Automatic Selection of High Quality Parses Created By a Fully Unsupervised Parser

Roi Reichart
ICNC
The Hebrew University
roiri@cs.huji.ac.il

Ari Rappoport
Institute of computer science
The Hebrew University
arir@cs.huji.ac.il

Abstract

The average results obtained by unsupervised statistical parsers have greatly improved in the last few years, but on many specific sentences they are of rather low quality. The output of such parsers is becoming valuable for various applications, and it is radically less expensive to create than manually annotated training data. Hence, automatic selection of high quality parses created by unsupervised parsers is an important problem.

In this paper we present PUPA, a *POS-based Unsupervised Parse Assessment* algorithm. The algorithm assesses the quality of a parse tree using POS sequence statistics collected from a batch of parsed sentences. We evaluate the algorithm by using an unsupervised POS tagger and an unsupervised parser, selecting high quality parsed sentences from English (WSJ) and German (NEGRA) corpora. We show that PUPA outperforms the leading previous parse assessment algorithm for supervised parsers, as well as a strong unsupervised baseline. Consequently, PUPA allows obtaining high quality parses without any human involvement.

1 Introduction

In unsupervised parsing an algorithm should uncover the syntactic structure of an input sentence without using any manually created structural training data. The last decade has seen significant progress in this field of research (Klein and Manning, 2002; Klein and Manning, 2004; Bod, 2006a; Bod, 2006b; Smith and Eisner, 2006; Seginer, 2007).

Many NLP systems use the output of supervised parsers (e.g., (Kwok et al., 2001) for QA, (Moldovan et al., 2003) for IE, (Punyakanok et al., 2008) for SRL, (Srikumar et al., 2008) for Textual Inference and (Avramidis and Koehn, 2008) for MT). To achieve good performance, these parsers should be trained on large amounts of manually created training data from a domain similar to that of the sentences they parse (Lease and Charniak, 2005; McClosky and Charniak, 2008). In the highly variable Web, where many of these systems are used, it is very difficult to create a representative corpus for manual annotation. The high cost of manual annotation of training data for supervised parsers imposes a significant burden on their usage.

A possible answer to this problem can be provided by high quality parses produced by unsupervised parsers that require little to no manual efforts for their training. These parses can be used either as input for applications, or as training material for modern supervised parsers whose output will in turn be used by applications.

Although unsupervised parser results improve, the quality of many of the parses they produce is still too low for such goals. For example, the Seginer (2007) parser achieves an F-score of 75.9% on the WSJ10 corpus and 59% on the NEGRA10 corpus, but the percentage of individual sentences with an F-score of 100% is 21.5% for WSJ10 and 11% for NEGRA10. When requirements are relaxed, only asking for an F-score higher than 85%, percentage is still low, 42% for WSJ10 and 15% for NEGRA10.

In this paper we address the task of a fully unsupervised assessment of high quality parses cre-

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 156–164,
Boulder, Colorado, June 2009.

ated by an unsupervised parser. The assessment should be unsupervised in order to avoid the problems mentioned above with manually trained supervised parsers. Assessing the quality of a learning algorithm's output and selecting high quality instances has been addressed for supervised algorithms (Caruana and Niculescu-Mizil, 2006) and specifically for supervised parsers (Yates et al., 2006; Reichart and Rappoport, 2007; Kawahara and Uchimoto, 2008; Ravi et al., 2008). Moreover, it has been shown to be valuable for supervised parser adaptation between domains (Sagae and Tsujii, 2007; Kawahara and Uchimoto, 2008; Chen et al., 2008). However, as far as we know the present paper is the first to address the task of unsupervised assessment of the quality of parses created by *unsupervised* parsers.

Our *POS-based Unsupervised Parse Assessment* (PUPA) algorithm uses statistics about POS tag sequences in a batch of parsed sentences[1]. The constituents in the batch are represented using the POS sequences of their yield and of the yields of neighboring constituents. Constituents whose representation is frequent in the output of the parser are considered to be of a high quality. A score for each range of constituent length is calculated, reflecting the robustness of statistics used for the creation of the constituents of that length. The final sentence score is a weighted average of the scores calculated for each constituent length. The score thus integrates the quality of short and long constituents into one score reflecting the quality of the whole parse tree.

PUPA provides a quality score for every sentence in a parsed sentences set. An NLP application can then decide if to use a parse or not, according to its own definition of a high quality parse. For example, it can select every sentence whose score is above some threshold, or the k top scored sentences. The selection strategy is application dependent and is beyond the scope of this paper.

The unsupervised parser we use is the Seginer (2007) incremental parser[2], which achieves state-of-the-art results without using manually created POS tags. The POS tags we use are induced by the unsupervised tagger of (Clark, 2003)[3]. Since both tagger and parser do not require any manual annotation, PUPA identifies high quality parses without any human involvement.

The incremental parser of (Seginer, 2007) does not give any prediction of its output quality, and extracting such a prediction from its internal data structures is not straightforward. Such a prediction can be given by supervised parsers in terms of the parse likelihood, but this was shown to be of medium quality (Reichart and Rappoport, 2007). While the algorithms of Yates et al. (2006), Kawahara and Uchimoto (2008) and Ravi et al. (2008) are supervised (Section 3), the ensemble based SEPA algorithm (Reichart and Rappoport, 2007) can be applied to unsupervised parsers in a way that preserves the unsupervised nature of the selection task.

To compare between two algorithms, we use each of them to assess the quality of the sentences in English and German corpora (WSJ and NEGRA)[4]. We show that for every sentence length (up to 20) the quality of the top scored k sentences according to PUPA is higher than the quality of SEPA's list (for every k). As in (Reichart and Rappoport, 2007), the quality of a set selected from the parser's output is evaluated using two measures: constituent F-score[5] and average sentence F-score.

Section 2 describes the PUPA algorithm, Section 3 discusses previous work, and Sections 4 and 5 present the evaluation setup and results.

2 The POS-based Unsupervised Parse Assessment (PUPA) Algorithm

In this section we detail our parse assessment algorithm. Its input consists of a set I of parsed sentences, which in our evaluation scenario are produced by an unsupervised parser. The algorithm assigns each parsed sentence a score reflecting its quality.

[1] The algorithm can be used with supervised POS taggers and parsers, but we focus here on the fully unsupervised scenario, which is novel and more useful. For completeness of analysis, we experimented with PUPA using a supervised POS tagger (see Section 5). Using PUPA with supervised parsers is left for future work.

[2] www.seggu.net/ccl.

[3] www.cs.rhul.ac.uk/home/alexc/RHUL/Downloads.html, the neyessenmorph model.

[4] This is in contrast to algorithms for selection from the results of supervised constituency parsers, which were evaluated only for English (Yates et al., 2006; Reichart and Rappoport, 2007; Ravi et al., 2008).

[5] This is the traditional parsing F-score.

The algorithm has three steps. First, the words in I are POS tagged (in our case, using the fully unsupervised POS induction algorithm of Clark (2003)). Second, POS statistics about the constituents in I are collected. Finally, a quality score is calculated for each parsed sentence in I using the POS statistics. In the following we detail the last two steps.

Collecting POS statistics. In its second step, the algorithm collects statistics about the constituents in the input set I. Recall that the *yield* of a constituent is the set of words covered by it. The PUPA *constituent representation (PCR)* consists of three features: (1) the ordered POS tag sequence of the constituent's yield, (2) the constituents' right context, and (3) the constituents' left context.

We define *context* to be the *leftmost and rightmost* POS tags in the yield of the *neighbor* of the constituent (if there is only one POS tag in the neighbor's yield, this POS tag is the context). For the right and left contexts we consider the right and left neighbors respectively. A constituent C_1 is the right neighbor of a constituent C_2 if C_1 is the highest level constituent such that the first word in the yield of C_1 comes immediately after the last word in the yield of C_2. A constituent C_1 is the left neighbor of a constituent C_2 if C_1 is the highest level constituent such that the first word in the yield of C_2 comes immediately after the last word in the yield of C_1.

Figure 1 shows an example, an unlabeled tree for the sentence 'I will give you the ball'. The tree has 6 constituents (C0-C5). C3 and C4 have both right and left neighbors. For C3, the POS sequence of its yield is POS2, POS3 , the left neighbor is C1 and thus the left context is POS1, and the right neighbor is C4 and thus the right context is POS4. Note that the left and right neighbors of C3 have only one POS tag in their yield and therefore this POS tag is the context. For C4 the yield is POS4, the left neighbor is C3 (and thus the left context is POS2,POS3), and the right neighbor is C5 (and thus the right context is POS5,POS6). C1, whose yield is POS1, has only a right neighbor, C2, and thus its right context is POS2,POS6 and its left context is NULL. C2 and C5 (whose yields are POS2, POS3, POS4, POS5, POS6 for C2 and POS5, POS6 for C5) have only a left neighbor. For C2, this is C1 (and the context is POS1) while for C5 this is C4 (with the context POS4).

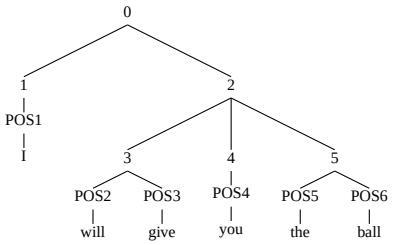


Figure 1: An example parse tree for contexts and neighbors (see text).

The right context of both constituents is NULL. As all sentence level constituents, C0 has no neighbors, and thus both its left and right contexts are NULL.

We have also explored other representations of left and right contexts based on the POS tags of their yields. In these, we represented the left/right neighbor using only the leftmost/rightmost POS tags of its yield or other subsets of the yield's POS tags. These variations produced lower quality results than the main variant above in our experiments, which were for English and German. Exploring the suitability of our representation for other languages is left for future research.

Score computation. The third and last step of the algorithm is a second pass over I for computing a quality score for each parse tree.

Short constituents tend to be more frequent than long ones. In order not to distort our score due to parsing errors in short constituents, PUPA computes the grade using a division into lengths, in three steps. First, constituents are assigned to bins according to their length, each bin containing the constituents of a certain range of lengths. Denote this range by W (for width), and the number of bins by $N(W)$. For example, in our experiments the longest possible constituent is of length 20, so we can take $W = 5$, resulting in $N(W) = 4$: bin 1 for constituents of length 1-5, bin 2 for constituents of length 6-10, and so on for bins 3, 4.

The score of bin_i is given by

$$(1)\ BinScore(bin_i) = \sum_{t=2}^{t=X}(X - t + 2) \cdot \frac{|C_t^i|}{|C^i|}$$

Where X is the maximal number of occurrences of constituents in the bin that we consider as important for the score (see below for its selection), $|C_t^i|$ is the number of constituents in bin i occurring at

least t times in the batch of parsed sentences, and $|C^i|$ is the number of constituents in bin i. In words, the score is a weighted average: the fraction of the constituents in the bin occuring at least 2 times (with weight X), plus the fraction of the constituents in the bin occuring at least 3 times (with weight $X-1$), etc, until the fraction of the constituents in the bin occuring at least X times (with weight 2).

A score for the division into N bins is given by

(2) $Score(N(W)) = \frac{\sum_{i=1}^{N(W)} BinScore(bin_i)}{Z \cdot M}$

Where Z is the maximum bin score (according to (1)) and M is the number of bins containing at least one constituent. If, for example, $N(W) = 4$ and there is no constituent whose length is between 11 and 15 then bin number 3 is empty. If every other bin contains at least one constituent, $M = 3$.

To get a final score for the parse tree of sentence S that is independent of a specific bin division, we sum the scores of the various bin division:

(3) $PupaScore(S) = \frac{\sum_{W=1}^{W=Y} Score(N(W))}{Y}$

where Y is the length of S (which is also its maximum bin width). $PupaScore$ thus takes values in the $[0, 1]$ range.

In equation (1), if, for example, $X = 20$ then the weight of the fraction of the bin's constituents occurring at least 2 times is 20 while the weight of the fraction of the constituents occurring at least 10 times is 12 and of the fraction of constituents occurring at least 20 times is 2. We consider the number of times a constituent appears in a batch to be an indication of its correctness. The difference between 3 and 2 occurrences is therefore more indicative than the difference between 20 and 19 occurrences. More generally, the more times a constituent occurs, the less indicative any additional appearance is.

In equation (2) we give all bins the same weight. Short constituents are more frequent and are generally more likely to be correct. However, the correctness of long constituents is an indication that the parser has a correct interpretation of the tree structure and that it is likely to create a high quality tree. The usage of equal bin weights was done to balance the tendency of parse trees to have more short constituents.

Parameters. PUPA has two parameters: X, the maximal number of occurrences considered in equation (1), and P, the number of POS tags induced by the unsupervised POS tagger. In the following we present the unsupervised technique we used to tune these parameters.

Figure 2 shows $nc(t)$, the number of constituents appearing at least t times in WSJ20 (left) and NEGRA20 (right). For both corpora, the pattern is shown when using 5 POS tags ($P = 5$, solid line) and 50 POS tags (P = 50, dashed line). The distribution obeys Zipf's law: many constituents appear a small number of times while a few constituents appear a large number of times. We denote the t value where the slope changes from steep to moderate by t_{elbow}. Practically, we approximate the 'real' elbow value and define t_{elbow} to be the smallest t for which $nc(t+1) - nc(t) = 1$. When $P = 5$, t_{elbow} is 32 for WSJ and 19 for NEGRA. When P = 50, t_{elbow} is 15 for WSJ and 9 for NEGRA.

The number of constituents appearing more than t_{elbow} times is considerably smaller than the number of constituents appearing t_{elbow} times or less. Therefore, the fact that a constituent appears $t_{elbow} + S$ times (for a positive integer S) is not a better indication of its quality than the fact that it appears t_{elbow} times. We thus select X to be t_{elbow}.

The graphs also demonstrate that for both corpora, t_{elbow} for $P = 50$ is smaller than t_{elbow} for $P = 5$. Generally, t_{elbow} is a monotonically decreasing function of P. Lower t_{elbow} values imply that PUPA would be less distinctive between constituents quality (see equation (1); recall that $X = t_{elbow}$). We thus want to select the P value that maximizes t_{elbow}. We therefore minimize P. t_{elbow} values for $P \in \{3, \ldots, 10\}$ are very similar. Indeed, PUPA achieves its best performance for $P \in \{3, \ldots, 10\}$ and it is insensitive to the selection of P in this range. In Section 5 we report results with $P = 5$.

3 Related Work

Unsupervised parsing has been explored for several decades (see (Klein, 2005) for a recent review). Recently, unsupervised parsing algorithms have for the first time outperformed the right branching heuristic baseline for English. These include CCM (Klein and Manning, 2002), the DMV and DMV+CCM models (Klein and Manning, 2004), (U)DOP based mod-

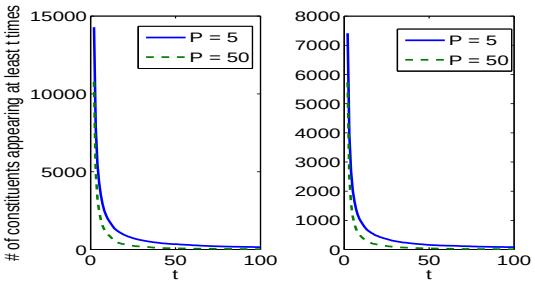


Figure 2: Number of constituents appearing at least t times ($nc(t)$) as a function of t. Shown are WSJ (left) and NEGRA (right), where constituents are represented according to PUPA's PCR with 5 POS tags ($P = 5$, solid line) or 50 POS tags ($P = 50$, dashed line).

els (Bod, 2006a; Bod, 2006b), an exemplar based approach (Dennis, 2005), guiding EM using contrastive estimation (Smith and Eisner, 2006), and the incremental parser of Seginer (2007) that we use in this work. To obtain good results, manually created POS tags are used as input in all of these algorithms except Seginer's, which uses plain text.

Quality assessment of a learning algorithm's output and selection of high quality instances have been addressed for supervised algorithms (see (Caruana and Niculescu-Mizil, 2006) for a survey) and specifically for supervised constituency parsers (Yates et al., 2006; Reichart and Rappoport, 2007; Ravi et al., 2008). For dependency parsing in a corpus adaptation scenario, (Kawahara and Uchimoto, 2008) built a binary classifier that classifies each parse in the parser's output as reliable or not. To do that, they selected 2500 sentences from the parser's output, compared them to their manually created gold standard, and used accurate (inaccurate) parses as positive (negative) examples for the classifier. Their approach is supervised and the features used by the classifier are dependency motivated .

As far as we know, the present paper is the first to address the task of selecting high quality parses from the output of unsupervised parsers. The algorithms of Yates et al. (2006), Kawahara and Uchimoto (2008) and Ravi et al. (2008) are supervised, performing semantic analysis of the parse tree and gold standard-based calssification, respectively. However, the SEPA algorithm of Reichart and Rappoport (2007), an algorithm for supervised constituency parsers, can be applied to unsupervised parsers in a way that preserves the unsupervised nature of the selection task. In Section 5 we provide a detailed comparison between PUPA and SEPA showing the first to be superior. Below is a brief description of the SEPA algorithm.

The input of the SEPA algorithm consists of a parsing algorithm A, a training set, and a test set (which in the unsupervised case might be the same set). The algorithm provides, for each of the test set's parses generated by A when trained on the full training set, a grade assessing the parse quality, on a continuous scale between 0 to 100. The quality grade is calculated in the following way: N random samples of size S are sampled from the training data and used for training the parsing algorithm A. In that way N committee members are created. Then, each of the test sentences is parsed by each of the N committee members and an agreement score ranging from 0 to 100 between the committee members is calculated. All unsupervised parsers mentioned above (including the Seginer parser), have a training phase where parameter values are estimated from unlabeled data. SEPA can thus be applied to the unsupervised case.

Automatic selection of high quality parses has been shown to improve parser adaptation. Sagae and Tsujii (2007) and Kawahara and Uchimoto (2008) applied a self-training protocol to a parser adaptation scenario but used only high quality parses to retrain the parser. In the first work, high quality parses were selected using an ensemble method, while in the second a binary classifier was used (see above). The first system achieved the highest score in the CoNLL 2007 shared task on domain adaptation of dependency parsers, and the second system improved over the basic self-training protocol. Chen et al. (2008) parsed target domain sentences and used short dependencies information, which is often accurate, to adapt a dependency parser to the Chinese language.

Automatic quality assessment has been extensively explored for machine translation (Ueffing and Ney, 2007) and speech recognition (Koo et al., 2001). Other NLP tasks where it has been explored include semi-supervised relation extraction (Rosenfeld and Feldman, 2007), IE (Culotta and McCallum, 2004), QA (Chu-Carroll et al., 2003), and dialog systems (Lin and Weng, 2008).

The idea of representing a constituent by its yield

and (a different definition of) context is used by the CCM unsupervised parsing model (Klein and Manning, 2002). As far as we know the current work is the first to use unsupervised POS tags for the selection of high quality parses.

4 Evaluation Setup

We experiment with sentences of up to 20 words from the English WSJ Penn Treebank (WSJ20, 25236 sentences, 225126 constituents) and the German NEGRA corpus (Brants, 1997) (NEGRA20, 15610 sentences, 108540 constiteunts), both containing newspaper texts.

The unsupervised parsers of the kind addressed in this paper output unlabeled parse trees. To evaluate the quality of a single parse tree with respect to another, we use the unlabeled F-score ($UF = \frac{2 \cdot UR \cdot UP}{UR+UP}$), where UR and UP are unlabeled recall and unlabeled precision respectively.

Following the unsupervised parsing literature, multiple brackets and brackets covering a single word are not counted, but the sentence level bracket is. We exclude punctuation and null elements according to the scheme of (Klein, 2005).

The performance of unsupervised parsers markedly degrades as sentence length increases. For example, the Average sentence F–score for WSJ sentences of length 10 is 71.4% compared to 58.5 for sentences of length 20 (the numbers for NEGRA are 48.2% and 36.9%). We therefore evaluate PUPA (and the baseline) for sentences of a given length. We do this for every sentence of length 2-20 in WSJ20 and NEGRA20.

For every sentence length L, we use PUPA and the baseline algorithm (SEPA) to give a quality score to each of the sentences of that length in the experimental corpus. We then compare the quality of the top k parsed sentences according to each algorithm. We do this for every k from 1 to the number of sentences of length L.

Following Reichart and Rappoport (2007), we use two measures to evaluate the quality of a *set* of parses: the *constituent F-score* (the traditional F-score used in the parsing literature), and the *average F-score* of the parses in the set. In the first measure we treat the whole set as a bag of constituents. Each constituent is marked as correct (if it appears in the gold standard parses of the set) or erroneous (if it does not). Then, recall, precision and F-score are calculated over these constituents. In the second measure, the constituent F-score of each of the parses in the set is computed, and then results are averaged.

There are applications that use individual constituents from the output of a parser while others need the whole parse tree. For example, if the selected set is used for training supervised parsers such as the Collins parser (Collins, 1999), which collects constituent statistics, the constituent F-score of the selected set is the important measure. In applications such as the syntax based machine translation model of (Yamada and Knight, 2001), a low quality tree might lead to errorenous translation of the sentence. For such applications the average F-score is more indicative. These measures thus represent complementary aspects of a set quality and we consider both of them.

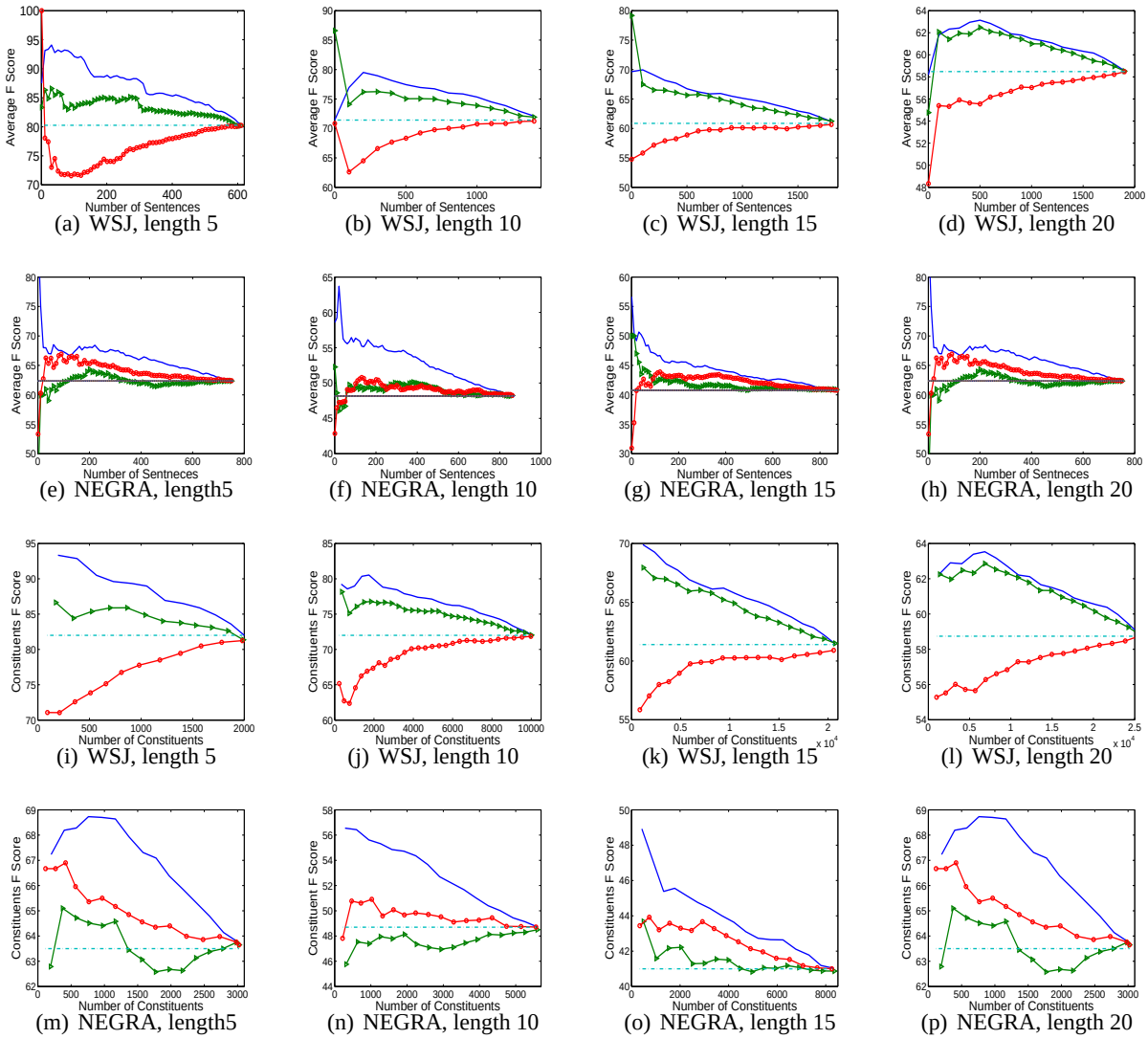
The parser we use is the incremental parser of (Seginer, 2007), POS tags are induced using the unsupervised POS tagger of ((Clark, 2003), neyessenmorph model). In each experiment, the tagger was trained with the raw sentences of the experiment corpus, and then the corpus words were POS tagged.

The output of the unsupervised POS tagger depends on a random initialization. We ran the tagger 5 times, each time with a different random initialization, and then ran PUPA with its output. The results we report for PUPA are the average over these 5 runs. Random selection results (given for reference) were also averages over 5 samples.

PUPA 's parameter estimation is completely unsupervised (see Section 2). No development data was used to tune its parameters.

A 200 sentences development set from each corpus was used for calibrating the parameters of the SEPA algorithm. Based on the analysis of SEPA performance with different assignments of its parameters given by Reichart and Rappoport (2007) (see Section 3), we ran the SEPA algorithm with sample size (SEPA parameter S) of 30% and 80%, and with 2 – 10 committee members (N)[6]. The optimal parameters were $N = 10$,$S = 80$ for WSJ20, and

[6]We tried higher N values but observed no improvements in SEPA's performance.



(a) WSJ, length 5 (b) WSJ, length 10 (c) WSJ, length 15 (d) WSJ, length 20
(e) NEGRA, length5 (f) NEGRA, length 10 (g) NEGRA, length 15 (h) NEGRA, length 20
(i) WSJ, length 5 (j) WSJ, length 10 (k) WSJ, length 15 (l) WSJ, length 20
(m) NEGRA, length5 (n) NEGRA, length 10 (o) NEGRA, length 15 (p) NEGRA, length 20

Figure 3: **In all graphs:** PUPA: solid line. SEPA: line with triangles. MC: line with circles. Random selection is presented for reference as a dotted line. **Top two rows:** Average F-score for PUPA, SEPA and MC for sentences from WSJ (top row) and NEGRA (bottom row). **Bottom two rows:** Constituents F-score for PUPA, SEPA and MC for sentences from WSJ (top row) and NEGRA (bottom row). Results are presented for sentence lengths of 5,10,15 and 20 (patterns for other sentence lengths between 2 and 20 are very similar). PUPA is superior in all cases. The graphs for PUPA and SEPA show a downward trend because parsed sentences were sorted according to score, which correlates positively with F-score (unlike MC). The graphs converge because on the extreme right all test sentences were selected.

$N = 10$, $S = 30$ for NEGRA20.

We also compare PUPA to a baseline selecting the sentences with the lowest number of constituents. Since the number of constituents is an indication of the complexity of the syntactic structure of a sentence, it is reasonable to assume that selecting the sentences with the lowest number of constituents is a good selection strategy. We denote this baseline by MC (for minimum constituents).

The incremental parser does not give any prediction of its output quality as supervised generative parsers do. We are thus not able to compare to such a score.

5 Results

Figure 3 shows Average F-score and Constituents F-score results for PUPA SEPA and MC, for sentences

of lengths 5,10,15 and 20 in WSJ20 and NEGRA20. The top two rows are for Average F-score (top row: WSJ, bottom row: NEGRA), while the bottom two rows are for Constituents F-score (top row: WSJ, bottom row: NEGRA).

PUPA and SEPA are both better than random selection for both corpora for every sentence length. The MC baseline is better than random selection only for NEGRA (in which case it outperforms SEPA). For WSJ, however, random selection is a better strategy than MC.

It is clear from the graphs that PUPA outperforms SEPA and MC in all experimental conditions. We observed very similar patterns in all other sentence lengths in WSJ20 and NEGRA20 for both Average F-score and Constituent F-score. In other words, for every sentence length in both corpora, PUPA outperforms SEPA and MC in terms of both measures. we present our results per sentence length to deprive the possibility that PUPA is useful only for short sentences or that it prefers sentences whose syntactic structure is not complex (i.e. with a small number of constituents, like MC).

Table 1 shows that the same pattern of results holds when evaluating on the whole corpus (WSJ20 or NEGRA20) without any sentence length restriction.

Note that while PUPA is a fully unsupervised algorithm, SEPA requires a few hundreds of sentences for its parameters tuning.

The main result of this paper is for sentences whose length is up to 20 words (note that most unsupervised parser literature reports numbers for sentences up to length 10). We have also ran the experiments for the remaining length range, 20-40. For NEGRA, PUPA is superior over MC up to length 36, and both are much better than SEPA. For WSJ, PUPA and SEPA both outperform MC, but SEPA is a bit better than PUPA. When evaluating on the whole corpus (i.e. without sentence length restriction, like in Table 1) PUPA is superior over both SEPA and MC for WSJ40 and NEGRA40.

For completeness of analysis we also experimented in the condition where PUPA uses gold standard POS tags as input. The number of these tags is 35 for WSJ and 57 for NEGRA. Interestingly, PUPA achieves in this condition the same performance as when using the same number of POS tags induced by an unsupervised POS tagger. Since PUPA's performance for a smaller number of POS tags is better (see our parameter tuning discussion above), the bottom line is that PUPA pefers using induced POS tags over gold POS tags.

	5%	10%	20%	30%	40%	50%
WSJ20						
PUPA	**82.75**	**79.34**	**75.77**	**73.46**	**71.68**	**70.3**
SEPA	78.68	75.7	72.64	70.72	69.54	68.58
MC	76.75	74.6	72.1	70.35	68.97	67.77
NEGRA20						
PUPA	**70.66**	**67.06**	**61.89**	**58.75**	**56.6**	**54.73**
SEPA	66.19	62.75	59.41	57.16	55.23	53.7
MC	69.41	65.79	60.87	58.08	55.9	54.36

Table 1: Average F–score for the top k% of constituents selected from WSJ20 (up) and NEGRA20 (down). No sentence length restriction is imposed. Results presented for PUPA , SEPA and MC. Average F–score of random selection is 66.55 (WSJ20) and 47.05 (NEGRA20). PUPA is superior over all methods.

6 Conclusions

We introduced PUPA, an algorithm for unsupervised parse assessment that utilizes POS sequence statistics. PUPA is a fully unsupervised algorithm whose parameters can be tuned in an unsupervised manner. Experimenting with the Seginer unsupervised parser and Clark's unsupervised POS tagger on English and German corpora, PUPA was shown to outperform both the leading parse assessment algorithm for supervised parsers (SEPA, even when its parameters are tuned on manually annotated development data) and a strong baseline (MC).

Using PUPA, we extracted high quality parses from the output of a parser which requires raw text as input, using POS tags induced by an unsupervised tagger. PUPA thus provides a way of obtaining high quality parses without any human involvement.

For future work, we intend to use parses selected by PUPA from the output of unsupervised parsers as training data for supervised parsers, and in NLP applications that use parse trees. A challenge for the first direction is the fact that state of the art supervised parsers require labeled parse trees, while modern unsupervised parsers create unlabeled trees. Combining PUPA with algorithms for labeled parse trees induction (Haghighi and Klein, 2006; Reichart and Rappoport, 2008) is a one direction to overcome this challenge. We also intend to use PUPA to assess the quality of parses created by supervised parsers.

References

Eleftherios Avramidis and Philipp Koehn, 2008. Enriching Morphologically Poor Languages for Statistical Machine Translation. *ACL '08*.

Rens Bod, 2006a. An All-Subtrees Approach to Unsupervised Parsing. *ACL '06*.

Rens Bod, 2006b. Unsupervised Parsing with U-DOP. *CoNLL X*.

Thorsten Brants, 1997. The NEGRA Export Format. *CLAUS Report, Saarland University*.

Rich Caruana and Alexandru Niculescu-Mizil, 2006. An Empirical Comparison of Supervised Learning Algorithms. *ICML '06*.

Jennifer Chu-Carroll, Krzysztof Czuba, John Prager and Abraham Ittycheriah, 2003. In Question Answering, Two Heads Are Better Than One. *HLT-NAACL '03*.

Wenliang Chen, Youzheng Wu and Hitoshi Isahara, 2008. Learning Reliable Information for Dependency Parsing Adaptation. *Coling '08*.

Alexander Clark, 2003. Combining Distributional and Morphological Information for Part of Speech Induction. *EACL '03*.

Michael Collins, 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Aron Culotta and Andrew McCallum, 2004. Confidence Estimation for Information Extraction. *HLT-NAACL '04*.

Simon Dennis, 2005. An Exemplar-based Approach to Unsupervised Parsing. *Proceedings of the 27th Conference of the Cognitive Science Society*.

Aria Haghighi and Dan Klein, 2006. Prototype-driven Grammar Induction. *ACL '06*.

Daisuke Kawahara and Kiyotaka Uchimoto 2008. Learning Reliability of Parses for Domain Adaptation of Dependency Parsing. *IJCNLP '08*.

Dan Klein and Christopher Manning, 2002. A Generative Constituent-Context Model for Improved Grammar Induction. *ACL '02*.

Dan Klein and Christopher Manning, 2004. Corpus-based Induction of Syntactic Structure: Models of Dependency and Constituency. *ACL '04*.

Dan Klein, 2005. *The Unsupervised Learning of Natural Language Structure.* Ph.D. thesis, Stanford University.

Myoung–Wan Koo, Chin-Hui Lee and Biing–Hwang Juang 2001. Speech Recognition and Utterance Verification Based on a Generalized Confidence Score. *IEEE Transactions on Speech and Audio Processing*, 9(8):821–832.

Cody Kwok, Oren Etzioni and Daniel S. Weld, 2001. Scaling Question Answering to the Web. *WWW '01*.

Matthew Lease and Eugene Charniak, 2005. Towards a Syntactic Account of Punctuation. *IJCNLP '05*.

Feng Lin and Fuliang Weng, 2008. Computing Confidence Scores for All Sub Parse Trees. *ACL '08, short paper*.

David McClosky and Eugene Charniak, 2008. Self-Training for Biomedical Parsing. *ACL '08, short paper*.

Dan Moldovan, Christine Clark, Sanda Harabagiu and Steve Maiorano, 2003. Cogex: A Logic Prover for Question Answering. *HLT-NAACL '03*.

Vasin Punyakanok and Dan Roth and Wen-tau Yih, 2008. The Importance of Syntactic Parsing and Inference in Semantic Role Labeling. *Computational Linguistics*, 34(2):257-287.

Sujith Ravi, Kevin Knight and Radu Soricut, 2008. Automatic Prediction of Parser Accuracy. *EMNLP '08*.

Roi Reichart and Ari Rappoport, 2007. An Ensemble Method for Selection of High Quality Parses. *ACL '07*.

Roi Reichart and Ari Rappoport, 2008. Unsupervised Induction of Labeled Parse Trees by Clustering with Syntactic Features. *COLING '08*.

Benjamin Rosenfeld and Ronen Feldman, 2007. Using Corpus Statistics on Entities to Improve Semi–Supervised Relation Extraction From The WEB. *ACL '07*.

Kenji Sagae and Junichi Tsujii, 2007. Dependency Parsing and Domain Adaptation with LR Models and Parser Ensemble. *EMNLP-CoNLL '07*.

Yoav Seginer, 2007. Fast Unsupervised Incremental Parsing. *ACL '07*.

Vivek Srikumar, Roi Reichart, Mark Sammons, Ari Rappoport and Dan Roth, 2008. Extraction of Entailed Semantic Relations Through Syntax-based Comma Resolution. *ACL '08*.

Noah A. Smith and Jason Eisner, 2006. Annealing Structural Bias in Multilingual Weighted Grammar Induction. *ACL '06*.

Nicola Ueffing and Hermann Ney, 2007. Word-Level Confidence Estimation for Machine Translation. *Computational Linguistics*, 33(1):9–40.

Kenji Yamada and Kevin Knight, 2001. A Syntax-Based Statistical Translation Model. *ACL '01*.

Alexander Yates, Stefan Schoenmackers and Oren Etzioni, 2006. Detecting Parser Errors Using Web-based Semantic Filters . *EMNLP '06*.