun), but appears many times in our Web test data. Table 3 shows the distribution of part-of-speech tags that different parsers assign to "click". The WSJ-only parsers tag "click" as a noun far too frequently. The WSJ-only shift-reduce parser refuses to tag "click" as a verb even with targeted self-training, but BerkeleyParser does learn to tag "click" more often as a verb.

It turns out that the shift-reduce parser's stubbornness is not due to a fundamental problem of the parser, but due to an artifact in TnT. To increase speed, TnT restricts the choices of tags for known words to previously-seen tags. This causes the parser's n-best lists to never hypothesize "click" as a verb, and self-training doesn't click no matter how targeted it is. This shows that the targeted self-training approach heavily relies on the diversity of the baseline parser's n-best lists.

It should be noted here that it would be easy to combine our approach with the *uptraining* approach of Petrov et al. (2010). The idea would be to use the BerkeleyParser to generate the n-best lists; perhaps we could call this *targeted uptraining*. This way, the shift-reduce parser could benefit both from the generally higher quality of the parse trees produced by the BerkeleyParser, as well as from the information provided by the extrinsic scoring function.

5.3.2 Imperatives

As Table 3 shows, the WSJ training set contains only 0.7% imperative sentences.[5] In contrast, our test sentences from the web contain approximately 10% imperatives. As a result, parsers trained exclusively on the WSJ underproduce imperative parses, especially a case-sensitive version of the baseline. Targeted self-training helps the parsers to predict imperative parses more often.

Targeted self-training works well for generating training data with correctly-annotated imperative constructions because the reordering of main subjects and verbs in an SOV language like Japanese is very distinct: main subjects stay at the beginning of the sentence, and main verbs are reordered to the end of the sentence. It is thus especially easy to know whether an imperative parse is correct or not by looking at the reference reordering. Figure 1 gives an example: the bad (WSJ-only) parse doesn't catch on to the imperativeness and gets a low reordering score.

6 Targeted Self-Training vs Training on Treebanks for Domain Adaptation

If task-specific annotation is cheap, then it is reasonable to consider whether we could use targeted self-training to adapt a parser to a new domain as a cheaper alternative to making new treebanks. For example, if we want to build a parser that can reorder question sentences better than our baseline WSJ-only parser, we have these two options:

1. Manually construct PTB-style trees for 2000

[5] As an approximation, we count every parse that begins with a root verb as an imperative.

questions and train on the resulting treebank.

2. Create reference reorderings for 2000 questions and then do targeted self-training.

To compare these approaches, we created reference reordering data for our train (2000 sentences) and test (1000 sentences) splits of the Question Treebank (Judge et al., 2006). Table 4 shows that both ways of training on QTB-Train sentences give similarly large improvements in reordering score on QTB-Test. Table 5 confirms that this corresponds to very large increases in English→Japanese BLEU score and subjective translation quality. In the human side-by-side comparison, the baseline translations achieved an average score of 2.12, while the targeted self-training translations received a score of 2.94, where a score of 2 corresponds to "some meaning/grammar" and "4" corresponds to "most meaning/grammar".

But which of the two approaches is better? In the shift-reduce parser, targeted self-training gives higher reordering scores than training on the treebank, and in BerkeleyParser, the opposite is true. Thus both approaches produce similarly good results. From a practical perspective, the advantage of targeted self-training depends on whether the extrinsic metric is cheaper to calculate than treebanking. For MT reordering, making reference reorderings is cheap, so targeted self-training is relatively advantageous.

As before, we can examine whether labeled attachment score measured on the test set of the QTB is predictive of reordering quality. Table 4 shows that targeted self-training raises LAS from 64.78→69.17%. But adding the treebank leads to much larger increases, resulting in an LAS of 84.75%, without giving higher reordering score. We can conclude that high LAS is not necessary to achieve top reordering scores.

Perhaps our reordering rules are somehow deficient when it comes to reordering correctly-parsed questions, and as a result the targeted self-training process steers the parser towards producing pathological trees with little intrinsic meaning. To explore this possibility, we computed reordering scores after reordering the QTB-Test treebank trees directly. Table 4 shows that this gives reordering scores similar to those of our best parsers. Therefore it is at least possible that the targeted self-training process could have resulted in a parser that achieves high reordering score by producing parses that look like those in the QuestionBank.

7 Related Work

Our approach to training parsers for reordering is closely related to self/up-training (McClosky. et al., 2006; Petrov et al., 2010). However, unlike uptraining, our method does not use only the 1-best output of the first-stage parser, but has access to the n-best list. This makes it similar to the work of McClosky. et al. (2006), except that we use an extrinsic metric (MT reordering score) to select a high quality parse tree, rather than a second, reranking model that has access to additional features.

Targeted self-training is also similar to the retraining of Burkett et al. (2010) in which they jointly parse unannotated bilingual text using a multiview learning objective, then retrain the monolingual parser models to include each side of the jointly parsed bitext as monolingual training data. Our approach is different in that it doesn't use a second parser and bitext to guide the creation of new training data, and instead relies on n-best lists and an extrinsic metric.

Our method can be considered an instance of weakly or distantly supervised structured prediction (Chang et al., 2007; Chang et al., 2010; Clarke et al., 2010; Ganchev et al., 2010). Those methods attempt to learn structure models from related external signals or aggregate data statistics. This work differs in two respects. First, we use the external signals not as explicit constraints, but to compute an oracle score used to re-rank a set of parses. As such, there are no requirements that it factor by the structure of the parse tree and can in fact be any arbitrary metric. Second, our final objective is different. In weakly/distantly supervised learning, the objective is to use external knowledge to build better structured predictors. In our case this would mean using the reordering metric as a means to train better dependency parsers. Our objective, on the other hand, is to use the extrinsic metric to train parsers that are specifically better at the reordering task, and, as a result, better suited for MT. This makes our work more in the spirit of Liang et al. (2006), who train a per-

Parser	QTB-Test reordering	QTB-Test LAS
Shift-reduce WSJ baseline	0.663	64.78%
+ treebank 1x	0.704	77.12%
+ treebank 10x	**0.768**	84.75%
+ targeted self-training 1x	0.746	67.84%
+ targeted self-training 10x	**0.779**	69.17%
Berkeley WSJ baseline	0.733	76.50%
+ treebank 1x	**0.800**	87.79%
+ targeted self-training 1x	**0.775**	80.64%
(using treebank trees directly)	0.788	100%

Table 4: Reordering and labeled attachment scores on QTB-Test for treebank training and targeted self-training on QTB-Train.

English to	QTB-Test BLEU		Human evaluation (scores range 0 to 6)		
	WSJ-only	Targeted	WSJ-only	Targeted	Sig. difference?
Japanese	0.2379	0.2615	2.12	2.94	yes (at 95% level)

Table 5: BLEU scores and human evaluation results for English→Japanese translation of the QTB-Test corpus, varying only the parser between systems between the WSJ-only shift-reduce parser and the QTB-Train targeted self-training 10x shift-reduce parser.

ceptron model for an end-to-end MT system where the alignment parameters are updated based on selecting an alignment from a n-best list that leads to highest BLEU score. As mentioned earlier, this also makes our work similar to Hall et al. (2011) who train a perceptron algorithm on multiple objective functions with the goal of producing parsers that are optimized for extrinsic metrics.

It has previously been observed that parsers often perform differently for downstream applications. Miyao et al. (2008) compared parser quality in the biomedical domain using a protein-protein interaction (PPI) identification accuracy metric. This allowed them to compare the utility of extant dependency parsers, phrase structure parsers, and deep structure parsers for the PPI identification task. One could apply the targeted self-training technique we describe to optimize any of these parsers for the PPI task, similar to how we have optimized our parser for the MT reordering task.

8 Conclusion

We introduced a variant of self-training that targets parser training towards an extrinsic evaluation metric. We use this targeted self-training approach to train parsers that improve the accuracy of the word reordering component of a machine translation system. This significantly improves the subjective quality of the system's translations from English into three SOV languages. While the new parsers give improvements in these external evaluations, their intrinsic attachment scores go down overall compared to baseline parsers trained only on treebanks. We conclude that when using a parser as a component of a larger external system, it can be advantageous to incorporate an extrinsic metric into parser training and evaluation, and that targeted self-training is an effective technique for incorporating an extrinsic metric into parser training.

References

A. Birch and M. Osborne. 2010. LRscore for evaluating lexical and reordering quality in MT. In *ACL-2010 WMT*.

T. Brants. 2000. TnT – a statistical part-of-speech tagger. In *ANLP '00*.

D. Burkett and D. Klein. 2008. Two languages are better than one (for syntactic parsing). In *EMNLP '08*.

D. Burkett, S. Petrov, J. Blitzer, and D. Klein. 2010. Learning better monolingual models with unannotated bilingual text. In *CoNLL '10*.

M. Chang, L. Ratinov, and D. Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *ACL '07*.

M. Chang, D. Goldwasser, D. Roth, and V. Srikumar. 2010. Structured output learning with indirect supervision. In *ICML '10*.

E. Charniak. 2000. A maximum–entropy–inspired parser. In *NAACL '00*.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *CoNLL '10*.

M. Collins, P. Koehn, and I. Kučerová. 2005. Clause restructuring for statistical machine translation. In *ACL '05*.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *ACL '97*.

M.-C. de Marneffe, B. MacCartney, and C. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC '06*.

J. Foster. 2010. "cba to check the spelling": Investigating parser performance on discussion forum posts. In *NAACL '10*.

M. Galley, M. Hopkins, K. Knight, and D. Marcu. 2004. What's in a translation rule? In *HLT-NAACL '04*.

K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*.

D. Genzel. 2010. Automatically learning source-side reordering rules for large scale machine translation. In *COLING '10*.

D. Gildea. 2001. Corpus variation and parser performance. In *EMNLP '01*.

N. Habash. 2007. Syntactic preprocessing for statistical machine translation. In *MTS '07*.

K. Hall, R. McDonald, J. Katz-Brown, and M. Ringgaard. 2011. Training dependency parsers by jointly optimizing multiple objectives. In *EMNLP '11*.

H. Isozaki, T. Hirao, K. Duh, K. Sudoh, and H. Tsukada. 2010a. Automatic evaluation of translation quality for distant language pairs. In *EMNLP '10*.

H. Isozaki, K. Sudoh, H. Tsukada, and K. Duh. 2010b. Head finalization: A simple reordering rule for SOV languages. In *ACL-2010 WMT*.

J. Judge, A. Cahill, and J. v. Genabith. 2006. QuestionBank: creating a corpus of parse-annotated questions. In *ACL '06*.

A. Lavie and M. Denkowski. 2009. The Meteor metric for automatic evaluation of machine translation. *Machine Translation*, 23(2-3).

P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar. 2006. An end-to-end discriminative approach to machine translation. In *ACL '06*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. In *Computational Linguistics*.

D. McClosky., E. Charniak, and M. Johnson. 2006. Effective self-training for parsing. In *NAACL '06*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *ACL '05*.

Y. Miyao, R. Sætre, K. Sagae, T. Matsuzaki, and J. Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *ACL '08*.

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

F. Och. 2003. Minimum error rate training in statistical machine translation. In *ACL '03*.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL '02*.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *NAACL '07*.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL '06*.

S. Petrov, P. Chang, and M. Ringgaard H. Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *EMNLP '10*.

D. Talbot, H. Kazawa, H. Ichikawa, J. Katz-Brown, M. Seno, and F. Och. 2011. A lightweight evaluation framework for machine translation reordering. In *EMNLP-2011 WMT*.

C. Wang. 2007. Chinese syntactic reordering for statistical machine translation. In *EMNLP '07*.

F. Xia and M. McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *Coling '04*.

P. Xu, J. Kang, M. Ringgaard, and F. Och. 2009. Using a dependency parser to improve SMT for subject-object-verb languages. In *NAACL-HLT '09*.

K. Yamada and K. Knight. 2001. A syntax-based statistical translation model. In *ACL '01*.

R. Zens and H. Ney. 2006. Discriminative reordering models for statistical machine translation. In *NAACL-06 WMT*.

Y. Zhang and S. Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *EMNLP '08*.