Unsupervised Parse Selection for HPSG

Rebecca Dridan and **Timothy Baldwin**
Dept. of Computer Science and Software Engineering
University of Melbourne, Australia
rdridan@csse.unimelb.edu.au, tb@ldwin.net

Abstract

Parser disambiguation with precision grammars generally takes place via statistical ranking of the parse yield of the grammar using a supervised parse selection model. In the standard process, the parse selection model is trained over a hand-disambiguated treebank, meaning that without a significant investment of effort to produce the treebank, parse selection is not possible. Furthermore, as treebanking is generally streamlined with parse selection models, creating the initial treebank without a model requires more resources than subsequent treebanks. In this work, we show that, by taking advantage of the constrained nature of these HPSG grammars, we can learn a discriminative parse selection model from raw text in a purely unsupervised fashion. This allows us to bootstrap the treebanking process and provide better parsers faster, and with less resources.

1 Introduction

Parsing with precision grammars is generally a two-stage process: (1) the full parse yield of the precision grammar is calculated for a given item, often in the form of a packed forest for efficiency (Oepen and Carroll, 2000; Zhang et al., 2007); and (2) the individual analyses in the parse forest are ranked using a statistical model ("parse selection"). In the domain of treebank parsing, the Charniak and Johnson (2005) reranking parser adopts an analogous strategy, except that ranking and pruning are incorporated into the first stage, and the second stage is based on only the top-ranked parses from the first stage. For both styles of parsing, however, parse selection is based on a statistical model learned from a pre-existing treebank associated with the grammar. Our interest in this paper is in completely removing this requirement of parse selection on explicitly treebanked data, ie the development of fully unsupervised parse selection models.

The particular style of precision grammar we experiment with in this paper is HPSG (Pollard and Sag, 1994), in the form of the DELPH-IN suite of grammars (`http://www.delph-in.net/`). One of the main focuses of the DELPH-IN collaboration effort is multilinguality. To this end, the Grammar Matrix project (Bender et al., 2002) has been developed which, through a set of questionnaires, allows grammar engineers to quickly produce a core grammar for a language of their choice. Bender (2008) showed that by using and expanding on this core grammar, she was able to produce a broad-coverage precision grammar of Wambaya in a very short amount of time. However, the Grammar Matrix can only help with the first stage of parsing. The statistical model used in the second stage of parsing (ie parse selection) requires a treebank to learn the features, but as we explain in Section 2, the treebanks are created by parsing, preferably *with* a statistical model. In this work, we look at methods for bootstrapping the production of these statistical models without having an annotated treebank. Since many of the languages that people are building new grammars for are under-resourced, we can't depend on having any external information or NLP tools, and so the methods we examine are purely unsupervised, using nothing more than the grammars themselves and raw text. We find that, not only can we produce models that are suitable for kick-starting the treebanking process, but the accuracy of these models is comparable to parsers trained on gold standard data (Clark and Curran, 2007b; Miyao and Tsujii, 2008), which have been successfully used in applications (Miyao et al., 2008).

2 The problem

The current method of training a parse selection model uses the [incr tsdb()] treebanking mechanism (Oepen, 2001) and works well for updating models for mature grammars, although even for these grammars, building a new model for a different domain requires a time-consuming initial treebanking effort. The treebanks used with DELPH-IN grammars are *dynamic* treebanks (Oepen et al., 2004) created by parsing text and having an annotator select the correct analysis (or discard all of them). The annotation process involves making binary decisions based on so-called parse discriminants (Carter, 1997). Whenever the grammar is changed, the treebank can be quickly updated by re-parsing and re-applying the old annotation decisions. This treebanking process not only produces gold standard trees, but also a set of non-gold trees which provides the negative training data necessary for a discriminative maximum entropy model.

The standard process for creating a parse selection model is:

1. parse the training set, recording up to 500 highest-ranking parses for each sentence;

2. treebank the training set;

3. extract features from the gold and non-gold parses;

4. learn feature weights using the TADM toolkit.[1] (Malouf, 2002)

The useful training data from this process is the parses from those sentences for which: more than one parse was found; and at least one parse has been annotated as correct. That is, there needs to be both gold and non-gold trees for any sentence to be used in training the discriminative model.

[1] `http://tadm.sourceforge.net/`

There are two issues with this process for new grammars. Firstly, treebanking takes many person-hours, and is hence both time-consuming and expensive. Complicating that is the second issue: N-best parsing requires a statistical model. While it is possible to parse exhaustively with no model, parsing is much slower, since the unpacking of results is time-consuming. Selective unpacking (Zhang et al., 2007) speeds this up a great deal, but requires a parse selection model. Treebanking is also much slower when the parser must be run exhaustively, since there are usually many more analyses to manually discard.

This work hopes to alleviate both problems. By producing a statistical model without requiring human treebanking, we can have a working and efficient parser with less human effort. Even if the top-1 parses this parser produces are not as accurate as those trained on gold standard data, this model can be used to produce the N-best analyses for the treebanker. Since our models are much better than random selection, we can afford to reduce N and still have a reasonable likelihood that the correct parse is in that top N, making the job of the treebanker much faster, and potentially leading to even better parse selection accuracy based on semi-supervised or fully-supervised parse selection.

3 Data and evaluation

Our ultimate goal is to use these methods for under-resourced languages but, since there are no pre-existing treebanks for these languages, we have no means to measure which method produces the best results. Hence, in this work, we experiment with languages and grammars where we have gold standard data, in order to be able to evaluate the quality of the parse selection models. Since we have gold-standard trained models to compare with, this enables us to fully explore how these unsupervised methods work, and show which methods are worth trying in the more time-consuming and resource-intensive future experiments on other languages. It is worth reinforcing that the gold-standard data is used for evaluation only, except in calculating the supervised parse selection accuracy as an upper-bound.

The English Resource Grammar (ERG:

Language	Sentences	Average words	Average parses
Japanese	6769	10.5	49.6
English	4855	9.0	59.5

Table 1: Initial model training data, showing the average word length per sentence, and also the ambiguity measured as the average number of parses found per sentence.

Flickinger (2002)) is an HPSG-based grammar of English that has been under development for many person years. In order to examine the cross-lingual applicability of our methods, we also use Jacy, an HPSG-based grammar of Japanese (Siegel and Bender, 2002). In both cases, we use grammar versions from the "Barcelona" release, from mid-2009.

3.1 Training Data

Both of our grammars come with statistical models, and the parsed data and gold standard annotations used to create these models are freely available. As we are trying to simulate a fully unsupervised setup, we didn't want any influence from these earlier models. Hence, in our experiments we used the parsed data from those sentences that received less than 500 parses and ignored any ranking, thus annulling the effects of the statistical model. This led to a reduced data set, both in the number of sentences, and in the fact that the more ambiguous sentences were discarded, but it allows clean comparison between different methods, without incorporating external information. The details of our training sets are shown in Table 1,[2] indicating that the sentence lengths are relatively short, and hence the ambiguity (measured as average parses per sentence) is low for both our grammars. The ambiguity figures also suggest that the Japanese grammar is more constrained (less ambiguous) than the English grammar, since there are, on average, more parses per sentence for English, even with a lower average sentence length.

3.2 Test Data

The test data sets used throughout our experiments are described in Table 2. The *tc-006* data set is from the same Tanaka Corpus (Tanaka, 2001) which was used for the Japanese training data. There is a wider variety of treebanked data available for the English grammar than for the Japanese. We use the *jhpstg$_t$* data set, which consists of text from Norwegian tourism brochures, from the same LOGON corpus as the English training data (Oepen et al., 2004). In order to have some idea of domain effects, we also use the *catb* data set, the text of an essay on open-source development.[3] We see here that the sentences are longer, particularly for the English data. Also, since we are not artificially limiting the parse ambiguity by ignoring those with 500 or more parses, the ambiguity is much higher. This ambiguity figure gives some indication of the difficulty of the parse selection task. Again we see that the English sentences are more ambiguous, much more in this case, making the parse selection task difficult. In fact, the English ambiguity figures are an under-estimate, since some of the longer sentences timed out before producing a parse count. This ambiguity can be a function of the sentence length or the language itself, but also of the grammar. A more detailed and informative grammar makes more distinctions, not all of which are relevant for every analysis.

Test Set	Language	Sentences	Average words	Average parses
tc-006	Japanese	904	10.7	383.9
jhpstg$_t$	English	748	12.8	4115.1
catb	English	534	17.6	9427.3

Table 2: Test data, showing the average word length per sentence, and also the ambiguity measured as the average number of parses found per sentence. Note that the ambiguity figures for the English test sets are under-estimates, since some of the longer sentences timed out before giving an analysis count.

3.3 Evaluation

The exact match metric is the most common accuracy metric used in work with the DELPH-IN tool set, and refers to the percentage of sentences for which the top parse matched the gold parse in every way. This is akin to the sentence accuracy that is occasionally reported in the parsing literature, except

[2]Any sentences that do not have both gold and non-gold analyses (ie, had no correct parse, only one parse, or none) are not included in these figures.

[3]The Cathedral and the Bazaar, by Eric Raymond. Available from: `http://catb.org/esr/writings/cathedral-bazaar/`

that it also includes fine-grained syntactico-semantic features that are not often present in other parsing frameworks. Exact match is a useful metric for parse selection evaluation, but it is very blunt-edged, and gives no way of evaluating *how close* the top parse was to the gold standard. Since these are very detailed analyses, it is possible to get one detail wrong and still have a useful analysis. Hence, in addition to exact match, we also use the EDM$_{NA}$ evaluation defined by Dridan (2009). This is a predicate–argument style evaluation, based on the semantic output of the parser (MRS: Minimal Recursion Semantics (Copestake et al., 2005)). This metric is broadly comparable to the predicate–argument dependencies of CCGBank (Hockenmaier and Steedman, 2007) or of the ENJU grammar (Miyao and Tsujii, 2008), and also somewhat similar to the grammatical relations (GR) of the Briscoe and Carroll (2006) version of DepBank. The EDM$_{NA}$ metric matches triples consisting of predicate names and the argument type that connects them.[4]

4 Initial Experiments

All of our experiments are based on the same basic process: (1) for each sentence in the training data described in Section 3.1, label a subset of analyses as correct and the remainder as incorrect; (2) train a model using the same features and learner as in the standard process of Section 2; (3) parse the test data using that model; and (4) evaluate the accuracy of the top analyses. The differences lay in how the 'correct' analyses are selected each time. Each of the following sections detail different methods for nominating which of the (up to 500) analyses from the training data should be considered pseudo-gold for training the parse selection model.

4.1 Upperbound and baseline models

As a first step we evaluated each data set using an upperbound and a baseline model. The upperbound model in this case is the model trained with gold standard annotations. These accuracy figures are slightly lower than others found in the literature for this data, since, to allow for comparison, we limited the training data to the sets described in Table 1.

[4]The full EDM metric also includes features such as tense and aspect, but this is less comparable to the other metrics mentioned.

Test Set	Exact Match	EDM		
		Precision	Recall	F-score
tc-006	72.90	0.961	0.957	0.959
jhpstg$_t$	48.07	0.912	0.908	0.910
catb	22.29	0.838	0.839	0.839

Table 3: Accuracy of the gold standard-based parse selection model.

Test Set	Exact Match	EDM		
		Precision	Recall	F-score
tc-006	17.26	0.779	0.839	0.807
jhpstg$_t$	12.48	0.720	0.748	0.734
catb	8.30	0.646	0.698	0.671

Table 4: Accuracy of the baseline model, trained on randomly selected pseudo-gold analyses.

By throwing out those sentences with more than 500 parses, we exclude much of the data that is used in the standard model and so our exact match figures are slightly lower than might be expected.

For the baseline model, we used random selection to select our gold analyses. For this experiment, we randomly assigned one parse from each sentence in the training data to be correct (and the remainder of analyses as incorrect), and then used that 'gold standard' to train the model. Results for the upperbound and baseline models are shown in Tables 3 and 4.

As expected, the results for Japanese are much higher, since the lower ambiguity makes this an easier task. The *catb* test set results suffer, not only from being longer, more ambiguous sentences, but also because it is completely out of the domain of the training data.

The exact match results from the random baseline are approximately what one might expect, given the respective ambiguity levels in Table 2. The EDM figures are perhaps higher than might be expected given random selection from the entire parse forest. This results from using a precision grammar, with an inbuilt notion of grammaticality, hence constraining the parser to only produce somewhat reasonable parses, and creating a reasonably high baseline for our parse selection experiments.

We also tried a separate baseline, eliminating the parse selection model altogether, and using random selection directly to select the top analysis. The exact match and EDM precision results were slightly lower than using random selection to train a model, which may be due to the learner giving weight to features that are common across the training data, but the differences weren't significant. Recall was significantly lower when using random selection directly, due to the time outs caused by running without a model. For this reason, we use the random selection-based model results as our baseline for the other unsupervised parse selection models, noting that correctly identifying approximately three quarters of the dependencies in the *jhpstg*$_t$ set, and over 80% when using the Japanese grammar, is a fairly high baseline.

4.2 First attempts

As a first approach to unsupervised parse selection, we looked at two heuristics to designate some number of the analyses as 'gold' for training. Both of these heuristics looked independently at the parses of each sentence, rather than calculating any numbers across the whole training set.

The first method builds on the observation from the random selection-based model baseline experiment that just giving weight to common features could improve parser accuracy. In this case, we looked at the edges of the parsing chart. For each sentence, we counted the number of times an edge was present in an analysis, and used that number (normalised by the total number of times any edge was used) as the *edge weight*. We then calculated an analysis score by summing the edge weights of all the edges in that analysis, and dividing by the number of edges, to give an average edge weight for an analysis. All analyses that had the best analysis score for a sentence were designated 'gold'. Since it was possible for multiple analyses to have the same score, there could be multiple gold analyses for any one sentence. If all the analyses had the same score, this sentence could not be used as part of the training data. This method has the effect of selecting the parse(s) most like all the others, by some definitions the *centroid* of the parse forest. This has some relationship to the partial training method described by Clark and Curran (2006), where the most frequent dependencies where used to train a model for the C&C CCG parser. In that case, however, the dependencies were extracted only from analyses that matched the gold standard supertag sequence, rather than the whole parse forest.

Test Set	**Exact Match**		**F-score**	
	Edges	**Branching**	**Edges**	**Branching**
tc-006	17.48	21.35	0.815	0.822
jhpstg$_t$	15.27	17.53	0.766	0.780
catb	9.36	10.86	0.713	0.712

Table 5: Accuracy for each test set, measured both as percentage of sentences that exactly matched the gold standard, and f-score over elementary dependencies.

The second heuristic we tried is one often used as a baseline method: degree of right (or left) branching. In this instance, we calculated the degree of branching as the number of right branches in a parse divided by the number of left branches (and vice versa for Japanese, a predominantly left-branching language). In the same way as above, we designated all parses with the best branching score as 'gold'. Again, this is not fully discriminatory, and it was common to get multiple gold trees for a given sentence.

Table 5 shows the results for these two methods. All the results show an improvement over the baseline, with all but the F-score for the Edges method of *tc-006* being at a level of statistical significance.[5] The only statistically significant difference between the Edges and Branching methods is over the *jhpstg*$_t$ data set. While improvement over random is encouraging, the results were still uninspiring and so we moved on to slightly more complex methods, described in the next section.

5 Supertagging Experiments

The term *supertags* was first used by Bangalore and Joshi (1999) to describe fine-grained part of speech tags which include some structural or dependency information. In that original work, the supertags were LTAG (Schabes and Joshi, 1991) elementary trees, and they were used for the purpose of speeding up parsing by restricting the allowable leaf types. Subsequent work involving supertags has mostly focussed on this efficiency goal, but they can also be used to inform parse selection. Dalrymple (2006) and Blunsom (2007) both look at how discriminatory a tag sequence is in filtering a parse forest. This

[5] All statistical significance tests in these experiments use the computationally-intensive randomisation test described in Yeh (2000), with $p < 0.05$.

work has shown that tag sequences can be successfully used to restrict the set of parses produced, but generally are not discriminatory enough to distinguish a single best parse. Toutanova et al. (2002) present a similar exploration but also go on to include probabilities from a HMM model into the parse selection model as features. There has also been some work on using lexical probabilities for domain adaptation of a model (Hara et al., 2007; Rimell and Clark, 2008). In Dridan (2009), tag sequences from a supertagger are used together with other factors to re-rank the top 500 parses from the same parser and English grammar we use in this research, and achieve some improvement in the ranking where tagger accuracy is sufficiently high. We use a similar method, one level removed, in that we use the tag sequences to select the 'gold' parse(s) that are then used to train a model, as in the previous sections.

5.1 Gold Supertags

In order to test the viability of this method, we first experimented using gold standard tags, extracted from the gold standard parses. Supertags come in many forms, depending on both the grammar formalism and the implementation. For this work, we use HPSG lexical types (lextypes), the native word classes in the grammars. These lextypes encode part of speech and subcategorisation information, as well as some more idiosyncratic features of words, such as restrictions on preposition forms, mass/count distinctions and comparative versus superlative forms of adjectives. As a few examples from the English grammar, `v_np_le` represents a basic transitive verb, while `n_pp_c-of_le` represents a count noun that optionally takes a prepositional phrase complement headed by *of*. The full definition of a lextype consists of a many-featured AVM (attribute value matrix), but the type names have been deliberately chosen to represent the main features of each type. In the Dridan (2009) work, parse ranking showed some improvement when morphological information was added to the tags. Hence, we also look at more fine-grained tags constructed by concatenating appropriate morphological rules onto the lextypes, as in `v_np_le:past_verb_orule` (ie a simple transitive verb with past tense).

We used these tags by extracting the tag sequence from the leaf types of all the parses in the forest, marking as 'gold' any parse that had the same sequence as the gold standard parse and then training the models as before. Table 6 shows the results from parsing with models based on both the basic lextype and the lextype with morphology. The results are promising. They still fall well below training purely on gold standard data (at least for the in-domain sets), since the tag sequences are not fully discriminatory and hence noise can creep in, but accuracy is significantly better than the heuristic methods tried earlier. This suggested that, at least with a reasonably accurate tagger, this was a viable strategy for training a model. With no significant difference between the basic and +morph versions of the tag set, we decided to use the basic lextypes as tags, since a smaller tag set should be easier to tag with. However, we first had to train a tagger, without using any gold standard data.

Test Set	**Exact Match**		**F-score**	
	lextype	**+morph**	**lextype**	**+morph**
tc-006	40.49	41.37	0.903	0.903
jhpstg$_t$	32.93	32.93	0.862	0.858
catb	20.41	19.85	0.798	0.794

Table 6: Accuracy using gold tag sequence compatibility to select the 'gold' parse(s).

5.2 Unsupervised Supertagging

Research into unsupervised part-of-speech tagging with a tag dictionary (sometimes called weakly supervised POS tagging) has been going on for many years (cf Merialdo (1994), Brill (1995)), but generally using a fairly small tag set. The only work we know of on unsupervised tagging for the more complex *supertags* is from Baldridge (2008), and more recently, Ravi et al. (2010a). In this work, the constraining nature of the (CCG) grammar is used to mitigate the problem of having a much more ambiguous tag set. Our method has a similar underlying idea, but the implementation differs both in the way we extract the word-to-tag mappings, and also how we extract and use the information from the grammar to initialise the tagger model.

We chose to use a simple first-order Hidden Markov Model (HMM) tagger, using the implementation of Dekang Lin,[6] which re-estimates probabilities, given an initial model, using the Baum-Welch variant of the Expectation-Maximisation (EM) algorithm. One possibility for an initial model was to extract the word-to-lextype mappings from the grammar lexicon as Baldridge does, and make all starting probabilities uniform. However, our lexicon maps between lextypes and *lemmas*, rather than inflected word forms, which is what we'd be tagging. That is to say, from the lexicon we could learn that the lemma *walk* can be tagged as `v_pp*_dir_le`, but we could not directly extract the fact that therefore *walked* should also receive that tag.[7] For this reason, we decided it would be simplest to initialise our probability estimates using the output of the parser, feeding in only those tag sequences which are compatible with analyses in the parse forest for that item. This method takes advantage of the fact that, because the grammars are heavily constrained, the parse forest only contains viable tag sequences. Since parsing without a model is slow, we restricted the training set to those sentences shorter than a specific word length (12 for English and 15 for Japanese, since that was the less ambiguous grammar and hence faster).

Table 7 shows how much parsed data this gave us. From this parsed data we extracted tag-to-word and tag-to-tag frequency counts from all parses for all sentences, and used these frequencies to produce the emission and transition probabilities, respectively. The emission probabilities were taken directly from the normalised frequency counts, but for the transition probabilities we allow for all possible transitions, and add one to all counts before normalising. This model we call our *initial counts* model. The *EM trained* model is then produced by starting with this initial model and running the Baum-Welch algorithm using raw text sentences from the training corpus.

5.3 Supertagging-based parse selection models

We use both the *initial counts* and *EM trained* models to tag the training data from Table 1 and then compared this with the extracted tag sequences used in the gold tag experiment. Since we could no longer assume that our tag sequence would be present within the extracted tag sequences, we used the percentage of tokens from a parse whose lextype matched our tagged sequence as the parse score. Again, we marked as 'gold' any parse that had the best parse score for each sentence, and trained a new parse selection model.

Table 8 shows the results of parsing with these models. The results are impressive, significantly higher than all our previous unsupervised methods.

Interestingly, we note that there is no significant difference between the *initial count* and *EM trained* models for the English data. To explore why this might be so, we looked at the tagger accuracy for both models over the respective training data sets, shown in Table 9. The results are not conclusive. For both languages, the *EM trained* model is *less* accurate, though not significantly so for Japanese. However, this insignificant tagger accuracy decrease for Japanese produced a significant *increase* in parser accuracy, while a more pronounced tagger accuracy decrease had no significant effect on parser accuracy in English.

	Japanese	**English**
Parsed Sentences	9760	3770
Average Length	9.63	6.36
Average Parses	80.77	96.29
Raw Sentences	13500	9410
Raw Total Words	146053	151906

Table 7: Training data for the HMM tagger (both the parsed data from which the initial probabilities were derived, and the raw data which was used to estimated the *EM trained* models).

Test Set	**Exact Match**		**F-score**	
	Initial counts	**EM trained**	**Initial counts**	**EM trained**
tc-006	32.85	40.38	0.888	0.898
jhpstg$_t$	26.29	24.04	0.831	0.827
catb	14.61	14.42	0.782	0.783

Table 8: Accuracy using tag sequences from a HMM tagger to select the 'correct' parse(s). The *initial counts* model was based on using counts from a parse forest to approximate the emission and transition probabilities. The *EM trained* model used the Baum Welch algorithm to estimate the probabilities, starting from the initial counts state.

[6]Available from `http://webdocs.cs.ualberta.ca/~lindek/hmm.htm`

[7]Morphological processing occurs before lexicon lookup in the PET parser.

Language	Initial counts	EM trained
Japanese	84.4	83.3
English	71.7	64.6

Table 9: Tagger accuracy over the training data, using both the *initial counts* and the *EM trained* models.

There is much potential for further work in this direction, experimenting with more training data or more estimation iterations, or even looking at different estimators as suggested in Johnson (2007) and Ravi et al. (2010b). There is also the issue of whether tag accuracy is the best measure for indicating potential parse accuracy. The Japanese parsing results are already equivalent to those achieved using gold standard tags. It is possible that parsing accuracy is reasonably insensitive to tagger accuracy, but it is also possible that there is a better metric to look at, such as tag accuracy over frequently confused tags.

6 Discussion

The results of Table 8 show that, using no human annotated data, we can get exact match results that are almost half way between our random baseline and our gold-standard-trained upperbound. EDM F-scores of 90% and 83% over in-domain data compare well with dependency-based scores from other parsers, although a direct comparison is very difficult to do (Clark and Curran, 2007a; Miyao et al., 2007). It still remains to see whether this level of accuracy is good enough to be useful. The main aim of this work is to bootstrap the treebanking process for new grammars, but to conclusively show the efficacy of our methods in that situation requires a long-term experiment that we are now starting, based on the results we have here. Another possible use for these methods was alluded to in Section 2: producing a new model for a new domain.

Results at every stage have been much worse for the *catb* data set, compared to the other *jhpstg*$_t$ English data set. While sentence length plays some part, the major reason for this discrepancy was domain mismatch between the training and test data. One method that has been successfully used for domain adaption in parsing is self-training (McClosky et al., 2006). In this process, data from the new domain is parsed with the parser trained on the old domain, and then the top analyses of the parsed new domain data are added to the training data, and the parser is re-trained. This is generally considered a semi-supervised method, since the original parser is trained on gold standard data. In our case, we wanted to test whether parsing data from the new domain using our *unsupervised* parse selection model was accurate enough to still get an improvement using self-training for domain adaptation.

Source of 'Gold' Data	Exact Match	F-score
Random Selection	8.30	0.671
Supertags (*initial counts*)	14.61	0.782
Gold Standard	22.29	0.839
Self-training	15.92	0.791

Table 10: Accuracy results over the out-of-domain *catb* data set, using the *initial counts* unsupervised model to produce in-domain training data in a self-training set up. The previous results are shown for easy comparison.

It is not immediately clear what one might consider to be the 'domain' of the *catb* test set, since domain is generally very vaguely defined. In this case, there was a limited amount of text available from other essays by the same author.[8] While the topics of these essays vary, they all relate to the social side of technical communities, and so we used this to represent in-domain data for the *catb* test set. It is, however, a fairly small amount of data for self-training, being only around 1000 sentences. We added the results of parsing this data to the training set we used to create the *initial counts* model and again retrained and parsed. Table 10 shows the results. Previous results for the *catb* data set are given for comparison.

The results show that the completely unsupervised parse selection method produces a top parse that is at least accurate enough to be used in self-training, providing a cheap means of domain adaptation. In future work, we hope to explore this avenue of research further.

7 Conclusions and Further Work

Comparing Tables 8 and 4, we can see that for both English and Japanese, we are able to achieve parse selection accuracy well above our baseline of a ran-

[8]`http://www.catb.org/esr/writings/homesteading/`

dom selection-based model using only the information available in the grammar and raw text. This was in part because it is possible to extract a reasonable tagging model from uncorrected parse data, due to the constrained nature of these grammars. These models will hopefully allow grammar engineers to more easily build statistical models for new languages, using nothing more than their new grammar and raw text.

Since fully evaluating the potential for building models for new *languages* is a long-term ongoing experiment, we looked at a more short-term evaluation of our unsupervised parse selection methods: building models for new *domains*. A preliminary self-training experiment, using our *initial counts* tagger trained model as the starting point, showed promising results for domain adaptation.

There are plenty of directions for further work arising from these results. The issues surrounding what makes a good tagger for this purpose, and how can we best learn one without gold training data, would be one possibly fruitful avenue for further exploration. Another interesting slant would be to investigate domain effects of the tagger. Previous work has already found that training just a lexical model on a new domain can improve parsing results. Since the optimal tagger 'training' we saw here (for English) was merely to read off frequency counts for parsed data, it would be easy to retrain the tagger on different domains. Alternatively, it would be interesting so see how much difference it makes to train the tagger on one set of data, and use that to tag a model training set from a different domain. Other methods of incorporating the tagger output could also be investigated. Finally, a user study involving a grammar engineer working on a new language would be useful to validate the results we found here and confirm whether they are indeed helpful in bootstrapping a new grammar.

Acknowledgements

This research was supported by Australian Research Council grant no. DP0988242 and Microsoft Research Asia.

References

Jason Baldridge. 2008. Weakly supervised supertagging with grammar-informed initialization. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 57–64, Manchester, UK.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: an approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Emily M. Bender, Dan Flickinger, and Stephan Oepen. 2002. The grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, pages 8–14, Taipei, Taiwan.

Emily M. Bender. 2008. Evaluating a crosslinguistic grammar resource: A case study of Wambaya. In *Proceedings of the 46th Annual Meeting of the ACL*, pages 977–985, Columbus, USA.

Philip Blunsom. 2007. *Structured Classification for Multilingual Natural Language Processing*. Ph.D. thesis, Department of Computer Science and Software Engineering, the University of Melbourne.

Eric Brill. 1995. Unsupervised learning of disambiguation rules for part of speech tagging. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 1–13, Cambridge, USA.

Ted Briscoe and John Carroll. 2006. Evaluating the accuracy of an unlexicalised statistical parser on the PARC DepBank. In *Proceedings of the 44th Annual Meeting of the ACL and the 21st International Conference on Computational Linguistics*, pages 41–48, Sydney, Australia.

David Carter. 1997. The treebanker: a tool for supervised training of parsed corpora. In *Proceedings of a Workshop on Computational Environments for Grammar Development and Linguistic Engineering*, pages 9–15, Madrid, Spain.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 173–180, Ann Arbor, USA.

Stephen Clark and James R. Curran. 2006. Partial training for a lexicalized-grammar parser. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (NAACL)*, pages 144–151, New York City, USA.

Stephen Clark and James R. Curran. 2007a. Formalism-independent parser evaluation with CCG and DepBank. In *Proceedings of the 45th Annual Meeting of the ACL*, pages 248–255, Prague, Czech Republic.

Stephen Clark and James R. Curran. 2007b. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Ann Copestake, Dan Flickinger, Ivan A. Sag, and Carl Pollard. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, vol 3(no 4):pp 281–332.

Mary Dalrymple. 2006. How much can part-of-speech tagging help parsing? *Natural Language Engineering*, 12(4):373–389.

Rebecca Dridan. 2009. *Using lexical statistics to improve HPSG parsing*. Ph.D. thesis, Saarland University.

Dan Flickinger. 2002. On building a more efficient grammar by exploiting types. In Stephan Oepen, Dan Flickinger, Jun'ichi Tsujii, and Hans Uszkoreit, editors, *Collaborative Language Engineering*, pages 1–17. Stanford: CSLI Publications.

Tadayoshi Hara, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Evaluating impact of re-training a lexical disambiguation model on domain adaptation of an HPSG parser. In *Proceedings of the 10th International Conference on Parsing Technology (IWPT 2007)*, pages 11–22, Prague, Czech Republic.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396, September.

Mark Johnson. 2007. Why doesnt EM find good HMM POS-taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305, Prague, Czech Republic.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning*, Taipei, Taiwan.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 152–159, New York City, USA.

Bernard Merialdo. 1994. Tagging english text with a probabilistic model. *Computational Linguistics*, 20(2):155–171.

Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature forest models for probabilistic HPSG parsing. *Computational Linguistics*, 34(1):35–80.

Yusuke Miyao, Kenji Sagae, and Jun'ichi Tsujii. 2007. Towards framework-independent evaluation of deep linguistic parsers. In *Proceedings of the GEAF 2007 Workshop*, Palo Alto, California.

Yusuke Miyao, Rune Sætre, Kenji Sagae, Takuya Matsuzaki, and Jun'ichi Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *Proceedings of the 46th Annual Meeting of the ACL*, pages 46–54, Columbus, USA.

Stephan Oepen and John Carroll. 2000. Ambiguity packing in constraint-based parsing - practical results. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics*, pages 162–169, Seattle, USA.

Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christopher D. Manning. 2004. LinGO redwoods. a rich and dynamic treebank for HPSG. *Journal of Research in Language and Computation*, 2(4):575–596.

Stephan Oepen. 2001. [incr tsdb()] – competence and performance laboratory. User manual, Computational Linguistics, Saarland University, Saarbrücken, Germany.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, USA.

Sujith Ravi, Jason Baldridge, and Kevin Knight. 2010a. Minimized models and grammar-informed initialization for supertagging with highly ambiguous lexicons. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 495–503, Uppsala, Sweden.

Sujith Ravi, Ashish Vaswani, Kevin Knight, and David Chiang. 2010b. Fast, greedy model minimization for unsupervised tagging. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 940–948, Beijing, China.

Laura Rimell and Stephen Clark. 2008. Adapting a lexicalized-grammar parser to contrasting domains. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pages 475–484, Honolulu, USA.

Yves Schabes and Aravind K. Joshi. 1991. Parsing with lexicalized tree adjoining grammar. In Masaru Tomita, editor, *Current Issues in Parsing Technology*, chapter 3, pages 25–48. Kluwer.

Melanie Siegel and Emily M. Bender. 2002. Efficient deep processing of japanese. In *Proceedings of the 3rd Workshop on Asian Language Resources and International Standardization. Coling 2002 Post-Conference Workshop.*, Taipei, Taiwan.

Yasuhito Tanaka. 2001. Compilation of a multilingual parallel corpus. In *Proceedings of PACLING 2001*, pages 265–268, Kitakyushu, Japan.

Kristina Toutanova, Chistopher D. Manning, Stuart M. Shieber, Dan Flickinger, and Stephan Oepen. 2002. Parse disambiguation for a rich HPSG grammar. In *First Workshop on Treebanks and Linguistic Theories (TLT2002)*, pages 253–263.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, pages 947–953, Saarbrcken, Germany.

Yi Zhang, Stephan Oepen, and John Carroll. 2007. Efficiency in unification-based n-best parsing. In *Proceedings of the 10th international conference on parsing technologies (IWPT 2007)*, pages 48–59, Prague, Czech Republic.