Combining Active Learning and Partial Annotation for Domain Adaptation of a Japanese Dependency Parser

Daniel Flannery*
Vitei Inc.
Kankoboko-cho 79 Shimogyo-ku,
Kyoto, Japan
danielflannery@gmail.com

Shinsuke Mori
ACCMS, Kyoto University
Yoshida Honmachi, Sakyo-ku,
Kyoto, Japan
forest@i.kyoto-u.ac.jp

Abstract

The machine learning-based approaches that dominate natural language processing research require massive amounts of labeled training data. Active learning has the potential to substantially reduce the human effort needed to prepare this data by allowing annotators to focus on only the most informative training examples. This paper shows that active learning can be used for domain adaptation of dependency parsers, not just in single-domain settings. We also show that entropy-based query selection strategies can be combined with partial annotation to annotate informative examples in the new domain without annotating full sentences. Simulations are common in work on active learning, but we measured the actual time needed for manual annotation of data to better frame the results obtained in our simulations. We evaluate query strategies based on both full and partial annotation in several domains, and find that they reduce the amount of in-domain training data needed for domain adaptation by up to 75% compared to random selection. We found that partial annotation delivers better in-domain performance for the same amount of human effort than full annotation.

1 Introduction

Active learning is a promising approach for domain adaptation because it offers a way to reduce the amount of data needed to train classifiers, minimizing the amount of difficult in-domain annotation. This type of annotation requires annotators to have both domain knowledge plus familiarity with annotation standards. There has been much recent work on active learning for a variety of natural language processing tasks (Olsson, 2009), but most of it is concerned only with the single-domain case. Additionally, work on active learning commonly reports results for simulations only because of the high cost of annotation work.

We use active learning to perform domain adaptation for a Japanese dependency parsing task, and measure the actual time required for manual annotation of training data to better frame the results of our experiments. This kind of evaluation is crucial for assessing the effectiveness of active learning in practice.

Previous work on active learning for structured prediction tasks like parsing (Hwa, 2004) often assumes that the training data must be fully annotated. But recent work on dependency parsing (Spreyer et al., 2010; Flannery et al., 2011) has shown that models trained from partially annotated data (where only part of the tree structure is annotated) can achieve competitive performance. However, deciding which portion of the tree structure to annotate remains a difficult problem.

2 Related Work

Most previous work on active learning for parsing (Hwa, 2004; Sassano and Kurohashi, 2010) studies the single-domain case, where the initial labeled data set and the pool of unlabeled data share the same domain. An important difference from previous work is that we focus on domain adaptation, so we assume that the initial labeled data and annotation pool come from different domains.

Previous work on active learning for parsing (Tang et al., 2002; Hwa, 2004) has focused on selecting sentences to be fully annotated. Sassano and Kurohashi (2010) showed that smaller units like phrases (*bunsetsu*) could also be used in an active learning scenario for a Japanese dependency parser. Their work included results for partially

*This work was done when the first author was at Kyoto University.

Proceedings of the 14th International Conference on Parsing Technologies, pages 11–19,
Bilbao, Spain; July 22–24, 2015.

	i	1	2	3	4	5	6	7	8	9
annotation	w_i	政府	は	投資	に	つなが	る	と	歓迎	し
	Eng.	Gov.	*subj.*	investment	to	leads	*ending*	that	welcomes	do
	t_i	noun	part.	noun	part.	verb	infl.	part.	noun	verb
	d_i full	2	8	4	5	6	7	8	9	NULL
	d_i part.		8							
features	F1		6							
	F2		は						歓迎	
	F3		part.						noun	
	F4		NULL, NULL, 政府						つなが, る, と	
	F5		投資, に, つなが						し, NULL, NULL	
	F6		NULL, NULL, noun						verb, infl., part.	
	F7		noun, part., verb						verb, NULL, NULL	

The second word, the case marker は (*subj.*), has two grammatically possible heads: the verbs つなが (leads) and 歓迎 (welcomes). In the partial annotation framework, only this word needs to be annotated.

Table 1: An example of full annotation (d_i full) and partial annotation (d_i part.) for a sentence. Features for the dependency between the case marker は (*subj.*) and the verb 歓迎 (welcomes) are also shown.

annotated sentences, but did not use entropy-based query strategies (Tang et al., 2002; Hwa, 2004) designed for selecting whole sentences because of the difficulty of applying them. We use an even smaller unit, words, and show how entropy-based measures can be successfully applied to their selection.

Mirroshandel and Nasr (2011) also investigated selection of units smaller than sentences for a graph-based parser in the single-domain setting. Their query strategy used an entropy-based measure calculated from n-best lists, which are computationally expensive and require modification of the parser's edge scoring function to produce. In contrast, our query strategy is a simpler one that does not require n-best lists.

All of the work discussed here reports results for simulations only. This is common practice in active learning research because large-scale annotation is prohibitively expensive. Some recent work on active learning has started to include more realistic measures of the actual costs of annotation (Settles et al., 2008). In this paper, we measure the time needed to manually annotate sentences with dependencies to better understand the costs of active learning for dependency parsing.

3 MST Parsing

Currently, the two major types of data-driven dependency parsers are shift-reduce parsers (Nivre et al., 2006) and graph-based parsers (McDonald et al., 2005). Shift-reduce parsers perform parsing deterministically (so their time complexity can be as fast as linear in the size of the input). Graph-based dependency parsers treat parsing as the search for a directed maximum spanning tree (MST). We adopt the latter type in this paper because its accuracy is slightly higher especially for long sentences (McDonald and Nivre, 2011).

3.1 Partial Annotation

Our goal is to reduce the total cost of preparing data for domain adaptation. We do this by combining partial annotation with active learning. *Partial annotation* refers to an annotation method where only some dependencies in a sentence are annotated with their heads. The standard method in which all words must be annotated with heads is called *full annotation*. Table 1 shows an example of both types of annotation for a sentence.

Full sentences are the default unit of annotation in full annotation, even though the parser is trained from and operates on smaller units such as words or chunks. The motivation for partial annotation is to match the unit of annotation with the smallest unit that the parser uses for training. In the extreme case this is as small as a single dependency between two words. This fine-grained annotation unit is a natural fit for active learning, where we seek to find training examples with the greatest training value. However, fine-grained units are cognitively more difficult for a human annotator because less context is available. Thus, we must balance the granularity of annotations against the difficulty of processing them.

3.2 Pointwise MST Parsing

To enable the use of partial annotation in active learning, we use a pointwise MST parser (Flannery et al., 2011) that predicts each word's head independently. It uses only simple features based

on surface forms and part-of-speech (POS) tags of words, and first-order features between pairs of head and dependent words. Higher-order features that refer to chains of two or more dependencies, like the ones used in the second-order MST introduced by McDonald and Pereira (2006), are not used. These restrictions make it easier to train on partially annotated sentences without sacrificing accuracy. Flannery et al. (2011) reported that both this parser and McDonald and Pereira (2006)'s second-order MST parser achieved just under 97% accuracy on a Japanese dependency parsing task. The assumption that written Japanese is head-final and that dependencies only go from left to right may be one reason why there is less of a performance gap between these two approaches than in other languages. They also reported that the training time of their parser is fifteen times faster than the second-order MST parser, making it easier to use with active learning.

The following features, both individually and as combination features, are used in the pointwise parser that we adopt.

F1: The distance $j - i$ between a dependent word w_i and its candidate head w_j.

F2: The surface forms w_i and w_j.

F3: The parts-of-speech of w_i and w_j.

F4: The surface forms of up to three words to the left of w_i and w_j.

F5: The surface forms of up to three words to the right of w_i and w_j.

F6: The parts-of-speech of the words selected for F4.

F7: The parts-of-speech of the words selected for F5.

Table 1 shows the values of these features for a partially annotated example sentence where one word, the case marker は (*subj.*), has been annotated with its head, the verb 歓迎 (welcomes). Partial annotation allows annotators to ignore trivial dependencies that are assumed to have little training value.

4 Partial Annotation as a Query Strategy

In this section we give some background on active learning and outline the query strategies that we use to identify informative training examples.

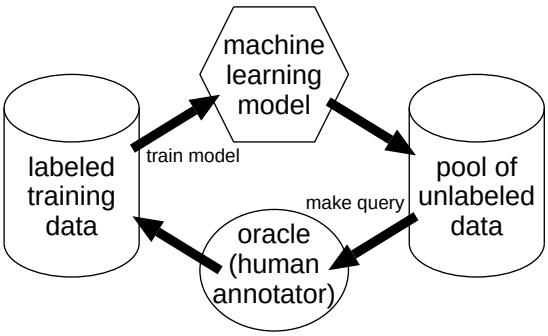

Figure 1: The pool-based active learning cycle.

4.1 Pool-Based Active Learning

We use the pool-based approach to active learning (Lewis and Gale, 1994), because it is a natural fit for domain adaptation. In this framework, we have both initial training data D_L (corresponding to labeled source domain corpora) and a large pool of unlabeled data D_U (corresponding to unlabeled target domain text) from which to choose training examples. While labeling domain-specific text is difficult, it is usually relatively easy to acquire (for example, from the web).

In each iteration the entire pool is evaluated sequentially and its members are ranked by their estimated training value as determined by some criterion, called the query strategy. The top instances are typically selected greedily. The basic flow of pool-based active learning is Figure 1 and described below.

1. Use a base learner B to train a classifier C from the labeled training set D_L.
2. Apply C to the unlabeled data set D_U and select I, the n most informative training examples.
3. Make a query to the oracle for the correct labels of training instances in I.
4. Move training instances in I from D_U to D_L.
5. Train a new classifier C' by applying B to D_L.
6. Repeat steps 2 to 5 until some stopping condition is fulfilled.

The stopping condition for terminating active learning depends on the application. It may be convenient to stop after a classifier C' with a given level of accuracy is reached or a fixed amount

of data has been labeled. In a realistic domain adaptation scenario we are usually concerned with achieving a reasonable level of in-domain performance while keeping down annotation costs, so these kinds of simple termination criteria are sufficient.

4.2 Tree Entropy

Hwa (2004) proposed an active learning query strategy called tree entropy for selecting sentences to be fully annotated. Choosing a parse tree v for a sentence from the set of possible parse trees $\mathcal{V}$ is treated as assigning a value to the random variable V. The entropy of V,

$$H(V) = -\sum_{v \in \mathcal{V}} p(v) \log_2(p(v)), \qquad (1)$$

is equivalent to the expected number of bits needed to encode the distribution of possible parse trees. Here, $p(v)$ is the probability of assigning a single parse tree $V = v$ using a given parsing model. Distributions close to uniform have higher entropy, corresponding to higher uncertainty of the model. Longer sentences will have more parse trees in $\mathcal{V}$ and thus a larger value of $H(V)$. To compare sentences of varying lengths we normalize $H(V)$ by the log of the number of parse trees in $\mathcal{V}$.

$$H_n(V) = \frac{H(V)}{\log_2(|\mathcal{V}|)} \qquad (2)$$

4.3 1-Stage Selection

To use tree entropy as a strategy for partial annotation, we propose to change the unit of selection to words as follows. Consider a word w_i in an input sentence $\boldsymbol{w} = \langle w_1, w_2, \ldots, w_n \rangle$, tagged with POS tags $\boldsymbol{t} = \langle t_1, t_2, \ldots, t_n \rangle$ by a tagger. We will model the distribution of its possible heads, or head entropy. Let w_j be a single head word for w_i, where $j > i$ and $w_j \neq w_i$[1]. Then we can redefine v as a choice of position j and $\mathcal{V}$ as the set of legal values for j. Thus $p(v)$ becomes the probability of choosing the word at position j as the head of the one at position i. The parser we use (Flannery et al., 2011) calculates $p(v) = p(j|i)$ as follows. The feature vector $\boldsymbol{\phi} = \langle \phi_1, \phi_2, \ldots, \phi_m \rangle$ consists of real values calculated from features on pairs (i, j) along with their contexts $\boldsymbol{w}$ and $\boldsymbol{t}$, with corresponding weights given by the parameter vector $\boldsymbol{\theta} = \langle \theta_1, \theta_2, \ldots, \theta_m \rangle$.

$$p(j|\boldsymbol{w}, \boldsymbol{t}, i, \boldsymbol{\theta}) = \frac{\exp\left(\boldsymbol{\theta} \cdot \boldsymbol{\phi}(i, j)\right)}{\sum_{j' \in \mathcal{J}} \exp\left(\boldsymbol{\theta} \cdot \boldsymbol{\phi}(i, j')\right)} \qquad (3)$$

The simplest way to combine this query strategy with partial annotation is to calculate the head entropy for each word appearing in a sentence in the annotation pool, and then choose individual words with the highest head entropy for annotation. We call this query strategy **1-stage**.

4.4 2-Stage Selection

We expect 1-stage to perform well at identifying words with high training value. However, in reality it is difficult to annotate heads for individual words without considering the overall sentence structure, so annotators must consider other dependencies. 1-stage does not realistically model annotation costs.

To address this problem, we propose a novel strategy called 2-stage which more accurately reflects the annotation process. It balances the ability to select fine-grained units for annotation against the difficultly of annotating them.

Words to annotate with heads are chosen in two steps. First, the entropy of each sentence in the pool is calculated by summing the head entropy of its words, and sentences are ranked from highest to lowest summed head entropy. Next, the sentence with the highest head entropy is chosen and the words it contains are ranked in decreasing order by their head entropy. A fixed proportion r of the highest-entropy words are then annotated. This value balances annotation granularity against annotation difficulty. A value of $r = 1.0$ is the standard full annotation case where all words are annotated with heads, which we refer to as **2-stage full**. A value of $r = 0.33$ means that the top 33% of the highest-entropy words in the sentence will be annotated, so we call this strategy **2-stage partial**[2]. In Section 5, we report results for these two values, though several were tried.

5 Evaluation

To evaluate the query strategies, we measured the reduction in target domain annotations needed to

[1] We assume that Japanese is a head-final language, and that each head w_j is located to the right of its dependent w_i in the sentence.

[2] We chose this value because it had good results in preliminary experiments where we tried values in the range from 0.1 to 1.0.

	ID	source	sentences	words	avg. length	dependencies
	EHJ-train	Dictionary examples	11,700	147,964	12.6	136,264
pool	NKN-train	Newspaper articles	9,023	263,425	29.2	254,402
pool	JNL-train	Journal abstracts	322	12,263	38.1	11,941
pool	NPT-train	NTCIR patents	450	18,378	40.8	17,928
test	NKN-test	Newspaper articles	1,002	29,037	29.0	28,035
test	JNL-test	Journal abstracts	32	1,116	34.9	1,084
test	NPT-test	NTCIR patents	50	2,275	45.5	2,225

Table 2: Sizes of corpora.

reach a certain level of in-domain accuracy. For the 2-stage strategy, we also measured how many dependencies a real annotator could annotate in a given time using partial and full annotation. Measuring the actual annotation time is important because our goal of active learning is to reduce the amount of human effort needed to prepare labeled training data for domain adaptation.

We used a corpus of example Japanese sentences from a dictionary as source domain training data (Mori et al., 2014). This data was used as to train the initial model in each experiment. We also collected Japanese text from three target domains: newspapers[3], journal article abstracts, and patents (Goto et al., 2011). For each domain, there is a large annotation pool of potential training examples and a smaller test set. See Table 2 for the details. Domain adaptation is needed in each case, because sentence length and vocabulary differs for each. Words in each sentence were manually segmented and assigned POS automatically with the tagger KyTea. This step can be done automatically because KyTea's F-measure score for word segmentation and POS tagging is about 98% (Neubig et al., 2011). Words were then manually annotated with their heads.

5.1 Number of Annotations

We first investigate how much strategies reduce the *number* of in-domain dependencies needed for domain adaptation. Because real annotation is costly and not strictly necessary to measure this reduction, we simulate active learning by selecting the gold standard dependency labels from the annotation pool. In practice, we are also concerned with the *time* needed for a human to annotate dependencies, which we examine in Section 5.3. Thus, good performance in this first experiment is a necessary but not sufficient condition for an effective strategy. Because we assume that Japanese is a head-final language and heads always occur to the right of their dependents, for all strategies the last word in each sentence skipped. For 1-stage and 2-stage, we also skipped the second-to-last word in each sentence.

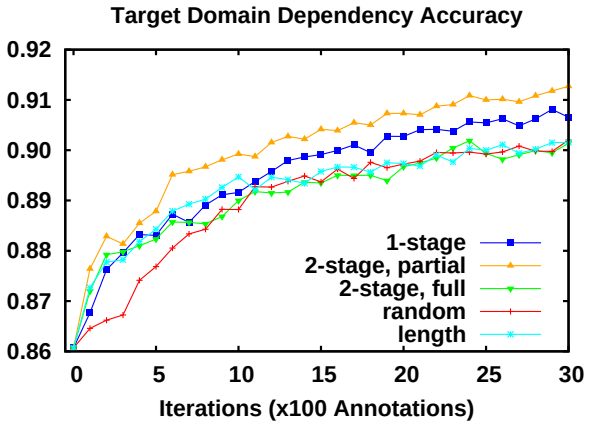

Figure 2: Newspaper (NKN) domain learning curves.

In addition to the 1-stage and 2-stage methods, we also tested two simple baselines. The strategy **random** simply selects words randomly from the pool. The **length** strategy simply chooses words with the longest possible dependency length[4]. This strategy reflects our intuition that long-distance dependencies are more difficult and thus more informative.

We use the dictionary example sentences (see Table 2) as the initial training set and performed thirty iterations of active learning. In each iteration, we select a batch of one hundred target domain dependency annotations, retrain the model, and then measure its in-domain accuracy.

[3]The newspaper is similar to the Wall Street Journal and focuses on economics.

[4]This is the same as selecting dependencies with the largest number of potential heads because we do not refer to the gold dependencies until after words have been selected.

Figure 2 shows the results for the newspaper domain. The accuracy of the random strategy increases slowly and peaks at just over 90.5%. For the first ten iterations the length strategy delivers an improvement over random, but performs essentially the same after that. This is probably because newspaper sentences are on average longer than dictionary examples (see Table 2), so at first words with the potential for longer dependencies are slightly more informative. However, this strategy is focused only on the training data and does not reflect the continuous updates of the model, and it soon begins to falter.

The 2-stage partial strategy dominates all other methods, though 1-stage reaches the same level after thirty-five iterations. Its peak accuracy is slightly higher than 91%, and it outperforms the best accuracy achieved by random after just seventeen iterations. In contrast, 2-stage full performs consistently worse than the partial annotation version, with behavior similar to length. While the 1-stage strategy always outperforms the random one, it lags behind 2-stage partial.

5.2 Annotation Pool Size

From Table 2, we can see that the size of the annotation pool for the newspaper domain is ten to twenty times as large as the ones for the other domains. The total number of dependencies selected is 3k, which is only 1.2% of the newspaper pool. Because the 2-stage strategy chooses some dependencies with lower entropy over competing ones with higher entropy from other sentences in the pool, we expect its accuracy to suffer when a much larger fraction of the pool is selected.

To investigate this effect, we created a smaller pool from NKN-train that is closer in size to the ones from the other domains. We used the first 12,165 dependencies for this smaller pool. The results are shown in Figure 3. It can be seen that 2-stage partial's lead over the 1-stage strategy has been eliminated. After seventeen rounds of annotation the 1-stage strategy begins to outperform the 2-stage strategy. The 2-stage partial strategy still dominates the 2-stage full strategy. This confirms our intuition that the relative performance of strategies is influenced by the size of the annotation pool. In general we expect the number of informative dependencies to increase as the pool size increases. Comparing these results with the results for the newspaper domain in Figure 2, we see that

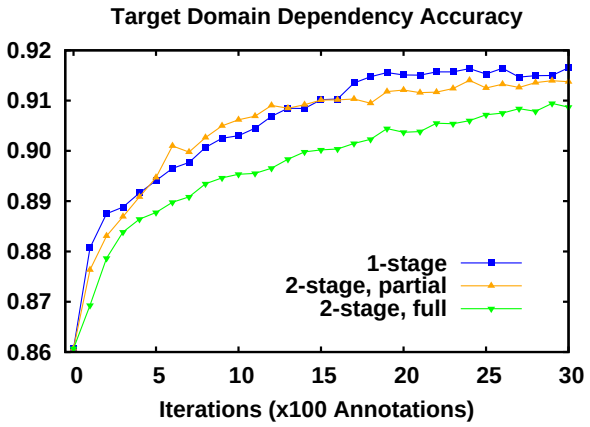

Figure 3: Newspaper (NKN) domain accuracy with a small annotation pool.

the 1-stage strategy is robust to changes in the pool size, but the 2-stage partial can outperform it for a very large pool.

5.3 Time Required for Annotation

Simulation experiments are still common when using active learning because the cost of annotation is very high. However, recently there is increased interest in measuring the true costs of annotation work when doing active learning (Settles et al., 2008). For a more realistic evaluation of active learning for parsing, we also measured annotation time for the 2-stage strategy. We trained a model on EHJ-train plus NKN-train and used this model and the 2-stage strategy to select dependencies to be annotated by a human annotator. The pool is 747 blog sentences[5] from the Balanced Corpus of Contemporary Written Japanese (Maekawa, 2008). We selected 2k dependencies in a single iteration so the annotator did not need to wait while the model was retrained after each batch of annotations. While real annotation times are not constant, this simplification is justified because we expect the annotation strategy (partial or full) to have a larger effect on the overall annotation speed than the dependencies that are selected.

A single annotator performed annotations for one hour each using the 2-stage strategy with both partial annotation and full annotation, alternating strategies every fifteen minutes. Sentences with more than forty words were not presented to the annotator. Table 3 shows the total number of dependencies annotated after each time period. After

[5]This data was taken from the Yahoo! Blog (OY) subcorpus.

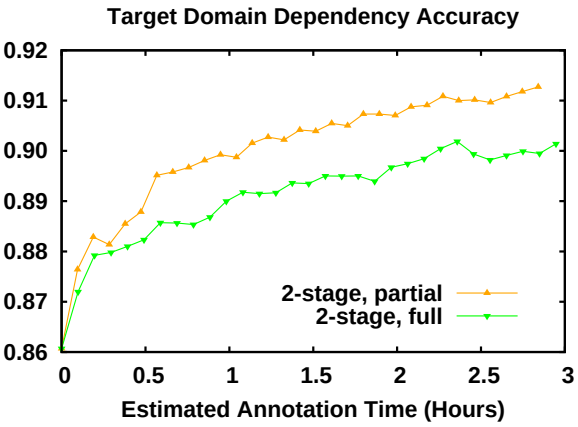

Figure 4: Estimated annotation time for the newspaper (NKN) domain.

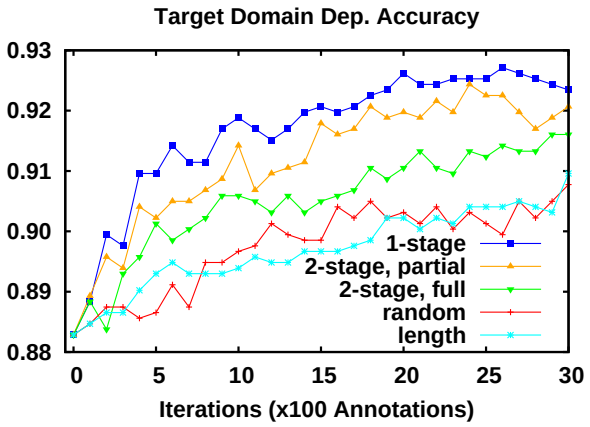

Figure 5: Journal (JNL) domain learning curves.

method	0.25 [h]	0.5 [h]	0.75 [h]	1.0 [h]
partial	226	458	710	1056
full	141	402	756	1018

Table 3: Annotation times for 2-stage methods.

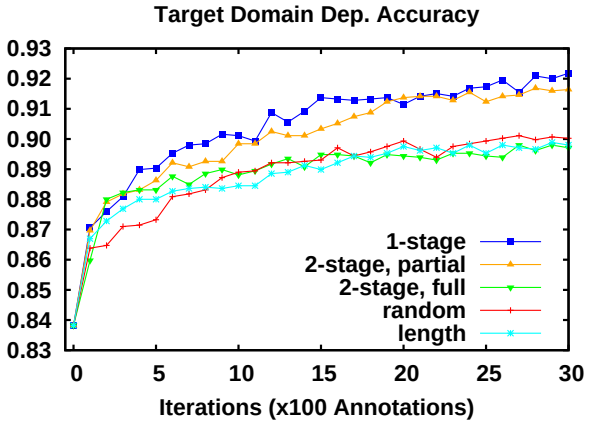

Figure 6: Patent (NPT) domain learning curves.

the first fifteen minutes, the annotator completed 226 annotations compared with 141 for full annotation, an increase of about 60%. However, as time progresses the difference becomes smaller, and after one hour the number of annotations was almost identical for both strategies.

From Table 3, we can see that the annotation speed reaches a maximum of about 350 annotations per fifteen minutes in the full annotation case, or 1.4k dependencies per hour. We expected more annotations to be completed when full annotation was used, because sentences have many trivial dependencies. However, the annotator reported that it was frustrating to check the annotation standard and how it handled subtle linguistic phenomena. Most of this work can be skipped when using partial annotation because the annotator was allowed to delete the estimated heads, so the annotation speeds ended up being almost identical. This result shows the importance of accurately modeling the annotation costs in active learning.

For both methods, the average speed is around 1k dependencies per hour. We used these speeds to estimate the rate of annotation for the experiments from Section 5.1. While this is not entirely realistic because speeds are likely to vary across domains, it is sufficient for comparing the relative performance of strategies in the same domain. The results are shown in Figure 4. We can see that accuracy improves faster for partial than it does for full, and the difference becomes pronounced after about half an hour of annotation work. In summary, partial annotation is more efficient and thus delivers a greater return on investment than full annotation for the proposed query strategy.

5.4 Results for Additional Domains

We also tested the proposed method in two additional domains. See Table 2 for the details of these corpora. Figure 5 and Figure 6 show results for the journal and patent domains, respectively. In these domains, 2-stage partial failed to outperform the 1-stage strategy. However, it still performed better than the 2-stage full strategy. As discussed in Section 5.2, the performance of the proposed method suffers when a large portion of the dependencies in the pool are selected. Here, the 3k dependencies that are selected are a much larger fraction of the pool – specifically, 16.7% for the patent domain and 25.1% for the journal domain. As in the

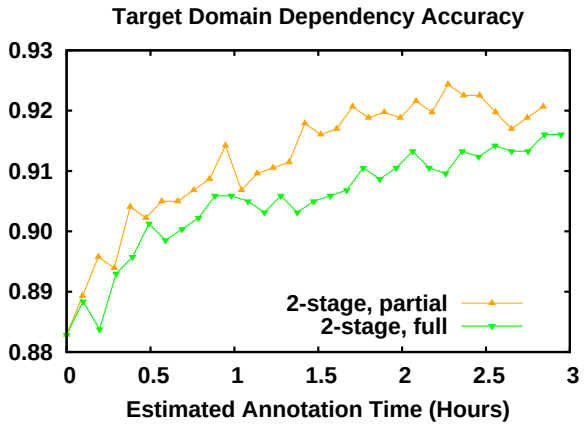

Figure 7: Estimated annotation time for the journal (JNL) domain.

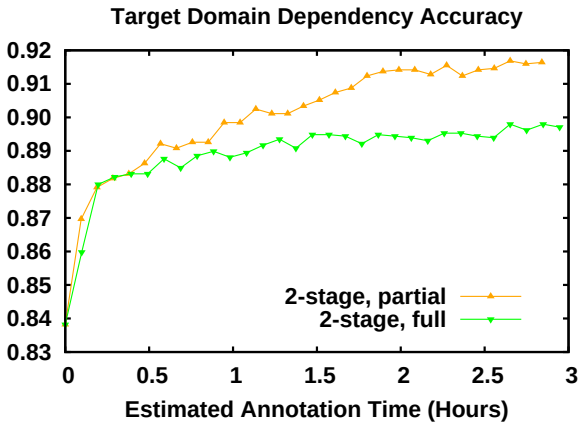

Figure 8: Estimated annotation time for the patent (NPT) domain.

domain	random	full	partial
NKN	3,000	–	1,300
JNL	3,000	1,800	900
NPT	2,700	–	1,500

Table 4: Reduction in in-domain data.

newspaper domain, in the patent domain the performance of 2-stage with full annotation is better than random for the first few iterations but soon becomes similar. This is not true in the journal domain, where this strategy consistently beats random. The length strategy edges out random for a few iterations in both domains, but ultimately their performance is similar.

Table 4 shows the number of annotations needed for the highest accuracy by the random baseline in the second column, while the next two show the number of annotations needed for the full and partial versions of 2-stage to outperform it. Thus, smaller numbers are better. Compared to the random strategy, 2-stage full had mixed results. In the journal abstract domain (JNL), it outperformed the random baseline while using only 60% of the amount of labeled data. However, it failed to outperform random selection in the other two domains. In contrast, 2-stage partial consistently outperforms random with only 45% to 55% of the labeled data. In terms of target domain data that must be prepared, it is clear that 2-stage partial offers large savings compared to random. It also does so more consistently and with less data than 2-stage full.

We also plotted the results for these domains in terms of estimated annotation time as we did in Section 5.3. Figure 7 shows the results for the journal domain and Figure 8 shows the results for the patent domain. As before, 2-stage full is more efficient than 2-stage partial. In these domains, partial dominates full after about one hour of annotation work. The gap between them is largest for the patent domain and smallest for the journal domain.

6 Conclusions

We combined partial annotation with active learning to adapt a Japanese dependency parser to new domains, and showed that active learning is not limited to single-domain settings. We showed that an entropy-based query strategy can successfully identify units smaller than sentences, and that partial annotation can be successfully applied to active learning of structured prediction tasks like parsing. This strategy reduced the amount of in-domain training data needed for domain adaptation by up to 75%. We also investigated how the overall size of the annotation pool affects the performance of the query strategy, and found that the proposed method performs best for large annotation pools.

To more accurately frame our results, we measured the actual annotation time required by a human annotator to prepare labeled data using different strategies. Using these results to estimate annotation times for earlier experiments, we showed that for the proposed method partial annotation is more efficient in terms of in-domain performance obtained per unit of annotation time than full annotation.

Acknowledgments

This work was supported by JSPS Grants-in-Aid for Scientific Research Grant Numbers 26280084 and 26540190, NTT agreement dated 06/19/2015, and Basic Research on Corpus Annotation project of The National Institute for Japanese Language and Linguistics. We are also grateful to the annotators for their contribution to the design of the annotation guidelines and their efforts in following them.

References

Daniel Flannery, Yusuke Miayo, Graham Neubig, and Shinsuke Mori. 2011. Training dependency parsers from partially annotated corpora. In *Proceedings of the Fifth International Joint Conference on Natural Language Processing*, Chiang Mai, Thailand.

Isao Goto, Bin Lu, Ka Po Chow, Eiichiro Sumita, and Benjamin K. Tsou. 2011. Overview of the patent machine translation task at the NTCIR-9 workshop. In *Proceedings of NTCIR-9 Workshop Meeting*, pages 559–578.

R. Hwa. 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30(3).

D. Lewis and W. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual ACM SIGIR conference on research and development in information retrieval*.

Kikuo Maekawa. 2008. Balanced corpus of contemporary written Japanese. In *Proceedings of the 6th Workshop on Asian Language Resources*.

Ryan McDonald and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(4):197–230.

Ryan McDonald and Fernando Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *Proceedings of the Eleventh European Chapter of the Association for Computational Linguistics*, volume 6.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530.

Seyed Abolghasem Mirroshandel and Alexis Nasr. 2011. Active learning for dependency parsing using partially annotated sentences. In *Proceedings of the 12th International Conference on Parsing Technologies*, Dublin, Ireland.

Shinsuke Mori, Hideki Ogura, and Tetsuro Sasada. 2014. A Japanese word dependency corpus. In *Proceedings of the Nineth International Conference on Language Resources and Evaluation*, pages 753–758.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*.

F. Olsson. 2009. A literature survey of active machine learning in the context of natural language processing. Technical Report T2009:06, Swedish Institute of Computer Science.

Manabu Sassano and Sadao Kurohashi. 2010. Using smaller constituents rather than sentences in active learning for Japanese dependency parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

B. Settles, M. Craven, and L. Friedland. 2008. Active learning with real annotation costs. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*.

Kathrin Spreyer, Lilja Øvrelid, and Jonas Kuhn. 2010. Training parsers on partial trees: a cross-language comparison. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*.

Min Tang, Xiaoqiang Luo, and Salim Roukos. 2002. Active learning for statistical natural language parsing. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.