Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence

Learning Domain Differences Automatically for Dependency Parsing Adaptation

Mo Yu, Tiejun Zhao, and Yalong Bai
Harbin Institute of Technology
China
{yumo, tjzhao, ylbai}@mtlab.hit.edu.cn

Abstract

In this paper, we address the relation between domain differences and domain adaptation for dependency parsing. Our quantitative analyses showed that it is the inconsistent behavior of same features cross-domain, rather than word or feature coverage, that is the major cause of performances decrease of out-domain model. We further studied those ambiguous features in depth and found that the set of ambiguous features is small and has concentric distributions. Based on the analyses, we proposed a DA method. The DA method can automatically learn which features are ambiguous cross domain according to errors made by out-domain model on in-domain training data. Our method is also extended to utilize multiple out-domain models. The results of dependency parser adaptation from WSJ to Genia and Question bank showed that our method achieved significant improvements on small in-domain datasets where DA is mostly in need. Additionally, we achieved improvement on the published best results of CoNLL07 shared task on domain adaptation, which confirms the significance of our analyses and our method.

1 Introduction

Statistical models are widely used in the field of dependency parsing. However, current models are usually trained and tested on data from the same domain. When test data belongs to a domain different from the training data, the performances of the current dependency parsing models will be greatly degraded. Therefore when the labeled Treebank of the target domain is insufficient, it is difficult to obtain accurate parsing results on this domain.

To quickly adapt parser to new domains where few in-domain labeled data is available, various techniques have been proposed. No labeled data from target domain is needed for most parser domain adaptation (DA) methods, e.g. self-training [McClosky *et al.*, 2008; Sagae, 2010], co-training [Steedman *et al.*, 2003; Sagae and Tsujii, 2007] and word clustering approach [Candito *et al.*, 2011]. These unsupervised methods improve performances by helping parsers cover more domain specific words or features [McClosky and Charniak, 2008].

However, as will be shown in this paper, word and feature coverges is not the only factor affecting cross-domain performance. Specifically, we take WSJ corpus and Genia corpus [Tateisi *et al.*, 2005] as examples. During analysis, even though we added gold POS tags and made the gap of word coverage no longer exist, performance decline is still not alleviated much. It is now the ambiguous feature, behaving inconstantly in different domains, that brings the performance drop. In addition, Dredze *et al.* [2007] pointed out that domain differences may exist in different annotation guidelines between Treebanks.

Above findings indicated that some labeled data was in need for handling such differences. Unlike unsupervised methods with difficulty to detect and handle these differences, current supervised and semi-supervised parser adaptation [Hall *et al.*, 2011] are proved to get better results. However, they do not directly solve the domain differences on features discussed above.

In this paper, we try to learn which features are ambiguous between domains with the help of only a small in-domain labeled dataset. The key idea is to learn which features are more likely to be associated with errors based on in-domain training data. Then the model could identify and correct the unreliable arcs based on the ambiguous features they contained, while still keeping as many reliable arcs outputted by out-domain model as possible.

There are two major contributions in this paper. First, in Section 2, quantitative analyses are performed to find out which types of domain differences affect the cross-domain performances of a parser mostly. As far as we know, few works [Gildea, 2001] had focused on this problem. Second, based on some general rules found in the analyses, in Section 3, we proposed a method to automatically learn domain differences from small in-domain dataset, meanwhile avoiding overfitting. Results of experiments are shown in Section 4. Section 5 gives the conclusion.

2 Analysis on Domain Differences

2.1 Experimental Settings

In this section, we set Genia and WSJ corpus as in-domian and out-domain data respectively. For WSJ corpus, sections

2-21 are used for training (39832 sentences) and section 23 for test. For Genia corpus, training (14326 sentences) and test data are divided as McClosky and Charniak [2008] did. Treebanks are converted to dependency format using Penn Converter [Johansson and Nugues, 2007]. First-order MST-Parser [McDonald *et al.*, 2005] is used to generate projective trees. Stanford POS-tagger [Toutanova *et al.*, 2003] is chosen when a new trained POS-tagger is in need. The performances are evaluated by unlabeled attachment accuracy (UAS).

2.2 Ceiling Analysis

We believe the sources of performance drop, when using dependency parsers cross-domain, can be divided into two parts: first, the accuracy of POS tags, which has been proved to be important and greatly affected by OOV and ambiguous words; second, some "real" grammatical differences, which exist even when the POS tags are completely accurate. So first of all, a ceiling analysis was done to determine how seriously the errors of POS-tagging stage would impact the performances.

Tagger	Method	UAS(%)	POS(%)
Gold tag	In-domain	89.58	100
	Out-domain	85.16	100
Auto tag	In-domain	88.80	98.35
	Out-domain	81.80	86.03

Table 1: Basic Results

Genia test set was used for evaluation and parser trained on it was called "in-domain". For "Out-domain" method, WSJ corpus is used for training. Different taggers are used to label test data. Here "Gold tag" means Gold POS tags from Treebanks are used for parsing. And "Auto tag" refers to the POS tagger trained on the same domain that was used for training parser. As shown in Table 1, when labeling POS tags automatically, out-domain parser performed much worse (great drop of 7%) than in-domain parser.

However, the accuracy of POS tagging, greatly affected by OOVs cross-domain, is not the only source of performance drop. Even when gold tags were used and therefore errors from POS tagging were eliminated, a decrease of 4.42% still existed. This implies the existence of other factors more important for domain differences. They are grammar level ambiguities, so we call them grammatical differences. Note that in the field of dependency parsing, we concentrate on syntactic structure rather than predicting POS tags. Hence, in the rest of the paper, gold POS tags are used to avoid influence of POS tag accuracies and to reveal grammatical differences.

2.3 Word Coverage

In this section, we show that domain differences of dependency parsing do not lie on the higher word coverage of in-domain model on Genia test set (1970 OOVs out of 35639 tokens) than that of out-domain model (8253 OOVs). To prove this, when decoding with in-domain model, words considered to be OOVs by out-domain model are treated as unknown words for in-domain model, making both models have the same word coverage on test data. Result shows only a slight reduction (0.11%) on in-domain model. Interestingly, Tateisi *et al.* [2005] also states that annotation of Genia Treebank can be stably done by linguists without knowledge of biology. This may suggest some similarities between parsing and human annotating, since parsers are also not affected much by unknown biological terms.

2.4 Feature Coverage

Experiments showed out-domain model had low feature coverage on in-domain data. In the 2918721 features extracted from Genia, only 860911 were covered by WSJ model. To verify whether this is related to the performances drop, we built both models on the intersection features of in-domain and out-domain models. Results showed that after eliminating the difference of feature coverages, out-domain model (84.63%) still performs much worse than in-domain model (89.15%).

Furthermore, since features with different signs are usually considered as major source of ambiguity, we also built a general feature set to verify their influence. Features with different signs were deleted from the intersection and those with same signs in both in-domain and out-domain models left. Two models were re-trained on this general feature set. Performance of the new in-domain model had a small drop (0.58%) to 89.00%, while that of out-domain model rose 1.31 percent to 85.94% by only deleting features with different signs.

We achieved three conclusions from this analysis: (1) domain-specific features do not contribute much to the high performance of in-domain models (since the performance drop caused by those features is trivial compared to still notable drop between performances of new in-domain and out-domain models); (2) removing ambiguous features with different sign can improve performance of out-domain model; (3) for different domains, different behaviors (i.e. weights) of the same feature may play a more important role than feature coverage, which are also the source of ambiguity.

2.5 Analyses of different behaviors of features between domains

Above experiments showed that word and feature coverages did not account for grammatical differences cross-domain. Instead, ambiguous features, weighted differently in different domains, are the crucial factor. In this section, a method is proposed to evaluate ambiguities of features between different domains. Further more, we will show that ambiguous features, which are the manifestations of domain differences, concentrate on a small subset with patterns and are related to underestimated gold arcs in target domain. The analyses directly lead to our design of DA algorithm in Section 3.

For intersection features in both WSJ and Genia corpus, significant tests were performed to see whether they behave differently cross-domain. Features behave significantly differently are those ambiguous features we want. For each feature, the conditional probabilities are defined:

$$p(y=1|f) = \frac{c(y=1,f)}{c(f)} + \lambda \cdot \frac{c(y=1)}{N} \qquad (1)$$

$c(y, f)$ is the count of co-occurrences of feature f and label y, where $y = 1$ means f appears on an arc and $y = 0$

otherwise. The proportion of arcs in all pair of words is interpolated for smoothing, where the parameter λ was set to 0.0001 in the experiments.

Z-tests were performed to test the differences of conditional probabilities between same features from two models. Since $p(y|f)$ is a Bernoulli distribution, its mean value μ_i is $p(y = 1|f)$, and the variance can be computed as $p(y = 1|f)(1 - p(y = 1|f))$. Results showed that about 16.35% of the intersection features behaved significantly differently with $p < 0.01$. We also randomly splited WSJ into two non-intersect parts (WSJ & WSJ) for comparison. Compared with WSJ & WSJ, ambiguous features cross-domain are much more for each level of significance test.

	Significance ($p < \alpha$)		
	$\alpha = 0.01$	0.005	0.001
Genia & WSJ	140798	120692	95838
WSJ & WSJ	37122	26843	19912

Table 2: Z-test for feature behavior difference cross-domain and in-domain

We also illustrate the concentration of ambiguous features by comparing scores of gold arcs from test data given by out-domain model. Gold arcs with low scores, which are mainly caused by ambiguous features, may not be selected and hence lead to mistakes. For each arc between a word w and its head h, we first define normalized score (probability) as $p(h|w) = e^{s(w,h)} / \sum_{h' \epsilon W \setminus w} e^{s(w,h')}$, where $s(w, h)$ is the score of arc and W is the set of all words in the sentence.

Figure 1 illustrates the numbers of gold arcs with different probabilities assigned by the out-domain and in-domain models. Most errors of out-domain model are concentrated (e.g. with probabilities less than 0.05). This implies that features behaving differently cross-domain appear mostly on these abnormally underestimated arcs. Meanwhile, these arcs also contains more ambiguous features. Thus, if ambiguous features on those arcs can be detected and re-weighted, those mistakes can be corrected.

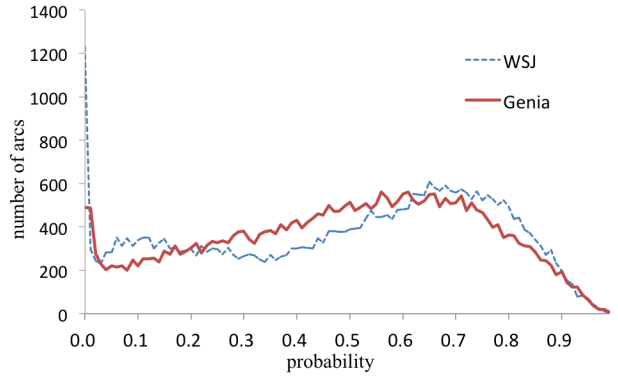

Figure 1: Probabilities of gold arcs assigned by different models

Above errors of arcs fall into some common types. Most errors are related to noun phrases and coordination structures

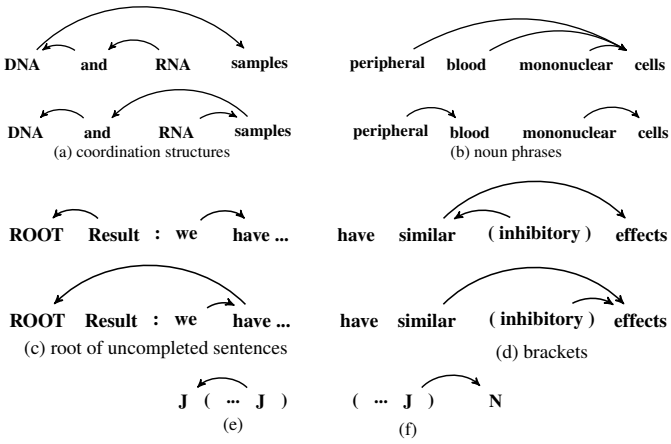

Figure 2: Unreliable arcs outputted by WSJ model

(Figure 2(a) and 2(b)), which mainly originate from the annotation standards of Genia corpus [Tateisi *et al.*, 2005]. Genre differences also cause many errors, e.g. errors in finding roots of incomplete Genia sentences as in Figure 2(c), where the WSJ model tends to label verbs as roots rather than nouns. Another example is the type of errors caused by frequently appearing brackets in Genia corpus as in Figure 2(d), where the word 'inhibitory' modifies the word 'similar' as an explanation, instead of directly modifying 'effects'. Figure 2(e) and 2(f) list the corresponding ambiguous features. WSJ model tends to give higher weight to feature 2(f) and lower weight to feature 2(e), while Genia model does in the opposite way.

Additionally, we consider two reasons for error occurrences: gold arc given a low score as well as another arc mistakenly scored higher. For each error, we count the numbers of ambiguous features appearing in both incorrectly labeled arc (overestimated) and gold one (underestimated). Interestingly, results showed that the numbers of ambiguous features are close for both over and underestimated arcs. For example when $\alpha = 0.001$, the numbers are 42.99 and 41.09 accordingly. This observation suggests that the gold arcs ignored by the out-domain model also contain important information about ambiguous features. So during re-weighting of ambiguous features, the algorithm should be able to learn from both types of arcs.

3 Proposed Method

3.1 Learning Domain Differences for Parser Adaptation

A common DA method is to use predictions of out-domain model as a new feature. In dependency parsing, this corresponds to adding a new feature, which indicates whether the arc is selected by the out-domain model, when scoring an arc. The underlying assumption is that the outputs of out-domain model are correct most of the time. However, as shown in Section 2.5, some ambiguous features behave differently cross-domain. Therefore, if an arc predicted by out-domain model contains some ambiguous features, it should

be less reliable and should not be fully trusted.

Trying to utilize such information, we not only added a unique feature indicating the outputs of out-domain, but also added conjunctions between the outputs and original features as new features so as to distinguish the ambiguities of original features. In this way the outputs of out-domain model are treated differently. If they contain ambiguous features, their confidences will decrease. Since we do not know exactly which features are ambiguous, our method learns this from the errors made by out-domain model on in-domain training data. Intuitively, when an error is made by out-domain model, the associated features are most likely to be ambiguous. Based on the discussion in the end of Section 2.5, features related to underestimated gold arcs are also more likely to be ambiguous. To detect ambiguous features, we consider both types of arcs and add the second and third terms in Eq.(2).

$$score(T|S,T_O)=\sum_{e\in T}\Bigg(\sum_{f\in F}\lambda_f\cdot f(e)+\delta(e\in T_O)\sum_{f\in F\&count(f)>1}\varphi_f\cdot f(e) +\delta(e\notin T_O)\sum_{f\in F\&count(f)>1}\theta_f\cdot f(e)+\mu\cdot\delta(e\in T_O)\Bigg) \quad (2)$$

Here T_O is the dependency tree predicted by out-domain model, e stands for a possible arc and $\delta(x)$ is indicator function. $f(e)$ means feature f is observed on e. The first item in the summation is the same score function with the feature set F as in [McDonald *et al.*, 2005]. The second (third) term corresponds to additional score of e when the out-domain model infers e does (not) exist. Only "frequent features", which are from F and appear more than once on gold trees, are calculated. λ, φ, θ and μ are weights of features. Intuitively, take the second term as an example. When φ_f is low, it reflects the existence of edge e predicted by out-domain model is not trustworthy, since e contains ambiguous feature f. The last item stands for the overall confidence when the arc is outputted by out-domain model.

We still take Figure 2 as an example to show how the method works. During training by MIRA, since gold arc 2(d) from "inhibitory" to "similar" is incorrectly not selected by out-domain model, features associated with gold arc, like 2(e), will get their weights in the third term be increased, to encourage selecting un-chosen arcs by out-domain model with those features. Meanwhile, features comprising the mistakenly chosen arc in T_O by out-domain output, for instance 2(d), will be punished, by decreasing their weights in the second term. All the mechanisms together will make the model score the overestimated arc in 2(d) lower and the underestimated gold arc in 2(d) higher. Through the training, our method could automatically detect which types of arcs in T_O are unreliable by re-weighting features associated with them.

Our method is similar to that in [McDonald and Nivre, 2011], in which whether an arc was outputted by another model is used as a feature for parser combination. However their method treats all features equally, while our method distinguish them by the second and third terms in Eq.(2). Our method is also different from most DA methods as discussed in the last paragraph of Section 2.5. We add features in the third term to capture ambiguous feature on underestimated gold arcs, while existing methods just extract features from output arcs but neglect the underestimated gold arcs that should be chosen.

There are two reasons for our method to work well on small in-domain data. First, only frequent features are used in the second and third terms, because domain differences are usually related to more common types of ambiguous features as discussed in Section 2.5. So our method can be seen as adjusting weights of a constrained feature set, which could be done on just a small dataset. Second, since out-domain model has an acceptable accuracy on in-domain data and the adapted parser is consistent with out-domain parser most of the time, actually only a small set of ambiguous features should have their weights adjusted. So the method can benefit from large out-domain data while take advantage of fewer in-domain data.

3.2 Domain Adaptation from multi-domain models

Our proposed DA method can be easily extended to use models from multi-domains to enhance the in-domain parser. Formula (2) is extended as following:

$$score(T|S,\{T_{O_i}\})=\sum_{e\in T}\Bigg(\sum_{f\in F}\lambda_f\cdot f(e) +\sum_i\delta(e\in T_{O_i})\sum_{f\in F\&count(f)>1}\varphi_{if}\cdot f(e) +\sum_i\delta(e\notin T_{O_i})\sum_{f\in F\&count(f)>1}\theta_{if}\cdot f(e) +\sum_i\mu_i\cdot\delta(e\in T_{O_i})\Bigg) \quad (3)$$

Here T_{O_i} is the dependency tree predicted by the model from the *i*th out-domain. Features associated with different domains have different weights φ_i, θ_i and μ_i.

4 Experiments

4.1 Overall Results on Adaptation from WSJ to Genia

To compare with our proposed method, we ran three baseline methods: training the parser on out-domain data only (Out-domain), on in-domain data only (In-domain) and on the combination of both in-domain and out-domain data (Combination). The results of the method proposed in [McDonald and Nivre, 2011] (McDonald) was also showed in the figure. Although their method was proposed for combination of parsers, we made it a DA method which combines the outputs of out-domain parser with in-domain parser. Same settings with Section 2.1 are used. Performances of these methods are plotted in Figure 3.

The proposed method outperforms the others in most cases. It significantly improved the performances against both in-domain ($> 1.5\%$) and combination ($> 1\%$) method on

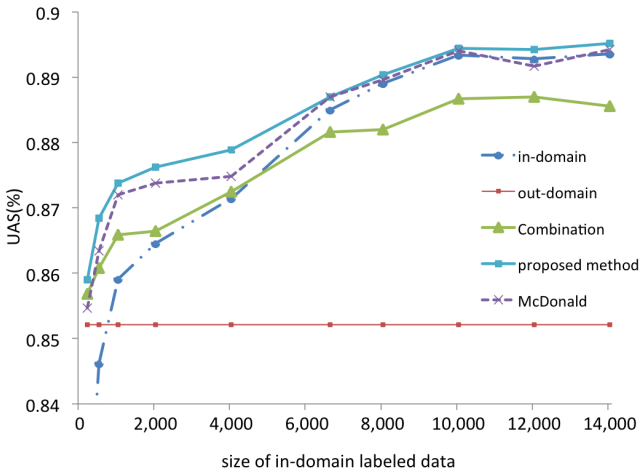

Figure 3: Performances of different methods

small in-domain dataset with size between 500 and 2000. Our method also outperforms the method of [McDonald and Nivre, 2011]. When the number of in-domain sentences is less than 500, our method utilizes more information from out-domain parser. With the increase of in-domain data, the introduced features could be better trained under our method. Until the system benefits more from sufficient in-domain data than out-domain parser where DA is no more necessary, the in-domain model starts to get similar scores to those of DA methods. Besides, analysis in Section 2 together with above results show the necessity of certain amount of in-domain labeled data, since the main domain differences lay on the inconsistent conditional probabilities of same features. This also agrees with the conclusion of [Dredze *et al.*, 2007].

Our method well suits the task of DA, where few in-domain data is available. For example our method achieved similar performance to the best result of in-domain model but with only 1/7 of in-domain data when 2000 in-domain sentences were used. One of the reason is the existence of some general domain differences which can be utilized even when the in-domain data set is small, as discussed in Section 2.5. Among all the errors made by WSJ model, 1932 can be corrected by in-domain model trained on only 2000 sentences of Genia corpus, leading to a potential performance improvement of 5.42% on UAS of out-domain model. While Performance further increases 6.67% when trained on all 14326 sentences. This shows the small in-domain data has comparable information to the much larger data for correcting the out-domain model. And the proposed method can untilize such information very well.

4.2 Performance on Question bank

To show the generalization strength of the method, we also conducted experiments on Question Bank [Judge *et al.*, 2006]. As in [Petrov *et al.*, 2010], the first 2000 sentences were for training and the last 1000 for test. Our method got 1.7% improvement on UAS when gold POS tags were used.

Meanwhile, our methods can be easily modified for domain adaptation of POS-taggers. So we also applied our method on DA of POS-tagging by replacing score function in Eq.(2) with that of POS-tagging. In this way, we can deal with both the two types of domain differences as described in Section 2.2. We carried out experiments where POS-tags were automatically labeled and adapted to target domain. In Table 3, adapted tagger greatly improved both POS tag and UAS performances against the taggers trained on either in-domain or out-domain data. When the tagger and parser were both adapted, there was an improvement of 3.12% against the best performance when no adaptation was taken.

Methods		UAS(%)	POS(%)
Auto POS-tag	In-domain	88.04	90.05
	Out-domain	83.26	87.12
Adapted POS-tag	In-domain	89.74	92.94
	Out-domain	87.97	92.94
	Parser Adapted	**91.16**	92.94
Gold POS-tag	In-domain	92.39	100
	Out-domain	91.43	100
	Parser Adapted	**94.09**	100

Table 3: Performances on Question Bank

4.3 Results on DA from multi-domain models

As proposed in Section 3.2, our DA method can be extended to further benefit from multi-domain models. To show the potential of performance improvement by adding more available out-domain models, in addition to WSJ model, we also add Genia model into the DA system in the way described by Eq.(3).

The results of adaptation to Question Bank with the help of both WSJ and Genia models are listed in Table 4. Note that Genia and Question Bank do not share much similarities, due to the lack of questions in Genia. Even so, when automatic tags are used, using multi-domain models still yielded an improvement of about 0.5% against using WSJ only. This shows Genia corpus could help to reduce errors accumulated in the pipeline of dependency parsing. We believe that if more corpora, especially those from more related domains, are available, the framework described in Eq.(3) will achieve further improvements.

Methods		Baseline	Adapted
Auto Tag	WSJ model	83.26	91.16
	Genia model	73.97	86.84
	Genia+WSJ model	-	**91.61**
Gold Tag	WSJ model	91.43	94.09
	Genia model	86.07	92.96
	Genia+WSJ model	-	**94.12**

Table 4: Results of Adapting from Multi-domains

4.4 Results on CoNLL-07 Domain Adaptation Track

In CoNLL-07 shared task, there was a track on DA of dependency parser [Nivre *et al.*, 2007]. The goal is to adapt

WSJ parsers to a test set of chemical abstracts denoted as PchemTB. According to [Dredze *et al.*, 2007], no system had succeeded in this task since their system without any adaptation scored in the 2nd place and the system ranked in the 1st place was only 0.04% higher than theirs. Especially, their attempts for domain adaptation did not improve the system with no adaptation either.

The shared task also provided 200 labeled sentences from biomedical abstracts as a development set, which is supposedly more similar to PchemTB. Since both the development set and the test set are from the PennBioIE[Kulick *et al.*, 2004] project, we believe that the grammatical gap between the two data sets is smaller than that between the training set and the test set. So we employed our adaptation algorithm to this task to see if there is any space for improvement when the development set of 200 sentences are used as in-domain data to bridge the gap between source and target domains.

We tried to reproduce the results of [Dredze *et al.*, 2007] with no adaptation but only found their published MSTParser with a slightly better version. So the reproduced baseline is higher (0.06%) than theirs. Although base on this better parser, our adaptation can still achieved 0.38% improvement on the reproduced baseline. The best performance of the newest published MSTParser is also listed in Table 5. Our method could improve it by 0.3%. The develop set with only 200 sentences is too small to capture enough in-domain knowledge and therefore limit the maximum improvement one can ever achieve. This is one possible reason why the best MSTparser improved less than reproduced one.

For comparison between existing DA approaches, we also re-implemented the method in [McDonald and Nivre, 2011] and the EasyDA method proposed in [Daumé, 2007] on the same task. Our method out-performed all participants' unsupervised DA and other supervised DA results, with the same resources as the "closed-track" provided. This results proved that our method indeed could well utilize small in-domain data. From the results, the method in [McDonald and Nivre, 2011] only slightly improve the performance, which is consistent with the conclusion in Section 4.1 that our method has advantages to theirs when the in-domain labeld set is small.

Method	**Baseline (UAS)**	**Adaptation (UAS)**
Proposed method (based on best mstparser)	83.66	**83.96**
EasyDA (based on best mstparser)	83.66	83.34
McDonald (based on best mstparser)	83.66	83.68
Proposed method (based on reproduced mstparser)	83.44	83.82
Dredze et al. (2nd in CoNLL07)	83.38	83.38
Sagae & Tsujii (1st in CoNLL07)	-	83.42

Table 5: Performances of Proposed Method on CoNLL07 Domain Adaptation Track

5 Conclusion and perspectives

In this paper, a DA method is proposed to better benefit from out-domain model's predictions by learning what kinds of features are ambiguous. The method improved the performances significantly.

The method can be improved from different perspectives in the future. Firstly, sparse constrains can be introduced to the algorithm and extra unlabeled in-domain data can be made use of, in order to avoid overfitting and get better generation strength. At the same time, since the results showed that adaptation from multi-domains may achieve better results than from a single out-domain, we plan to better explore information from multi-domains based on their similarities to the target domain.

Acknowledgments

We would like to thank Dr. Hailong Cao, Shujie Liu and Lemao Liu for many discussions and thank the anonymous reviewers for their valuable comments and helpful suggestions. This work was supported by National Natural Science Foundation of China (61173073), and the Key Project of the National High Technology Research and Development Program of China (2011AA01A207).

References

[Candito *et al.*, 2011] M. Candito, E. Henestroza Anguiano, D. Seddah, et al. A word clustering approach to domain adaptation: Effective parsing of biomedical texts. In *Proceedings of ICPT 2011*, 2011.

[Daumé, 2007] H. Daumé. Frustratingly easy domain adaptation. In *Annual meeting-association for computational linguistics*, volume 45, page 256, 2007.

[Dredze *et al.*, 2007] M. Dredze, J. Blitzer, P.P. Talukdar, K. Ganchev, J. Graca, and F. Pereira. Frustratingly hard domain adaptation for dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 1051–1055, 2007.

[Gildea, 2001] D. Gildea. Corpus variation and parser performance. In *Proceedings of EMNLP 2001*, volume 1, pages 167–202, 2001.

[Hall *et al.*, 2011] K. Hall, R. McDonald, and S. Petrov. Training structured prediction models with extrinsic loss functions. In *Proceedings of Domain Adaptation Workshop at NIPS 2011*, 2011.

[Johansson and Nugues, 2007] R. Johansson and P. Nugues. Extended constituent-to-dependency conversion for english. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*, 2007.

[Judge *et al.*, 2006] J. Judge, A. Cahill, and J. Van Genabith. Questionbank: creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 497–504. Association for Computational Linguistics, 2006.

[Kulick *et al.*, 2004] Seth Kulick, Ann Bies, Mark Liberman, Mark Mandel, Ryan McDonald, Martha Palmer, Andrew Schein, Lyle Ungar, Scott Winters, and Pete White. Integrated annotation for biomedical information extraction. In *Proc. of the Human Language Technology Conference and the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*, pages 61–68, 2004.

[McClosky and Charniak, 2008] D. McClosky and E. Charniak. Self-training for biomedical parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 101–104. Association for Computational Linguistics, 2008.

[McDonald and Nivre, 2011] R. McDonald and J. Nivre. Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(1):197–230, 2011.

[McDonald *et al.*, 2005] R. McDonald, K. Crammer, and F. Pereira. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98. Association for Computational Linguistics, 2005.

[Nivre *et al.*, 2007] J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. The conll 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 915–932, 2007.

[Petrov *et al.*, 2010] S. Petrov, P.C. Chang, M. Ringgaard, and H. Alshawi. Uptraining for accurate deterministic question parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 705–713. Association for Computational Linguistics, 2010.

[Tateisi *et al.*, 2005] Y. Tateisi, A. Yakushiji, T. Ohta, and J. Tsujii. Syntax annotation for the genia corpus. In *Proceedings of IJCNLP*, volume 5, pages 222–227, 2005.

[Toutanova *et al.*, 2003] K. Toutanova, D. Klein, C.D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.