Uptraining for Accurate Deterministic Question Parsing

Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, Hiyan Alshawi
Google Research
{slav,pichuan,ringgaard,hiyan}@google.com

Abstract

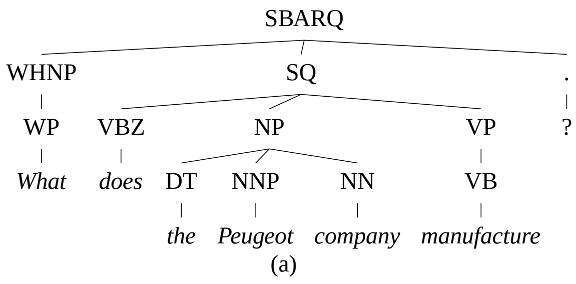
It is well known that parsing accuracies drop significantly on out-of-domain data. What is less known is that some parsers suffer more from domain shifts than others. We show that dependency parsers have more difficulty parsing questions than constituency parsers. In particular, deterministic shift-reduce dependency parsers, which are of highest interest for practical applications because of their linear running time, drop to 60% labeled accuracy on a question test set. We propose an *uptraining* procedure in which a deterministic parser is trained on the output of a more accurate, but slower, latent variable constituency parser (converted to dependencies). Uptraining with 100K unlabeled questions achieves results comparable to having 2K labeled questions for training. With 100K unlabeled and 2K labeled questions, uptraining is able to improve parsing accuracy to 84%, closing the gap between in-domain and out-of-domain performance.

1 Introduction

Parsing accuracies on the popular Section 23 of the Wall Street Journal (WSJ) portion of the Penn Treebank have been steadily improving over the past decade. At this point, we have many different parsing models that reach and even surpass 90% dependency or constituency accuracy on this test set (McDonald et al., 2006; Nivre et al., 2007; Charniak and Johnson, 2005; Petrov et al., 2006; Carreras et al., 2008; Koo and Collins, 2010). Quite impressively, models based on deterministic shift-reduce parsing algorithms are able to rival the other computationally more expensive models (see Nivre (2008) and references therein for more details). Their linear running time makes them ideal candidates for large scale text processing, and our model of choice for this paper.

Unfortunately, the parsing accuracies of all models have been reported to drop significantly on out-of-domain test sets, due to shifts in vocabulary and grammar usage (Gildea, 2001; McClosky et al., 2006b; Foster, 2010). In this paper, we focus our attention on the task of parsing questions. Questions pose interesting challenges for WSJ-trained parsers because they are heavily underrepresented in the training data (there are only 334 questions among the 39,832 training sentences). At the same time, questions are of particular interest for user facing applications like question answering or web search, which necessitate parsers that can process questions in a fast and accurate manner.

We start our investigation in Section 3 by training several state-of-the-art (dependency and constituency) parsers on the standard WSJ training set. When evaluated on a question corpus, we observe dramatic accuracy drops exceeding 20% for the deterministic shift-reduce parsers. In general, dependency parsers (McDonald et al., 2006; Nivre et al., 2007), seem to suffer more from this domain change than constituency parsers (Charniak and Johnson, 2005; Petrov et al., 2006). Overall, the latent variable approach of Petrov et al. (2006) appears to generalize best to this new domain, losing only about 5%. Unfortunately, the parsers that generalize better to this new domain have time complexities that are cubic in the sentence length (or even higher), rendering them impractical for web-scale text processing.



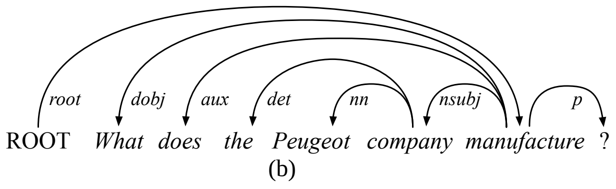


Figure 1: Example constituency tree from the QuestionBank (a) converted to labeled Stanford dependencies (b).

We therefore propose an *uptraining* method, in which a deterministic shift-reduce parser is trained on the output of a more accurate, but slower parser (Section 4). This type of domain adaptation is reminiscent of self-training (McClosky et al., 2006a; Huang and Harper, 2009) and co-training (Blum and Mitchell, 1998; Sagae and Lavie, 2006), except that the goal here is not to further improve the performance of the very best model. Instead, our aim is to train a computationally cheaper model (a linear time dependency parser) to match the performance of the best model (a cubic time constituency parser), resulting in a computationally efficient, yet highly accurate model.

In practice, we parse a large amount of unlabeled data from the target domain with the constituency parser of Petrov et al. (2006) and then train a deterministic dependency parser on this noisy, automatically parsed data. The accuracy of the linear time parser on a question test set goes up from 60.06% (LAS) to 76.94% after uptraining, which is comparable to adding 2,000 labeled questions to the training data. Combining uptraining with 2,000 labeled questions further improves the accuracy to 84.14%, fully recovering the drop between in-domain and out-of-domain accuracy.

We also present a detailed error analysis in Section 5, showing that the errors of the WSJ-trained model are primarily caused by sharp changes in syntactic configurations and only secondarily due to lexical shifts. Uptraining leads to large improvements across all error metrics and especially on important dependencies like subjects (nsubj).

2 Experimental Setup

We used the following experimental protocol throughout the paper.

2.1 Data

Our main training set consists of Sections 02-21 of the Wall Street Journal portion of the Penn Treebank (Marcus et al., 1993), with Section 22 serving as development set for source domain comparisons. For our target domain experiments, we evaluate on the QuestionBank (Judge et al., 2006), which includes a set of manually annotated questions from a TREC question answering task. The questions in the QuestionBank are very different from our training data in terms of grammatical constructions and vocabulary usage, making this a rather extreme case of domain-adaptation. We split the 4,000 questions contained in this corpus in three parts: the first 2,000 questions are reserved as a small target-domain training set; the remaining 2,000 questions are split in two equal parts, the first serving as development set and the second as our final test set. We report accuracies on the developments sets throughout this paper, and test only at the very end on the final test set.

We convert the trees in both treebanks from constituencies to labeled dependencies (see Figure 1) using the Stanford converter, which produces 46 types of labeled dependencies[1] (de Marneffe et al., 2006). We evaluate on both unlabeled (UAS) and labeled dependency accuracy (LAS).[2]

Additionally, we use a set of 2 million questions collected from Internet search queries as unlabeled target domain data. All user information was anonymized and only the search query string was retained. The question sample is selected at random after passing two filters that select queries that are

[1]We use the Stanford Lexicalized Parser v1.6.2.

[2]Because the QuestionBank does not contain function tags, we decided to strip off the function tags from the WSJ before conversion. The Stanford conversion only uses the -ADV and -TMP tags, and removing all function tags from the WSJ changed less than 0.2% of the labels (primarily tmod labels).

Training on WSJ Sections 02-21	Evaluating on WSJ Section 22				Evaluating on QuestionBank			
	F_1	UAS	LAS	POS	F_1	UAS	LAS	POS
Nivre et al. (2007)	—	88.42	84.89	95.00	—	74.14	62.81	88.48
McDonald et al. (2006)	—	89.47	86.43	95.00	—	80.01	67.00	88.48
Charniak (2000)	90.27	92.33	89.86	96.71	83.01	85.61	73.59	90.49
Charniak and Johnson (2005)	91.92	93.56	91.24	96.69	84.47	87.13	75.94	90.59
Petrov et al. (2006)	90.70	92.91	90.48	96.27	85.52	88.17	79.10	90.57
Petrov (2010)	92.10	93.85	91.60	96.44	86.62	88.77	79.92	91.08
Our shift-reduce parser	—	88.24	84.69	95.00	—	72.23	60.06	88.48
Our shift-reduce parser (gold POS)	—	90.51	88.53	100.00	—	78.30	68.92	100.00

Table 1: Parsing accuracies for parsers trained on newswire data and evaluated on newswire and question test sets.

similar in style to the questions in the QuestionBank: (i) the queries must start with an English function word that can be used to start a question (what, who when, how, why, can, does, etc.), and (ii) the queries have a maximum length of 160 characters.

2.2 Parsers

We use multiple publicly available parsers, as well as our own implementation of a deterministic shift-reduce parser in our experiments. The dependency parsers that we compare are the deterministic shift-reduce MaltParser (Nivre et al., 2007) and the second-order minimum spanning tree algorithm based MstParser (McDonald et al., 2006). Our shift-reduce parser is a re-implementation of the MaltParser, using a standard set of features and a linear kernel SVM for classification. We also train and evaluate the generative lexicalized parser of Charniak (2000) on its own, as well as in combination with the discriminative reranker of Charniak and Johnson (2005). Finally, we run the latent variable parser (a.k.a. BerkeleyParser) of Petrov et al. (2006), as well as the recent product of latent variable grammars version (Petrov, 2010). To facilitate comparisons between constituency and dependency parsers, we convert the output of the constituency parsers to labeled dependencies using the same procedure that is applied to the treebanks. We also report their F_1 scores for completeness.

While the constituency parsers used in our experiments view part-of-speech (POS) tagging as an integral part of parsing, the dependency parsers require 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.

3 Parsing Questions

We consider two domain adaptation scenarios in this paper. In the first scenario (sometimes abbreviated as WSJ), we assume that we do not have any labeled training data from the target domain. In practice, this will always be the case when the target domain is unknown or very diverse. The second scenario (abbreviated as WSJ+QB) assumes a small amount of labeled training data from the target domain. While this might be expensive to obtain, it is certainly feasible for narrow domains (e.g. questions), or when a high parsing accuracy is really important.

3.1 No Labeled Target Domain Data

We first trained all parsers on the WSJ training set and evaluated their performance on the two domain specific evaluation sets (newswire and questions). As can be seen in the left columns of Table 1, all parsers perform very well on the WSJ development set. While there are differences in the accuracies, all scores fall within a close range. The table also confirms the commonly known fact (Yamada and Matsumoto, 2003; McDonald et al., 2005) that constituency parsers are more accurate at producing dependencies than dependency parsers (at least when the dependencies were produced by a deterministic transformation of a constituency treebank, as is the case here).

This picture changes drastically when the performance is measured on the QuestionBank development set (right columns in Table 1). As one

Evaluating on QuestionBank	Training on WSJ + QB				Training on QuestionBank			
	F_1	UAS	LAS	POS	F_1	UAS	LAS	POS
Nivre et al. (2007)	—	83.54	78.85	91.32	—	79.72	73.44	88.80
McDonald et al. (2006)	—	84.95	80.17	91.32	—	82.52	77.20	88.80
Charniak (2000)	89.40	90.30	85.01	94.17	79.70	76.69	69.69	87.84
Petrov et al. (2006)	90.96	90.98	86.90	94.01	86.62	84.09	78.92	87.56
Petrov (2010)	92.81	92.23	88.84	94.48	87.72	85.07	80.08	87.79
Our shift-reduce parser	—	83.70	78.27	91.32	—	80.44	74.29	88.80
Our shift-reduce parser (gold POS)	—	89.39	86.60	100.00	—	87.31	84.15	100.00

Table 2: Parsing accuracies for parsers trained on newswire and question data and evaluated on a question test set.

might have expected, the accuracies are significantly lower, however, the drop for some of the parsers is shocking. Most notably, the deterministic shift-reduce parsers lose almost 25% (absolute) on labeled accuracies, while the latent variable parsers lose around 12%.[3] Note also that even with gold POS tags, LAS is below 70% for our deterministic shift-reduce parser, suggesting that the drop in accuracy is primarily due to a syntactic shift rather than a lexical shift. These low accuracies are especially disturbing when one considers that the average question in the evaluation set is only nine words long and therefore potentially much less ambiguous than WSJ sentences. We will examine the main error types more carefully in Section 5.

Overall, the dependency parsers seem to suffer more from the domain change than the constituency parsers. One possible explanation is that they lack the global constraints that are enforced by the (context-free) grammars. Even though the MstParser finds the globally best spanning tree, all constraints are local. This means for example, that it is not possible to require the final parse to contain a verb (something that can be easily expressed by a top-level production of the form $S \rightarrow NP\ VP$ in a context free grammar). This is not a limitation of dependency parsers in general. For example, it would be easy to enforce such constraints in the Eisner (1996) algorithm or using Integer Linear Programming approaches (Riedel and Clarke, 2006; Martins et al., 2009). However, such richer modeling capacity comes with a much higher computational cost.

Looking at the constituency parsers, we observe that the lexicalized (reranking) parser of Charniak and Johnson (2005) loses more than the latent variable approach of Petrov et al. (2006). This difference doesn't seem to be a difference of generative vs. discriminative estimation. We suspect that the latent variable approach is better able to utilize the little evidence in the training data. Intuitively speaking, some of the latent variables seem to get allocated for modeling the few questions present in the training data, while the lexicalization contexts are not able to distinguish between declarative sentences and questions.

To verify this hypothesis, we conducted two additional experiments. In the first experiment, we collapsed the question specific phrasal categories SQ and SBARQ to their declarative sentence equivalents S and SBAR. When the training and test data are processed this way, the lexicalized parser loses 1.5% F_1, while the latent variable parser loses only 0.7%. It is difficult to examine the grammars, but one can speculate that some of the latent variables were used to model the question specific constructions and the model was able to re-learn the distinctions that we purposefully collapsed. In the second experiment, we removed all questions from the WSJ training set and retrained both parsers. This did not make a significant difference when evaluating on the WSJ development set, but of course resulted in a large performance drop when evaluating on the QuestionBank. The lexicalized parser came out ahead in this experiment,[4] confirming our hypothesis that the latent variable model is better able to pick up the small amount of relevant evidence that is present in the WSJ training data (rather than being systematically

[3]The difference between our shift-reduce parser and the MaltParser are due to small differences in the feature sets.

[4]The F_1 scores were 52.40% vs. 56.39% respectively.

better suited for modeling questions).

3.2 Some Labeled Target Domain Data

In the above experiments, we considered a situation where we have no labeled training data from the target domain, as will typically be the case. We now consider a situation where a small amount of labeled data (2,000 manually parsed sentences) from the domain of interest is available for training.

We experimented with two different ways of utilizing this additional training data. In a first experiment, we trained models on the concatenation of the WSJ and QuestionBank training sets (we did not attempt to weight the different corpora). As Table 2 shows (left columns), even a modest amount of labeled data from the target domain can significantly boost parsing performance, giving double-digit improvements in some cases. While not shown in the table, the parsing accuracies on the WSJ development set where largely unaffected by the additional training data.

Alternatively, one can also train models exclusively on the QuestionBank data, resulting in question specific models. The parsing accuracies of these domain-specific models are shown in the right columns of Table 2, and are significantly lower than those of models trained on the concatenated training sets. They are often times even lower than the results of parsers trained exclusively on the WSJ, indicating that 2,000 sentences are not sufficient to train accurate parsers, even for quite narrow domains.

4 Uptraining for Domain-Adaptation

The results in the previous section suggest that parsers without global constraints have difficulties dealing with the syntactic differences between declarative sentences and questions. A possible explanation is that similar word configurations can appear in both types of sentences, but with very different syntactic interpretation. Local models without global constraints are therefore mislead into dead-end interpretations from which they cannot recover (McDonald and Nivre, 2007). Our approach will therefore be to use a large amount of unlabeled data to bias the model towards the appropriate distribution for the target domain. Rather than looking for feature correspondences between the domains

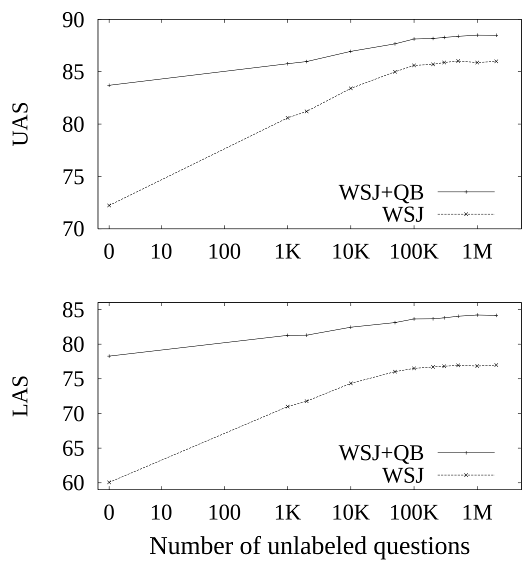


Figure 2: Uptraining with large amounts of unlabeled data gives significant improvements over two different supervised baselines.

(Blitzer et al., 2006), we propose to use automatically labeled target domain data to learn the target domain distribution directly.

4.1 Uptraining vs. Self-training

The idea of training parsers on their own output has been around for as long as there have been statistical parsers, but typically does not work well at all (Charniak, 1997). Steedman et al. (2003) and Clark et al. (2003) present co-training procedures for parsers and taggers respectively, which are effective when only very little labeled data is available. McClosky et al. (2006a) were the first to improve a state-of-the-art constituency parsing system by utilizing unlabeled data for self-training. In subsequent work, they show that the same idea can be used for domain adaptation if the unlabeled data is chosen accordingly (McClosky et al., 2006b). Sagae and Tsujii (2007) co-train two dependency parsers by adding automatically parsed sentences for which the parsers agree to the training data. Finally, Suzuki et al. (2009) present a very effective semi-supervised approach in which features from multiple generative models estimated on unlabeled data are combined in a discriminative system for structured prediction.

All of these approaches have in common that their ultimate goal is to improve the final performance. Our work differs in that instead of improving the

Uptraining with different base parsers	Using only WSJ data			Using WSJ + QB data		
	UAS	LAS	POS	UAS	LAS	POS
Baseline	72.23	60.06	88.48	83.70	78.27	91.32
Self-training	73.62	61.63	89.60	84.26	79.15	92.09
Uptraining on Petrov et al. (2006)	86.02	76.94	90.75	88.38	84.02	93.63
Uptraining on Petrov (2010)	85.21	76.19	90.74	88.63	84.14	93.53

Table 3: Uptraining substantially improves parsing accuracies, while self-training gives only minor improvements.

performance of the best parser, we want to build a more efficient parser that comes close to the accuracy of the best parser. To do this, we parse the unlabeled data with our most accurate parser and generate noisy, but fairly accurate labels (parse trees) for the unlabeled data. We refer to the parser used for producing the automatic labels as the base parser (unless otherwise noted, we used the latent variable parser of Petrov et al. (2006) as our base parser). Because the most accurate base parsers are constituency parsers, we need to convert the parse trees to dependencies using the Stanford converter (see Section 2). The automatically parsed sentences are appended to the labeled training data, and the shift-reduce parser (and the part-of-speech tagger) are trained on this new training set. We did not increase the weight of the WSJ training data, but weighted the QuestionBank training data by a factor of ten in the WSJ+QB experiments.

4.2 Varying amounts of unlabeled data

Figure 2 shows the efficacy of uptraining as a function of the size of the unlabeled data. Both labeled (LAS) and unlabeled accuracies (UAS) improve sharply when automatically parsed sentences from the target domain are added to the training data, and level off after 100,000 sentences. Comparing the end-points of the dashed lines (models having access only to labeled data from the WSJ) and the starting points of the solid lines (models that have access to both WSJ and QuestionBank), one can see that roughly the same improvements (from 72% to 86% UAS and from 60% to 77% LAS) can be obtained by having access to 2,000 labeled sentences from the target domain or uptraining with a large amount of unlabeled data from the target domain. The benefits seem to be complementary and can be combined to give the best results. The final accuracy of 88.63 / 84.14 (UAS / LAS) on the question evaluation set is comparable to the in-domain performance on newswire data (88.24 / 84.69).

4.3 Varying the base parser

Table 3 then compares uptraining on the output of different base parsers to pure self-training. In these experiments, the same set of 500,000 questions was parsed by different base parsers. The automatic parses were then added to the labeled training data and the parser was retrained. As the results show, self-training provides only modest improvements of less than 2%, while uptraining gives double-digit improvements in some cases. Interestingly, there seems to be no substantial difference between uptraining on the output of a single latent variable parser (Petrov et al., 2006) and a product of latent variable grammars (Petrov, 2010). It appears that the roughly 1% accuracy difference between the two base parsers is not important for uptraining.

4.4 POS-less parsing

Our uptraining procedure improves parse quality on out-of-domain data to the level of in-domain accuracy. However, looking closer at Table 3, one can see that the POS accuracy is still relatively low (93.53%), potentially limiting the final accuracy.

To remove this limitation (and also the dependence on a separate POS tagger), we experimented with word cluster features. As shown in Koo et al. (2008), word cluster features can be used in conjunction with POS tags to improve parsing accuracy. Here, we use them instead of POS tags in order to further reduce the domain-dependence of our model. Similar to Koo et al. (2008), we use the Brown clustering algorithm (Brown et al., 1992) to produce a deterministic hierarchical clustering of our input vocabulary. We then extract features based on vary-

	UAS	LAS	POS
Part-of-Speech Tags	88.35	84.05	93.53
Word Cluster Features	87.92	83.73	—

Table 4: Parsing accuracies of uptrained parsers with and without part-of-speech tags and word cluster features.

ing cluster granularities (6 and 10 bits in our experiments). Table 4 shows that roughly the same level of accuracy can be achieved with cluster based features instead of POS tag features. This change makes our parser completely deterministic and enables us to process sentences in a single left-to-right pass.

5 Error Analysis

To provide a better understanding of the challenges involved in parsing questions, we analyzed the errors made by our WSJ-trained shift-reduce parser and also compared them to the errors that are left after uptraining.

5.1 POS errors

Many parsing errors can be traced back to POS tagging errors, which are much more frequent on out-of-domain data than on in-domain data (88.8% on the question data compared to above 95.0% on WSJ data). Part of the reason for the lower POS tagging accuracy is the higher unknown word ratio (7.3% on the question evaluation set, compared to 3.4% on the WSJ evaluation set). Another reason is a change in the lexical distribution.

For example, wh-determiners (WDT) are quite rare in the WSJ training data (relative frequency 0.45%), but five times more common in the QuestionBank training data (2.49%). In addition to this frequency difference, 52.43% of the WDTs in the WSJ are the word “which” and 46.97% are“that”. In the QuestionBank on the other hand, “what” is by far the most common WDT word (81.40%), while “which” and “that” account only for 13.65% and 4.94% respectively. Not surprisingly the most common POS error involves wh-determiners (typically the word “what”) being incorrectly labeled as Wh-pronouns (WP), resulting in head and label errors like the one shown in Figure 3(a).

To separate out POS tagging errors from parsing errors, we also ran experiments with correct (gold) POS tags. The parsing accuracies of our shift-reduce parser using gold POS tags are listed in the last rows of Tables 1 and 2. Even with gold POS tags, the deterministic shift-reduce parser falls short of the accuracies of the constituency parsers (with automatic tags), presumably because the shift-reduce model is making only local decisions and is lacking the global constraints provided by the context-free grammar.

Dep. Label	Frequency	WSJ	Uptrained
nsubj	934	41.02	88.64
amod	556	78.21	86.00
dobj	555	70.10	83.12
attr	471	8.64	93.49
aux	467	77.31	82.56

Table 5: F_1 scores for the most frequent labels in the QuestionBank development set. Uptraining leads to huge improvements compared to training only on the WSJ.

5.2 Dependency errors

To find the main error types, we looked at the most frequent labels in the QuestionBank development set, and analyzed the ones that benefited the most from uptraining. Table 5 has the frequency and F-scores of the dependency types that we are going to discuss in the following. We also provide examples which are illustrated in Figure 3.

nsubj: The WSJ-trained model is often producing parses that are missing a subject (nsubj). Questions like “What is the oldest profession?” and “When was Ozzy Osbourne born?” should have “profession” and “Osbourne” as nsubjs, but in both cases the WSJ-trained parser did not label any subj (see Figures 3(b) and 3(c)). Another common error is to mislabel nsubj. For example, the nsubj of “What are liver enzymes?” should be enzymes, but the WSJ-trained parser labels “What” as the nsubj, which makes sense in a statement but not in a question.

amod: The model is overpredicting “amod”, resulting in low precision figures for this label. An example is “How many points make up a perfect fivepin bowling score?”. The Stanford dependency uses “How” as the head of “many” in noun phrases like “How many points”, and the relation is a generic “dep”. But in the WSJ model prediction, “many’s” head is “points,” and the relation mislabeled as amod. Since it’s an adjective preceding the noun,

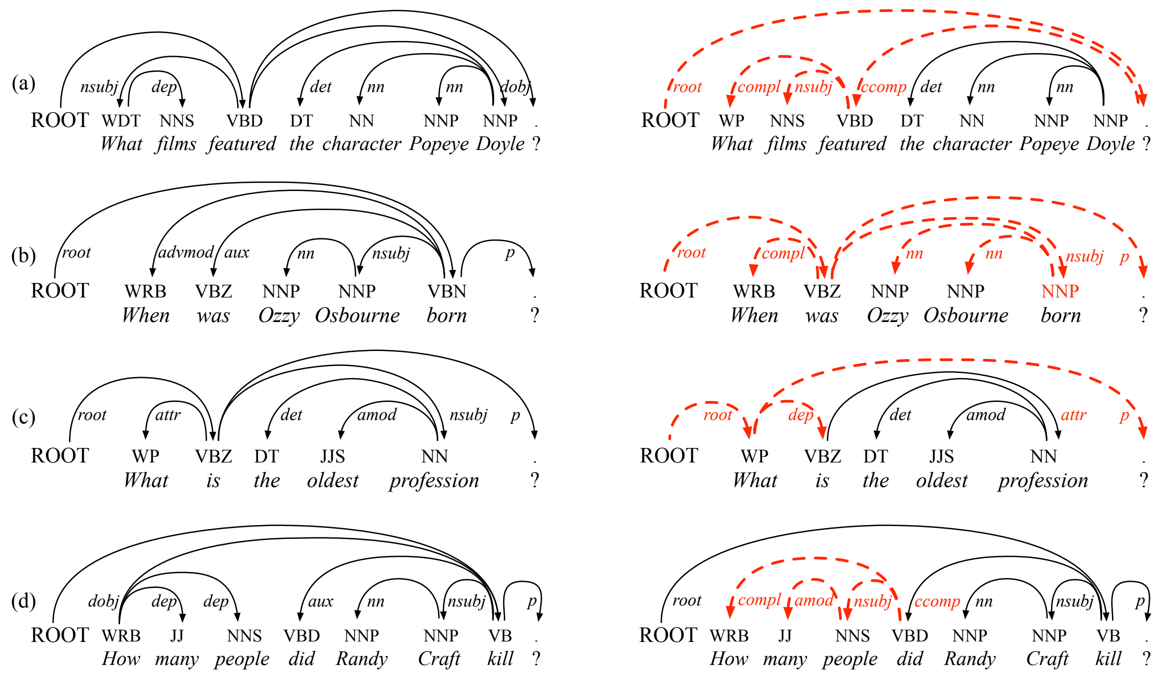


Figure 3: Example questions from the QuestionBank development set and their correct parses (left), as well as the predictions of a model trained on the WSJ (right).

the WSJ model often makes this mistake and therefore the precision is much lower when it doesn't see more questions in the training data.

dobj: The WSJ model doesn't predict object extraction well. For example, in "How many people did Randy Craft kill?" (Figure 3(d)), the direct object of kill should be "How many people." In the Stanford dependencies, the correct labels for this noun phrase are "dobj dep dep," but the WSJ model predicts "compl amod nsubj." This is a common error caused by the different word order in questions. The uptrained model is much better at handling these type of constructions.

attr: An attr (attributive) is a wh-noun phrase (WHNP) complement of a copular verb. In the WSJ training data, only 4,641 out of 950,028 dependencies are attr (0.5%); in the QuestionBank training data, 1,023 out of 17,069 (6.0%) are attr. As a consequence, the WSJ model cannot predict this label in questions very well.

aux: "What does the abbreviation AIDS stand for?" should have "stand" as the main head of the sentence, and "does" as its aux. However, the WSJ model labeled "does" as the main head. Similar patterns occur in many questions, and therefore the WSJ has a very low recall rate.

In contrast, mostly local labels (that are not related to question/statement structure differences) have a consistently high accuracy. For example: det has an accuracy of 98.86% with the WSJ-trained model, and 99.24% with the uptrained model.

6 Conclusions

We presented a method for domain adaptation of deterministic shift-reduce parsers. We evaluated multiple state-of-the-art parsers on a question corpus and showed that parsing accuracies degrade substantially on this out-of-domain task. Most notably, deterministic shift-reduce parsers have difficulty dealing with the modified word order and lose more than 20% in accuracy. We then proposed a simple, yet very effective *uptraining* method for domain-adaptation. In a nutshell, we trained a deterministic shift-reduce parser on the output of a more accurate, but slower parser. Uptraining with large amounts of unlabeled data gives similar improvements as having access to 2,000 labeled sentences from the target domain. With 2,000 labeled questions and a large amount of unlabeled questions, uptraining is able to close the gap between in-domain and out-of-domain accuracy.

Acknowledgements

We would like to thank Ryan McDonald for running the MstParser experiments and for many fruitful discussions on this topic. We would also like to thank Joakim Nivre for help with the MatlParser and Marie-Catherine de Marneffe for help with the Stanford Dependency Converter.

References

J. Blitzer, R. McDonald, and F. Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP '06*.

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT '98*.

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

P. Brown, V. Della Pietra, P. deSouza, J. Lai, and R. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*.

X. Carreras, M. Collins, and T. Koo. 2008. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *CoNLL '08*.

E. Charniak and M. Johnson. 2005. Coarse-to-Fine N-Best Parsing and MaxEnt Discriminative Reranking. In *ACL'05*.

E. Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *AI '97*.

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

S. Clark, J. Curran, and M. Osborne. 2003. Bootstrapping pos-taggers using unlabelled data. In *CoNLL '03*.

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

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING '96*.

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

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

Z. Huang and M. Harper. 2009. Self-training PCFG grammars with latent annotations across languages. In *EMNLP '09*.

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

T. Koo and M. Collins. 2010. Efficient third-order dependency parsers. In *ACL '10*.

T. Koo, X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *ACL '08*.

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

A.F.T. Martins, N.A. Smith, and E.P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *ACL '09*.

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

D. McClosky, E. Charniak, and M. Johnson. 2006b. Reranking and self-training for parser adaptation. In *ACL '06*.

R. McDonald and J. Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *EMNLP '07*.

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

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *CoNLL '06*.

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kbler, S. Marinov, and E. Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2).

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

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

S. Petrov. 2010. Products of random latent variable grammars. In *NAACL '10*.

S. Riedel and J. Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *EMNLP '06*.

K. Sagae and A. Lavie. 2006. Parser combination by reparsing. In *NAACL '06*.

K. Sagae and J. Tsujii. 2007. Dependency parsing and domain adaptation with lr models and parser ensembles. In *CoNLL '07*.

M. Steedman, M. Osborne, A. Sarkar, S. Clark, R. Hwa, J. Hockenmaier, P. Ruhlen, S. Baker, and J. Crim. 2003. Bootstrapping statistical parsers from small datasets. In *EACL '03*.

J. Suzuki, H. Isozaki, X. Carreras, and M. Collins. 2009. An empirical study of semi-supervised structured conditional models for dependency parsing. In *EMNLP '09*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *IWPT '03*.