Semantic Role Labeling Improves Incremental Parsing

Ioannis Konstas and Frank Keller
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
{ikonstas,keller}@inf.ed.ac.uk

Abstract

Incremental parsing is the task of assigning a syntactic structure to an input sentence as it unfolds word by word. Incremental parsing is more difficult than full-sentence parsing, as incomplete input increases ambiguity. Intuitively, an incremental parser that has access to semantic information should be able to reduce ambiguity by ruling out semantically implausible analyses, even for incomplete input. In this paper, we test this hypothesis by combining an incremental TAG parser with an incremental semantic role labeler in a discriminative framework. We show a substantial improvement in parsing performance compared to the baseline parser, both in full-sentence F-score and in incremental F-score.

1 Introduction

When humans listen to speech, the input becomes available gradually as the speech signal unfolds. Reading happens in a similarly gradual manner when the eyes scan a text. There is good evidence that the human language processor is adapted to this and works incrementally, i.e., computes an interpretation for an incoming sentence on a word-by-word basis (Tanenhaus et al., 1995; Altmann and Kamide, 1999). Also language processing systems often deal with speech as it is spoken, or text as it is typed. A dialogue system should start interpreting a sentence while it is spoken, and an information retrieval system should start retrieving results while the user is typing.

Incremental processing is therefore essential both for realistic models of human language processing and for NLP applications that react to user input in real time. In response to this, a number of incremental parsers have been developed, which use context-free grammar (Roark, 2001; Schuler et al., 2010), dependency grammar (Chelba and Jelinek, 2000; Nivre, 2007; Huang and Sagae, 2010), or tree-substitution grammars (Sangati and Keller, 2013). Typical applications of incremental parsers include speech recognition (Chelba and Jelinek, 2000; Roark, 2001; Xu et al., 2002), machine translation (Schwartz et al., 2011; Tan et al., 2011), reading time modeling (Demberg and Keller, 2008), or dialogue systems (Stoness et al., 2004).

Incremental parsing, however, is considerably harder than full-sentence parsing: when processing the n-th word in a sentence, a_n, the parser only has access to the left context (words $a_1 \ldots a_{n-1}$); the right context (words $a_{n+1} \ldots a_N$) is not known yet. This can lead to **local ambiguity,** i.e., produce additional syntactic analyses that are valid for the sentence prefix, but become invalid as the right context is processed. As an example consider the sentence prefix in (1):

(1) The athlete realized her goals ...
  a. at the competition
  b. were out of reach

The prefix could continue as in (1-a), i.e., as a main clause structure. Or the next words could be as in (1-b), in which case *her goals* is part of a subordinate clause.

Intuitively, an incremental parser that has access to semantic information would be able to decide which of these two analyses is likely to be correct, even without knowing the rest of the sentence. If the NP *her goals* is a likely ARG1 of *realized* the parser should prefer the main clause structure. On the other hand, if the NP is a likely ARG0 of an (as yet unseen) embedded verb, then the parser should go for the subordinate clause structure. This is illustrated in Figure 2. Note that the preference can easily be reversed: if the prefix was *the athlete realized her shoes*, then *her shoes* is very likely to be an ARG0 rather than an ARG1.

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 1191–1201, Beijing, China, July 26-31, 2015.

The basis of this paper is the hypothesis that semantic information can aid incremental parsing. To test this hypothesis, we combine an incremental TAG parser with an **incremental semantic role labeling** (iSRL) system. The iSRL system takes prefix trees and computes their most likely semantic role assignments. We show that these role assignments can be used to re-rank the output of the incremental parser, leading to substantial improvements in parsing performance compared to the baseline parser, both in full-sentence F-score and in incremental F-score.

2 Incremental Semantic Role Labeling

The current work builds on an existing incremental parser, the Psycholinguistically Motivated Tree Adjoining Grammar (PLTAG) parser of Demberg et al. (2013). The distinguishing feature of this parser is that it builds fully connected structures (no words are left unattached during incremental parsing); this requires it to make predictions about the right context, which are verified as more of the input becomes available. Konstas et al. (2014) show that semantic information can be attached to PLTAG structures, making it possible to assign semantic roles incrementally. In the present paper, we use these semantic roles to re-rank the output of the PLTAG parser.

2.1 Psycholinguistically Motivated TAG

PLTAG extends standard TAG (Joshi and Schabes, 1992) in order to enable incremental parsing. Standard TAG assumes a lexicon of **elementary trees,** each of which contains at least one lexical item as an anchor and at most one leaf node as a **foot node,** marked with $A*$. All other leaves are marked with $A\downarrow$ and are called **substitution nodes.** To derive a TAG parse for a sentence, we start with the elementary tree of the head of the sentence and integrate the elementary trees of the other lexical items of the sentence using two operations: **adjunction** at an internal node and **substitution** at a substitution node (the node at which the operation applies is the **integration point**). Standard TAG derivations are not guaranteed to be incremental, as adjunction can happen anywhere in a sentence, possibly violating left-to-right processing order. PLTAG addresses this limitation by introducing **prediction trees,** elementary trees without a lexical anchor. These are used to predict syntactic structure anchored by words that appears later in an incremental derivation. This ensures

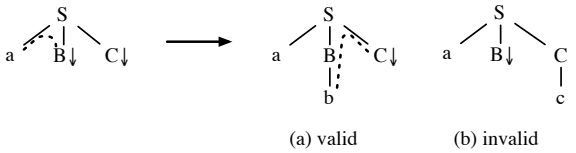

Figure 1: The current fringe (dashed line) indicates where valid substitutions can occur. Other substitutions result in an invalid prefix tree.

that fully connected prefix trees can be built for every prefix of the input.

In order to efficiently parse PLTAG, Demberg et al. (2013) introduce the concept of **fringes.** Fringes capture the fact that in an incremental derivation, a prefix tree can only be combined with an elementary tree at a limited set of nodes. For instance, the prefix tree in Figure 1 has two substitution nodes, for B and C. However, only substitution into B leads to a valid new prefix tree; if we substitute into C, we obtain the tree in Figure 1b, which is not a valid prefix tree (i.e., it represents a non-incremental derivation).

2.2 Incremental Role Propagation

The output of a semantic role labeler is a set of semantic dependency triples $\langle l, r, p \rangle$, where l is a semantic role label (e.g., ARG0, ARG1, ARGM in Propbank), and r and p are the words (argument and predicate) to which the role applies. An *incremental* semantic role labeler assigns semantic dependency triples to a prefix of the input sentence. Note that not every word is an argument to a predicate, therefore the set of triples will not necessarily change at every input word. Also, triples can be incomplete, as either the predicate or the argument may not have been observed yet.

Konstas et al. (2014) propose an iSRL system based on a PLTAG parser with a semantically augmented lexicon. They parse an input sentence incrementally, applying their incremental role propagation algorithm (IRPA) to the resulting prefix trees. This creates new semantic triples (or updates existing, incomplete ones) whenever an elementary or prediction tree that carries semantic role information is attached to the prefix tree. As soon as a triple is completed a two-stage classification process is applied, that first identifies whether the predicate/argument pair is a good candidate, and then disambiguates the role label (often multiple roles are possible for a lexical entry). Figure 2 shows the incremental role assignment for the two readings of the prefix *the athlete realized her goals*

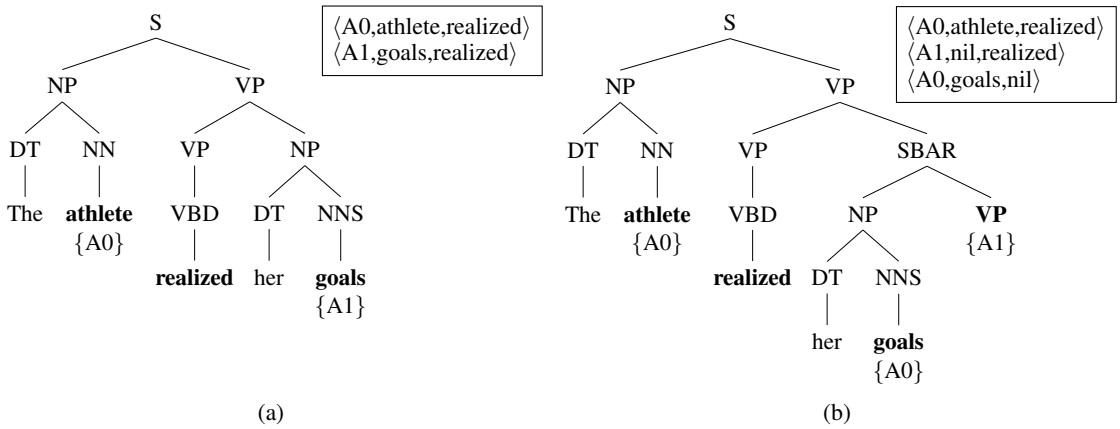

Figure 2: Incremental Role Propagation Algorithm application for two different prefix trees of the sentence prefix *the athlete realized her goals*. In (a) the parser builds a main clause, so IRPA assigns an A1 to *goals* with *realized* as predicate. In (b) the parser predicts an embedded clause, so IRPA delays the assignment of the A1 to *realized*, and instead introduces two incomplete triples: the first one is predicate-incomplete, with the argument *goals* assigned an A0, waiting to be attached to a predicate. The second one is argument-incomplete with predicate *realized* assigned an A1, waiting for an argument to follow.

(see Section 1). Note the use of incomplete semantic role triples in Figure 2b.

3 Model

We use a discriminative model in order to re-rank the output of the baseline PLTAG parser based on semantic roles assigned by the iSRL system.

3.1 Problem Formulation

Our overall approach is closely related to the discriminative incremental parsing framework of Collins and Roark (2004). The goal is to learn a mapping from input sentences $x \in \mathcal{X}$ to parse trees $y \in \mathcal{Y}$. For a given set of training pairs of sentences and gold-standard parse trees $(x,y) \in \mathcal{X} \times \mathcal{Y}$, the output $\hat{y}$ can be defined as:

$$\hat{y} = \underset{y \in GEN(x)}{\operatorname{argmax}} \Phi(x,y) \cdot \bar{w} \qquad (1)$$

where $GEN(x)$ is a function that enumerates candidate parse trees for a given input x, Φ is a representation that maps each training example (x,y) to a feature vector $\Phi(x,y) \in \mathbb{R}^d$, and $\bar{w} \in \mathbb{R}^d$ is a vector of feature weights.

During training, the task is to estimate $\bar{w}$ given the training examples. In terms of efficiency, a crucial part of Equation (1) is the search strategy over parses produced by *GEN* and, to a smaller degree, the dimensionality of $\bar{w}$. One common decoding technique is to implement a dynamic program, thus avoiding the explicit enumeration of all analyses for a given timestamp (Huang, 2008). However, central to the discriminative approach is the exploration of features that cannot be straightforwardly embedded into the parser using a dynamic program. These include arbitrarily long-range dependencies contained in a parse tree, and more importantly non-isomorphic representations of the input sentence such as its semantic frame, i.e., the set of all semantic roles tripes that pertain to the same predicate. In order to accommodate these, we decode via beam search over candidate parses. We keep a list of the k-best analyses and prune those whose score $scr(x) = \Phi(x,y) \cdot \bar{w}$ falls below a threshold.

3.2 Incremental k-best Parsing

What we described in the previous section could equally apply to k-best re-ranking for full-sentence parsing (e.g., Charniak and Johnson, 2005). For incremental parsing, in addition to outputting $\hat{y}$ for the full sentence, we need to output **prefix trees** $\hat{y}_n$ for every prefix of length $n \in \{1 \ldots N\}$ of sentence $x = a_1 \ldots a_N$ with length N. Let $\langle x_n, \hat{y}_n, n \rangle$, be the state of our model after we have parsed the first n words of sentence x, resulting in analysis $\hat{y}_n$. The initial state is defined as $\langle x_0, \emptyset, 0 \rangle$, where $\emptyset$ is the empty analysis, and the final state is $\langle x, \hat{y}, N \rangle$, which represents a full analysis for the input sentence. We need a function *ADV* that transitions from a state at word a_n to a set of states at word

a_{n+1} by combining the prefix tree $\hat{y}_n$ with a_{n+1}:

$$ADV\big(\langle x_n, \hat{y}_n, n\rangle\big) = \langle x_n, \hat{y}_n, n\rangle \otimes a_{n+1}$$
$$= \{\langle x_{n+1}, \hat{y}_{n+1}, n+1\rangle\}$$

Next, we define the set of states representing prefix trees as π, with $\pi_0 = \{\langle x_0, \emptyset, 0\rangle\}$, and $\pi_n = \cup_{\pi' \in \pi_{n-1}} ADV(\pi')$. We can now redefine $GEN(x_n) = \pi_n$, for any prefix of length n.

We enumerate prefix trees (function *GEN*) with the incremental parser of Demberg et al. (2013). The states of the model are stored in a chart; each cell holds the top-k prefix trees. The transition to the next state (function *ADV*) is performed by combining each prefix tree with a set of candidate of elementary (and prediction) trees via adjunction and substitution, subject to restrictions imposed by incrementallity (see Figure 2). In order to efficiently compute all combinations, the PLTAG parser computes only the fringes (see Section 2) of the prefix tree, and the candidate elementary trees and matches these two; this avoids the computation of the prefix tree entirely.[1]

Each prefix tree is weighted using a probability model estimated over PLTAG operations and the lexicon. This probability is used as a feature in Φ. In addition, we define a set of features of increasing sophistication, which include features specific to PLTAG, standard tree-based features, and, crucially, features extracted from the semantic role triples produced incrementally by the iSRL system of Konstas et al. (2014). The features are computed for each prefix tree y_n, so Φ can be rewritten as $\Phi(x_n, y_n)$, and therefore Equation (1) becomes:

$$\hat{y}_n = \underset{y_n \in \pi_n}{\operatorname{argmax}}\, \Phi(x_n, y_n) \cdot \bar{w} \qquad (2)$$

Our goal now becomes to learn mappings between sentence *prefixes* x_n and *prefix* trees $\hat{y}_n$. In contrast to models that estimate features weights on full sentence parses (Collins and Roark, 2004; Charniak and Johnson, 2005), we do not observe gold-standard prefix trees during training. However, we can use gold-standard lexicon entries when parsing the training data with the PLTAG parser, which gives an approximation of gold-standard prefix trees y_n^+. Finally, during testing, given an unseen sentence x and a trained set of feature weights $\bar{w}$, our model generates prefix trees y_n for every sentence prefix of size n.

[1] As in a chart parser, the prefix tree can be re-constructed by following backpointers in the chart. This is done only for evaluation at the end of the sentence or incrementally on demand.

4 Reranking Features

This section describes the features used for reranking the prefix trees generated by the incremental parser. We include three different classes of features, based on local information from PLTAG elementary trees, based on global and structural information from prefix trees, and based on semantic information provided by iSRL triples. In contrast to work on discriminative full-sentence parsing (e.g., Charniak and Johnson, 2005; Collins and Koo, 2005), we can only use features extracted from the prefix trees being constructed incrementally as the sentence is parsed. The right context of the current word cannot be used, as this would violate incrementality. Every feature combination we try also includes the following baseline features:

Prefix Tree Probability is the log probability of the prefix tree as scored by the probability model of the baseline parser. The score is normalized by prefix length, to avoid getting larger negative log probability scores for longer prefixes.

Elementary Tree Probability is the log probability of the elementary tree corresponding to the word just added to the prefix tree according to the probability model of the baseline parser.

4.1 PLTAG Features

The baseline generative model of the PLTAG parser employs features based on parsing actions, the elementary trees used at each timestamp, and the previous word and PoS tag. In the discriminative model, we extend the locality of these features, as well as addressing sparsity issues arising from rare elementary trees. In all cases, both lexicalized and unlexicalized versions of the elementary trees are used.

Unigram Trees is a family of binary features that record the local elementary trees chosen by the parser for the n-th word, i.e., current word for $n = 1$ and previous word for $n = 2$.

Parent-Unigram Trees is a variation of the previous feature, where we encode the elementary tree of the current word along with the category of the node it attaches to in the prefix tree. This captures the attachment decisions the parser makes.

Bigram Trees are pairs of elementary trees for adjacent words (i.e., the elementary tree currently added to the prefix tree and the previous one). This extends the history the parser has access to,

and captures pairs of elementary trees that are frequently chosen together, e.g., a verb-headed tree with a PP foot node, followed by an NP-headed prepositional tree.

4.2 Tree Features

The following features are inspired by Charniak and Johnson (2005) and attempt to encode properties of the prefix tree, as well as capture regularities for specific syntactic construction such as coordination. Even though the PLTAG parser builds fully connected structures and predicts upcoming context, some constituents in a given prefix tree may be incomplete. We therefore compute the features in this group only for those constituents that have been completed in the current prefix tree (i.e., constituents that are complete at word a_n, but were incomplete at word a_{n-1}). This ensures each of the features is only counted once per constituent. For example, the coordination parallelism feature (see below) should be computed only after all the words in the yield of the CC non-terminal have been observed.

Right Branch encodes the number of nodes on the longest path from the root of the prefix tree to the rightmost pre-terminal. We also include the symmetrical feature which records the number of the remaining nodes in the prefix tree. This feature allows the parser to prefer right-branching trees, commonly found in English syntax.

Coordination Parallelism records whether the two sibling subtrees of a coordination node are identical in terms of structure and node categories up to depth l. We encode identity in a bit mask, and set $l = 4$ (e.g., 1100 means the subtrees have identical children and grandchildren).

Coordination Parallelism Length indicates the binned difference in size between the yields of each sibling subtree under a coordination node. It also stores whether the second subtree is at the end of the sentence.

Heavy stores the category of each node in a completed constituent, along with the binned length of its yield and whether it is at the end of the sentence. This feature captures the tendency of larger constituents to occur towards the end of the sentence.

Neighbors encodes the category of each node in the completed constituent, the binned yield size, and the PoS tags of the l preceding words, were $l = 1$ or 2.

Word stores the current word along with the categories of its l immediate ancestor nodes (excluding pre-terminals); $l = 2$ or 3.

4.3 SRL Features

The features in this group are extracted from the output of iSRL system of Konstas et al. (2014), which annotates prefix trees with semantic roles. The setup proposed in the current paper makes it possible to feed the semantic information back to the PLTAG parser by using it to re-rank the k-best prefix trees generated by the parser. (The re-ranked prefix trees could then also result in better iSRL performance, an issue we will return to in Section 6.3.)

Recall that the SRL information comes in the form of triples $\langle l, r, p \rangle$, where l is a semantic role label and r and p are the words to which the role applies (see Figure 2 for examples). For each feature, we also compute an unlexicalized version by replacing the argument and predicates in the triples with their PoS tags.

Complete SRL Triples stores the complete triples (if any) generated by the current word. The word can be the predicate or the argument in one or more dependency relations involving previous words.

Semantic Frame records all the arguments of a predicate (if present) for frequent semantic labels, i.e., A0, A1 and A2, as well as the presence of a modifier (e.g., AM-TMP, AM-LOC, etc.). This feature usually fires when a verb is added to the prefix tree and generates several complete SRL triples. The feature captures the semantic frame of a verb as a whole (while the previous feature just records it as a collection of triples).

Back-off SRL Triples are generated by removing either the argument, or the predicate, or the role label, from a complete triple. This provides a way of generalizing between triples that share some information without being completely identical.

Predicate/Argument/Role encodes the elements of a complete SRL triple individually (argument, predicate, or role). This allows for further generalization and reduces sparsity.

5 Feature Weight Estimation

We estimate the vector of feature weights $\bar{w}$ in Equation (2) using the averaged structured perceptron algorithm of Collins (2002); we give the pseudocode in Algorithm 1. The perceptron makes T passes over L training examples. In each iteration, for each sentence prefix/prefix tree pair (x_n, y_n), it computes the best scoring prefix tree $\hat{y}_n$ among the candidate prefix trees, given the current feature weights $\bar{w}$. In line 7, the algorithm updates $\bar{w}$ with the difference (if any) between the feature representations of the best scoring prefix tree $\hat{y}_n$ and the approximate gold-standard prefix tree y_n^+ (see Section 3.2). Note that since we use a constant beam during decoding with the PLTAG parser in order to enumerate the set of prefix trees π_n, there is no guarantee that the argmax in line 5 will find the highest scoring (in terms of F-score) prefix tree $y_n^* \neq \hat{y}_n$. Search errors due to the best analysis falling out of the beam at a given prefix length will create errors both when decoding unseen sentences at test time, and when learning the feature weights with the perceptron algorithm. The final weight vector $\bar{w}$ is the average of the weight vectors over T iterations, L examples and N words. The averaging avoids overfitting and produces more stable results (Collins, 2002).

Note that features are computed for every prefix of the input sentence. Recall that the parser avoids the explicit computation of the prefix trees in π_n through the use of the fringes (see Sections 2.1 and 3.2). This is sufficient for the computation of PLTAG and SRL features, but we need to explicitly calculate every prefix tree y_n for the computation of the tree features (see Section 4.2). This is an expensive operation if we are parsing the whole training corpus. To overcome this time bottleneck, we compute features only for those analyses of every input sentence prefix that belongs to the k-best analyses at the end of the sentence. In other words, π_n is the set of only those prefix trees that are used by the k-best analyses at the end of the sentence. This results in a much smaller number of prefix trees that need to be computed for each word. However, during testing, given the trained $\bar{w}$ and an unseen sentence, we compute all features for each prefix length of the sentence, hence calculate all prefix trees in π_n and incrementally re-rank the chart entries of the parser on the fly.

Algorithm 1: Averaged Structured Perceptron

Input: Training Examples: $(x,y)_{i=1}^{L}, x_i = a_1 \dots a_N$

1 $\bar{w} \leftarrow 0$
2 **for** $t \leftarrow 1 \dots T$ **do**
3 **for** $i \leftarrow 1 \dots L$ **do**
4 **for** $n \leftarrow 1 \dots N$ **do**
5 $\hat{y}_n = \text{argmax}_{y_n \in \pi_n} \Phi(x_n, y_n) \cdot \bar{w}$
6 **if** $y_n^+ \neq \hat{y}_n$ **then**
7 $\bar{w} \leftarrow \bar{w} + \Phi(x_n, y_n^+) - \Phi(x_n, y_n)$
8 **return** $\frac{1}{T}\sum_{t=1}^{T}\frac{1}{L}\sum_{i=1}^{L}\sum_{n=1}^{N}\frac{1}{N} w_{t,i,n}$

6 Experiments

6.1 Setup

We use the PLTAG parser of Demberg et al. (2013) to enumerate prefix trees y_n and to compute the prefix tree and word probability scores which we use as features. We also use the iSRL system of Konstas et al. (2014) to generate incremental SRL triples. Their system includes a semantically-enriched lexicon extracted from the Wall Street Journal (WSJ) part of the Penn Treebank corpus (Marcus et al., 1993), converted to PLTAG format. Semantic role annotation is sourced from Propbank. We trained the probability model of the parser and the identification and labeling classifiers of the iSRL system using the intersection of Sections 2–21 of WSJ and the English portion of the CoNLL 2009 Shared Task (Hajič et al., 2009). We learn the weight vector $\bar{w}$ by training the perceptron algorithm also on Sections 2–21 of WSJ (see Section 5 for details). We use the PoS tags predicted by the parser, rather than gold standard PoS tags. Testing is performed on section 23 of WSJ, for sentences up to 40 words.

6.2 Evaluation

In addition to standard full-sentence labeled bracket score, we evaluate our model incrementally, by scoring the prefix trees generated for each sentence prefix (Sangati and Keller, 2013). For each prefix of the input sentence (two words or more), we compute the labeled bracket score for the minimal structure spanning that prefix. The minimal structure is defined as the subtree rooted in the lowest common ancestor of the prefix nodes, while removing any leftover intermediate nodes on the right edge of the subtree that do not have a word in the prefix as their yield.

Although not the main focus of this paper, we also report full-sentence combined SRL accuracy (counting verb-predicates only). This score is obtained by re-applying the iSRL system to the syn-

System	Prec	Rec	F	AUC	SRL
BASELINE	75.51	76.93	76.21	71.49	69.43
TREE	75.99	77.52	76.75	73.02	68.80
SRL	75.99	77.65	76.81	73.97	69.96
TREE+PLTAG	76.67	78.27	77.47	72.27	70.27
TREE+PLTAG +SRL	77.00	78.57	77.77	74.97	70.00

Table 1: Full-sentence parsing results[2], area under the curve (AUC) for the incremental parsing results of Figure 3, and combined SRL score across different groups of features.

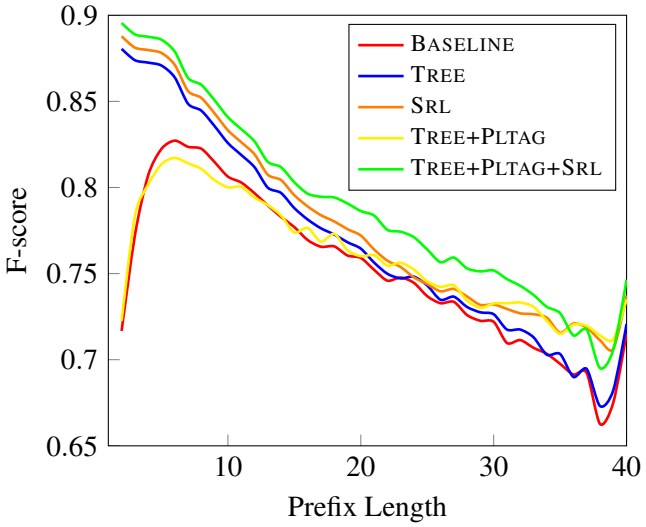

Figure 3: Incremental parsing F-score for increasing sentence prefixes, up to 40 words.

tactic parses output by our re-ranker. (In contrast, Konstas et al. (2014) work with gold-standard syntactic parses.)

We evaluate four variants of our model (see Section 4 for an explanation of the different groups of features):

TREE is the model that uses tree features only; this essentially simulates standard parse reranking approaches such as the one of Charniak and Johnson (2005).

SRL uses only features based on iSRL triples; it provides a proof-of-concept, demonstrating that the semantic information encoded in SRL triples can help the parser building better syntactic trees.

TREE+PLTAG adds PLTAG Features to the TREE model, taking advantage of local information specific to elementary PLTAG trees; TREE+PLTAG essentially provides a strong syntax-only baseline.

TREE+PLTAG+SRL combines SRL features and syntactic features.

Finally, our baseline is the PLTAG parser of Demberg et al. (2013), using the original probability model without any re-ranking. A comparison with other incremental parsers would be desirable, but is not trivial to achieve. This is because the PLTAG parser is trained and evaluated on a version of the Penn Treebank that was converted to PLTAG format. This renders our results not directly comparable to parsers that reproduce the Penn Treebank bracketing. For example, the PLTAG parser produces deeper tree structures informed by Propbank and the noun phrase annotation of Vadas and Curran (2007).

6.3 Results

Figure 3 gives the results of evaluating incremental parsing performance. The x-axis shows prefix length, and the y-axis shows incremental F-score computed as suggested by Sangati and Keller (2013). Each point is averaged over all prefixes of a given length in the test set. To quantify the trends shown in this figure, we also compute the area under the curve (AUC) for each feature combination; this is given in Table 1.

We find that TREE performs consistently better than the baseline for short prefixes (up to the first 20 words), and then is very close to the baseline. This is expected given that tree features add structure-specific information (e.g., about coordination) to the baseline model, and is consistent with results obtained using similar features in the literature (Charniak and Johnson, 2005). Adding PLTAG features (TREE+PLTAG) hurts incremental performance for short prefixes (up to about 20 words), but then performance gradually increases over the baseline and over TREE alone. It seems that the PLTAG features, which are specific to the grammar formalism used, are able to help with longer and more complex prefixes, but introduce noise in smaller prefixes.

The SRL feature set, on the other hand, results in a consistent increase in performance compared

[2]Note that the baseline score is lower than the published F = 77.41 of Demberg et al. (2013). This is expected, since we use a semantically-enriched lexicon, which increases the size of the lexicon, resulting in higher ambiguity per word as well as increased sparsity in the probability model.

to the baseline, across all prefix lengths. SRL provides semantic knowledge, while TREE provides syntactic knowledge, but the performance of both feature sets is very close to each other, up to a prefix length of about 30 words, after which SRL has a clear advantage. SRL features seem to filter out local ambiguity caused by creating prefix trees incrementally and result in correct parses closer to the end of sentence, even without the use of the syntactic information contained in the TREE+PLTAG feature set. Recall that SRL uses information provided by the semantic frame, something that a syntax-only model does not have access to. It seems that this makes it possible for SRL to (partially) compensate for mistakes made by the parser. The AUC of SRL is higher by 0.95 and 1.7 points compared to TREE and TREE+PLTAG, respectively.

We observe an additional boost in performance when using all features together in the TREE+PLTAG+SRL configuration, which outperforms SRL alone by 1.0 points in AUC. Recall that SRL features do not apply to every word; they fire only when semantic information is introduced to the parser via the semantically-enriched lexicon. Hence by adding tree and PLTAG features, which normally apply for every new word, we are able to perform effective re-ranking for all sentence prefixes, which explains the boost in performance. Note that for all variants of our model we observe a dip in performance at around 38 words. This is probably due to noise, caused by the small number of sentences of this length. The upward trend seen around word 40 is probably the effect of observing the end of the sentence, which boosts parsing accuracy.

Turning to full sentence evaluation (Table 1), we observe a similar trend. Both TREE and SRL beat the baseline by about 0.55 points in F-score. Progressively adding features increases performance, with the greatest gain of 1.56 points attained by the combination of all features in TREE+PLTAG+SRL.

We also report combined SRL F-score computed on the re-ranked syntactic trees (rightmost column of Table 1). We find that compared to the baseline, only a small improvement of 0.55 points is achieved by TREE+PLTAG+SRL, while TREE+PLTAG improves by 0.84 points. The syntax-only variant therefore outperforms the full model, but only by a small margin.

7 Related Work

The most similar approach in the literature is Collins and Roark's (2004) re-ranking model for incremental parsing. They learn the syntactic features of Roark (2001) using the perceptron model of Collins (2002). Similar to us, they use the incremental parser to search over candidate parses. However, they limited themselves to local derivation features (akin to our PLTAG features), and do not explore global syntactic feature (tree features) or SRL features. Even though they re-rank the output of an incremental parser, they only evaluate full sentence parsing performance. Other re-ranking approaches to syntactic parsing make use of an extensive set of global features, but apply it on the k-best list of full sentence parses (Charniak and Johnson, 2005; Collins and Koo, 2005) or the k-best list of derivations of a packed forest (Huang, 2008), i.e., these approaches are not incremental.

Based on the CoNLL Shared Tasks (e.g., Hajič et al., 2009), a number of systems exist that perform syntactic parsing and semantic role labeling jointly. Toutanova et al. (2008), Sutton and McCallum (2005) and Li et al. (2010) combine the scores of two separate models, i.e., a syntactic parser and a semantic role labeler, and re-rank the combination using features from each domain. Titov et al. (2009) and Gesmundo et al. (2009), instead of combining models, create a common search space for syntactic parsing and SRL, using a shift reduce-style technique (Nivre, 2007) and learn a latent variable model (Incremental Sigmoid Belief Networks) that optimizes over both tasks at the same time. Volokh and Neumann (2008) use a variant of Nivre's (2007) incremental shift-reduce parser and rely only on the current word and previous content to output partial dependency trees; then they output role labels given the full parser output. In contrast to all the joint approaches, we perform *both* parsing and semantic role labeling strictly incrementally, without having access to the whole sentence, outputting prefix trees and iSRL triples for every sentence prefix. Our approach creates a feedback loop, i.e., we generate a prefix tree using the baseline model, give it as input to iSRL, then re-rank it using a set of syntactic and SRL features. The resulting new prefix tree can then be fed back into iSRL, etc.

8 Conclusions

We started from the observation that human parsing uses semantic knowledge to rule out parses that lead to implausible interpretations. Based on this, we hypothesized that also in NLP, an incremental syntactic parser should benefit from semantic information. To test this hypothesis, we combined an incremental TAG parser with an incremental semantic role labeler. We used the output of the iSRL system to derive features that can be used to re-rank the prefix trees generated by the incremental parser. We found that SRL features, both in isolation and together with standard syntactic features, improve parsing performance, both when measured using full-sentence F-score, and in terms of incremental F-score.

In future work, we plan to combine our incremental parsing/role labeling approach with a compositional model of semantics, which would have to be modified to take semantic role triples as input (rather than words or word pairs). The resulting plausibility estimates could then be used as another source of semantic information for the parser, or employed in down-stream tasks.

Acknowledgments

EPSRC support through grant EP/I032916/1 "An integrated model of syntactic and semantic prediction in human language processing" to Frank Keller and Mirella Lapata is gratefully acknowledged.

References

Altmann, Gerry T. M. and Yuki Kamide. 1999. Incremental interpretation at verbs: Restricting the domain of subsequent reference. *Cognition* 73:247–264.

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

Chelba, Ciprian and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech and Language* 14:283–332.

Collins, Michael. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. pages 1–8.

Collins, Michael and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics* 31(1):25–70.

Collins, Michael and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics, Main Volume*. Barcelona, Spain, pages 111–118.

Demberg, Vera and Frank Keller. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition* 101(2):193–210.

Demberg, Vera, Frank Keller, and Alexander Koller. 2013. Incremental, predictive parsing with psycholinguistically motivated tree-adjoining grammar. *Computational Linguistics* 39(4):1025–1066.

Gesmundo, Andrea, James Henderson, Paola Merlo, and Ivan Titov. 2009. A latent variable model of synchronous syntactic-semantic parsing for multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*. pages 37–42.

Hajič, Jan, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task:

Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning*. Boulder, Colorado, USA.

Huang, Liang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*. Columbus, Ohio, pages 586–594.

Huang, Liang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, pages 1077–1086.

Joshi, Aravind K. and Yves Schabes. 1992. Tree adjoining grammars and lexicalized grammars. In Maurice Nivat and Andreas Podelski, editors, *Tree Automata and Languages*, North-Holland, Amsterdam, pages 409–432.

Konstas, Ioannis, Frank Keller, Vera Demberg, and Mirella Lapata. 2014. Incremental semantic role labeling with tree adjoining grammar. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, pages 301–312.

Li, Junhui, Guodong Zhou, and Tou Hwee Ng. 2010. Joint syntactic and semantic parsing of chinese. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 1108–1117.

Marcus, Mitchell P., Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The Penn treebank. *Computational Linguistics* 19(2):313–330.

Nivre, Joakim. 2007. Incremental non-projective dependency parsing. In *Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. Rochester, New York, pages 396–403.

Roark, Brian. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguististics* 27:249–276.

Sangati, Federico and Frank Keller. 2013. Incremental tree substitution grammar for parsing and word prediction. *Transactions of the Association for Computational Linguistics* 1(May):111–124.

Schuler, William, Samir AbdelRahman, Tim Miller, and Lane Schwartz. 2010. Broad-coverage parsing using human-like memory constraints. *Computational Linguististics* 36(1):1–30.

Schwartz, Lane, Chris Callison-Burch, William Schuler, and Stephen Wu. 2011. Incremental syntactic language models for phrase-based translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Volume 1*. Portland, OR, pages 620–631.

Stoness, Scott C., Joel Tetreault, and James Allen. 2004. Incremental parsing with reference interaction. In Frank Keller, Stephen Clark, Matthew Crocker, and Mark Steedman, editors, *Proceedings of the ACL Workshop Incremental Parsing: Bringing Engineering and Cognition Together*. Barcelona, pages 18–25.

Sutton, Charles and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*. Ann Arbor, Michigan, pages 225–228.

Tan, Ming, Wenli Zhou, Lei Zheng, and Shaojun Wang. 2011. A large scale distributed syntactic, semantic and lexical language model for machine translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Volume 1*. Portland, OR, pages 201–210.

Tanenhaus, Michael K., Michael J. Spivey-Knowlton, Kathleen M. Eberhard, and Julie C. Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science* 268:1632–1634.

Titov, Ivan, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online graph planarisation for synchronous parsing of semantic and syntactic dependencies. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pages 1562–1567.

Toutanova, Kristina, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics* 34(2):161–191.

Vadas, David and James Curran. 2007. Adding noun phrase structure to the penn treebank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics, Prague, Czech Republic, pages 240–247.

Volokh, Alexander and Günter Neumann. 2008. *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, Coling 2008 Organizing Committee, chapter A Puristic Approach for Joint Dependency Parsing and Semantic Role Labeling, pages 213–217.

Xu, Peng, Ciprian Chelba, and Frederick Jelinek. 2002. A study on richer syntactic dependencies for structured language modeling. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Philadelphia, pages 191–198.