Combining Active Learning and Partial Annotation for Japanese Dependency Parsing

Daniel FLANNERY[1] Yusuke MIYAO[2] Shinsuke MORI[1] Tatsuya KAWAHARA[1]
[1]Graduate School of Informatics, Kyoto University [2]National Institute of Informatics

1 Introduction

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 how active learning can be used for domain adaptation of dependency parsers, and how entropy-based strategies can be used to select smaller units for annotation. We show how these strategies can be combined with partial annotation to annotate informative examples in the new domain without annotating full sentences. Research on active learning often uses simulations, but we measured the actual time needed by a human annotator for annotation work to better frame the results obtained in our simulations.

We evaluate 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. While annotation speeds were similar for both, the proposed partial annotation strategy delivers better in-domain performance for the same amount of human effort.

2 Background

There has been much recent work on active learning for a variety of natural language processing tasks [6], but most of it is concerned only with the single-domain case. 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.

Previous work on active learning for structured prediction tasks like parsing [2] often assumes that the training data must be fully annotated. But recent work on dependency parsing [1] 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.

2.1 Pool-Based Active Learning

We use the pool-based approach to active learning [3], because it is a natural fit for domain adaptation. In this framework, we begin with a small amount of labeled initial training data D_L (corresponding to labeled source domain corpora) and a large pool of unlabeled initial data D_U (corresponding to unlabeled target domain text) from which to choose training examples.

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 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.

2.2 Query Strategy Design

An important difference from previous work is how we apply active learning for parsing. Both Sassano and Kurohashi [7] and Hwa [2] studied the single-domain case, where the initial labeled data set and the pool of unlabeled data share the same domain. This paper focuses on adaptation from one domain to another.

Previous work on active learning for parsing [8, 2] has focused on selecting sentences to be fully annotated. Sassano et al. [7] 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 annotated sentences, but did not use entropy-based query strategies [8, 2] 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.

2.3 Pointwise Dependency Parsing

The motivation for this approach is to design a query strategy that allows for partial annotation of individual words. We believe that this type of query strategy will be useful when adapting parsers to new domains. We use a pointwise dependency parser [1] in our experiments. In the pointwise approach to dependency parsing, each word's head is predicted independently. 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 are used. Higher-order features that refer to chains of two or more dependencies are not used. These restrictions make it easier to train on partially annotated sentences.

2.4 Tree Entropy

Hwa [2] 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,

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

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. Spiked distributions, corresponding to higher uncertainty of the model, have higher entropy. Longer sentences will have more parse trees in $\mathcal{V}$ and thus 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 $\mathcal{V}$.

3 Partial Annotation as a Query Strategy

3.1 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 part-of-speech (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 [1] calculates $p(v) = p(j|i)$ as follows. The feature vector $\boldsymbol{\phi} = \langle \phi_1, \phi_2, \ldots, \phi_m \rangle$ consists of non-negative 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 $\boldsymbol{\theta} = \langle \theta_1, \theta_2, \ldots, \theta_m \rangle$.

$$p(j|\boldsymbol{w}, \boldsymbol{t}, i, \boldsymbol{\theta}) = \frac{\exp(\boldsymbol{\theta} \cdot \boldsymbol{\phi}(x, j))}{\sum_{j' \in \mathcal{J}} \exp(\boldsymbol{\theta} \cdot \boldsymbol{\phi}(x, j'))} \tag{2}$$

The simplest way to combine this query strategy with partial annotation is to choose individual words from the pool with the highest head entropy. We call this query strategy 1-stage.

3.2 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, and $r = 0.33$ means that the top 33% of the highest-entropy words in the sentence will be annotated. In Section 4, we report results for these two values, though several were tried.

4 Evaluation

As an evaluation of the query strategies we proposed, we measured the reduction in annotated dependencies in the target domain needed to 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.

We used a corpus of example Japanese sentences from a dictionary as source domain training data. We also collected Japanese text from three target domains: newspapers, journal article abstracts, and patents. See Table 1 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 annotated with their heads. POS tags were automatically assigned with the tagger KyTea [5].

4.1 Number of Annotations

We first investigate how much the proposed strategies reduce the *number* of in-domain dependencies needed for domain adaptation. Because real annotation is very costly and not strictly necessary to measure this reduction, we simulate active learning by selecting gold 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 4.2. 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 is never chosen. For 1-stage and 2-stage, we also skipped the second-to-last word in each sentence.

In addition to the 1-stage and 2-stage methods, we also tested two simple baselines. The strategy random

[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.

Table 1: Sizes of Corpora.

	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

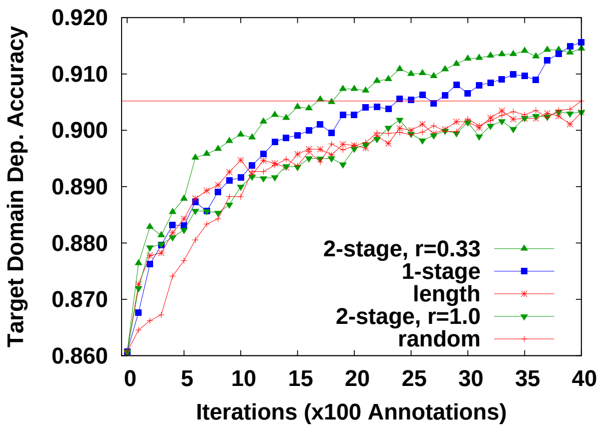

Figure 1: Newspaper (NKN) domain learning curves.

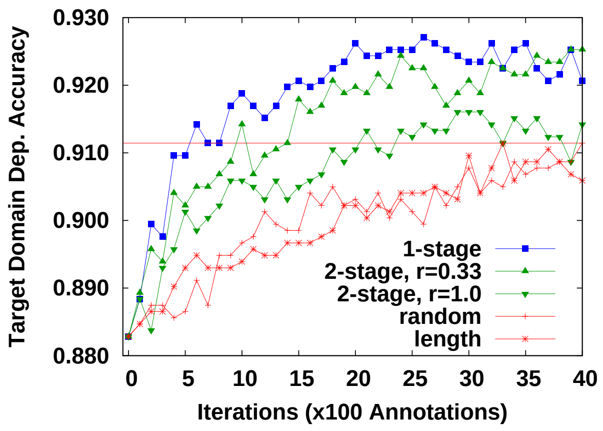

Figure 2: Journal (JNL) domain learning curves.

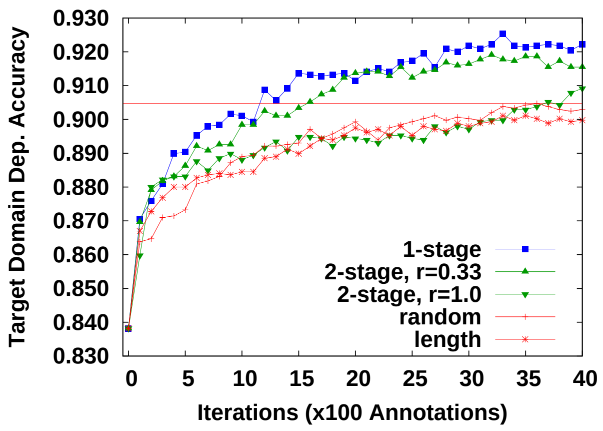

Figure 3: Patent (NPT) domain learning curves.

simply selects words randomly from the pool. The length strategy simply chooses words with the longest possible dependency length[2]. This strategy reflects our intuition that long-distance dependencies are more difficult.

We use EHJ-train as the initial training set and performed 40 iterations of active learning. In each iteration, we select 100 target domain dependency annotations, retrain the model, and then measure its in-domain accuracy.

Figure 1 shows the results for the newspaper domain. The accuracy of the random strategy increases slowly and peaks at just over 90.5%. The horizontal line shows the highest accuracy achieved by random. 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 1), 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 strategy with partial annotation ($r = 0.33$) dominates all other methods, though 1-stage reaches the same level after 35 iterations. Its peak accuracy reaches 91.5%, and it outperforms the best accuracy achieved by random after just 17 iterations. In contrast, the 2-stage strategy with full annotation ($r = 1.0$) performs consistently worse than the partial annotation version, with behavior similar to length. While 1-stage always outperforms random, somewhat surprisingly it lags behind 2-stage with partial annotation.

Figure 2 and Figure 3 show results for the journal and patent domains, respectively. For these domains, 2-stage with partial annotation failed to outperform 1-stage. However, it still performed better than the same strategy with full annotation. These results suggest that partial annotation is valuable in domains where sentences are usually very long. As in the 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 2 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 by 1-stage and 2-stage to outperform it. Numbers in parentheses show the difference between the second column as a percentage. Thus, larger percentages are better. 1-stage reduces the labeled in-domain data by 67% to 82% in each domain. In comparison, 2-stage has slightly lower reductions, ranging from 56% to 75%. Reductions were largest for the journal domain and

[2]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.

Table 2: Reduction in In-Domain Data.

domain	rand	1-stage	2-stage, r=0.33
NKN	4000	700 (82%)	1700 (58%)
JNL	4000	700 (82%)	1000 (75%)
NPT	3600	1200 (67%)	1600 (56%)

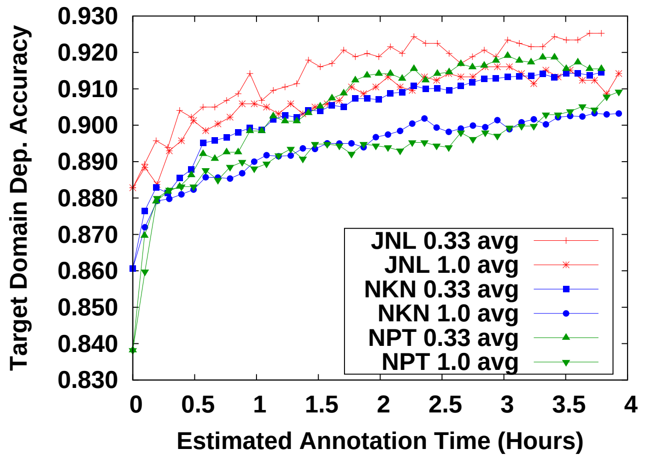

Figure 4: Learning curves in each domain.

smallest for the patent domain. We can see that entropy-based strategies substantially reduce the amount of target domain data needed for domain adaptation.

4.2 Time Required for Annotation

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[3] from the Balanced Corpus of Contemporary Written Japanese [4]. We selected 2k dependencies in a single iteration so the annotator did not need to wait while the model was retrained after each 100 annotations. While the annotation cost for dependencies is 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 for both $r = 0.33$ (partial annotation) and $r = 1.0$ (full annotation), alternating strategies every fifteen minutes. Sentences with more than forty words were not presented. Table 3 shows the total number of dependencies annotated after each time period. After 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 in methods becomes smaller, and after one hour the number of annotations was almost identical for both strategies.

Because several dependencies in a sentence are trivial, we expected the annotator to complete more when using full annotation case. However, the annotator reported that he was forced to spend much more time checking the annotation standard and examples of complicated linguistic phenomena in the case of full annotation, which increased the annotation time. He was able to skip much of this work when using partial annotation. This result shows the need for realistic models of annotation costs

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

Table 3: Number of Dependencies Annotated.

method	0.25 [h]	0.5 [h]	0.75 [h]	1.0 [h]
r=0.33	226	458	710	1056
r=1.0	141	402	756	1018

in active learning.

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. For both methods, the average speed is around 1k dependencies per hour (as shown in the fourth column). Figure 4 uses these average speeds to estimate the rate of annotation for the experiments from Section 4.1. While this is not entirely realistic because annotation speeds are likely to vary across domains, it is sufficient to compare strategies within the same domain. In each domain, we can see that accuracy improves at a faster rate for partial annotation than it does for full annotation. The gap between them is largest for the patent domain and smallest for the journal domain.

5 Conclusions

We combined partial annotation with active learning to adapt a Japanese dependency parser to new domains. We showed that an entropy-based query strategy can successfully identify units smaller than sentences, allowing annotators to use partial annotation. This strategy reduces the amount of in-domain training data that must be labeled for domain adaptation by up to 75%. To more accurately frame these results, we measured the annotation time a human annotator took to prepare labeled data using different strategies. While full and partial annotation had similar speeds, partial annotation better identified informative examples and delivered the most in-domain accuracy improvements in terms of time spent on annotation.

References

[1] D. Flannery, Y. Miyao, G. Neubig, and S. Mori. A pointwise approach to training dependency parsers from partially annotated corpora. *Journal of Natural Language Processing*, 19(3), 2012.

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

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

[4] K. Maekawa. Balanced corpus of contemporary written Japanese. In *Proceedings of the 6th Workshop on Asian Language Resources*, 2008.

[5] G. Neubig, Y. Nakata, and S. Mori. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *ACL-HLT 2011*, 2011.

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

[7] M. Sassano and S. Kurohashi. Using smaller constituents rather than sentences in active learning for Japanese dependency parsing. In *ACL 2010*, 2010.

[8] M. Tang, X. Luo, and S. Roukos. Active learning for statistical natural language parsing. In *ACL 2002*, 2002.