Incremental Semantic Role Labeling with Tree Adjoining Grammar

Ioannis Konstas[⋆], **Frank Keller**[⋆], **Vera Demberg**[†] and **Mirella Lapata**[⋆]
⋆: Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
{ikonstas,keller,mlap}@inf.ed.ac.uk
†: Cluster of Excellence Multimodal Computing and Interaction,
Saarland University
vera@coli.uni-saarland.de

Abstract

We introduce the task of incremental semantic role labeling (iSRL), in which semantic roles are assigned to incomplete input (sentence prefixes). iSRL is the semantic equivalent of incremental parsing, and is useful for language modeling, sentence completion, machine translation, and psycholinguistic modeling. We propose an iSRL system that combines an incremental TAG parser with a semantically enriched lexicon, a role propagation algorithm, and a cascade of classifiers. Our approach achieves an SRL F-score of 78.38% on the standard CoNLL 2009 dataset. It substantially outperforms a strong baseline that combines gold-standard syntactic dependencies with heuristic role assignment, as well as a baseline based on Nivre's incremental dependency parser.

1 Introduction

Humans are able to assign semantic roles such as agent, patient, and theme to an incoming sentence before it is complete, i.e., they incrementally build up a partial semantic representation of a sentence prefix. As an example, consider:

(1) The athlete realized [her goals]$_{\text{PATIENT/THEME}}$ were out of reach.

When reaching the noun phrase *her goals*, the human language processor is faced with a semantic role ambiguity: *her goals* can either be the PATIENT of the verb *realize*, or it can be the THEME of a subsequent verb that has not been encountered yet. Experimental evidence shows that the human language processor initially prefers the PATIENT role, but switches its preference to the *theme* role when it reaches the subordinate verb *were*. Such semantic garden paths occur because human language processing occurs word-by-word, and are well attested in the psycholinguistic literature (e.g., Pickering et al., 2000).

Computational systems for performing semantic role labeling (SRL), on the other hand, proceed non-incrementally. They require the whole sentence (typically together with its complete syntactic structure) as input and assign all semantic roles at once. The reason for this is that most features used by current SRL systems are defined globally, and cannot be computed on sentence prefixes.

In this paper, we propose incremental SRL (iSRL) as a new computational task that mimics human semantic role assignment. The aim of an iSRL system is to determine semantic roles while the input unfolds: given a sentence prefix and its partial syntactic structure (typically generated by an incremental parser), we need to (a) identify which words in the input participate in the semantic roles as arguments and predicates (the task of **role identification**), and (b) assign correct semantic labels to these predicate/argument pairs (the task of **role labeling**). Performing these two tasks incrementally is substantially harder than doing it non-incrementally, as the processor needs to commit to a role assignment on the basis of incomplete syntactic and semantic information. As an example, take (1): on reaching *athlete*, the processor should assign this word the AGENT role, even though it has not seen the corresponding predicate yet. Similarly, upon reaching *realized*, the processor can complete the AGENT role, but it should also predict that this verb also has a PATIENT role, even though it has not yet encountered the argument that fills this role. A system that performs SRL in a fully incremental fashion therefore needs to be able to assign **incomplete semantic roles,** unlike existing full-sentence SRL models.

The uses of incremental SRL mirror the applications of incremental parsing: iSRL models can be used in language modeling to assign better string probabilities, in sentence completion systems to

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 301–312,
October 25-29, 2014, Doha, Qatar.

provide semantically informed completions, in any real time application systems, such as dialog processing, and to incrementalize applications such as machine translation (e.g., in speech-to-speech MT). Crucially, any comprehensive model of human language understanding needs to combine an incremental parser with an incremental semantic processor (Padó et al., 2009; Keller, 2010).

The present work takes inspiration from the psycholinguistic modeling literature by proposing an iSRL system that is built on top of a cognitively motivated incremental parser, viz., the Psycholinguistically Motivated Tree Adjoining Grammar parser of Demberg et al. (2013). This parser includes a predictive component, i.e., it predicts syntactic structure for upcoming input during incremental processing. This makes PLTAG particularly suitable for iSRL, allowing it to predict incomplete semantic roles as the input string unfolds. Competing approaches, such as iSRL based on an incremental dependency parser, do not share this advantage, as we will discuss in Section 4.3.

2 Related Work

Most SRL systems to date conceptualize semantic role labeling as a supervised learning problem and rely on role-annotated data for model training. Existing models often implement a two-stage architecture in which role identification and role labeling are performed in sequence. Supervised methods deliver reasonably good performance with F-scores in the low eighties on standard test collections for English (Màrquez et al., 2008; Björkelund et al., 2009).

Current approaches rely primarily on syntactic features (such as path features) in order to identify and label roles. This has been a mixed blessing as the path from an argument to the predicate can be very informative but is often quite complicated, and depends on the syntactic formalism used. Many paths through the parse tree are likely to occur infrequently (or not at all), resulting in very sparse information for the classifier to learn from. Moreover, as we will discuss in Section 4.4, such path information is not always available when the input is processed incrementally. There is previous SRL work employing Tree Adjoining Grammar, albeit in a non-incremental setting, as a means to reduce the sparsity of syntax-based features. Liu and Sarkar (2007) extract a rich feature set from TAG derivations and demonstrate that this improves SRL performance.

In contrast to incremental parsing, incremental semantic role labeling is a novel task. Our model builds on an incremental Tree Adjoining Grammar parser (Demberg et al., 2013) which predicts the syntactic structure of upcoming input. This allows us to perform incremental parsing and incremental SRL in tandem, exploiting the predictive component of the parser to assign (potentially incomplete) semantic roles on a word-by-word basis. Similar to work on incremental parsing that evaluates incomplete trees (Sangati and Keller, 2013), we evaluate the incomplete semantic structures produced by our model.

3 Psycholinguistically Motivated TAG

Demberg et al. (2013) introduce Psycholinguistically Motivated Tree Adjoining Grammar (PLTAG), a grammar formalism that 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.** Elementary trees that contain a foot node are called **auxiliary trees**; those that do not are called **initial trees**. Examples for TAG elementary trees are given in Figure 1a–c.

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 can be used to predict syntactic structure anchored by words that appear later in an incremental derivation. The use of prediction trees ensures that fully connected **prefix trees** can be built for every prefix of the input sentence.

Each node in a prediction tree carries **markers** to indicate that this node was predicted, rather than being anchored by the current sentence prefix. An example is Figure 1d, which contains a prediction tree with marker "1". In PLTAG, markers are eliminated through a new operation called **verification**, which matches them with the nodes

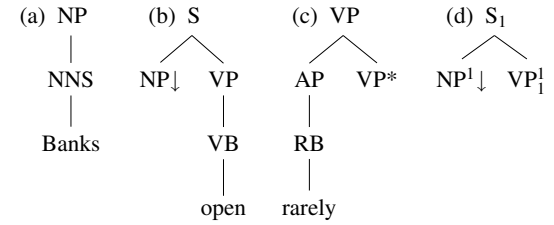


Figure 1: PLTAG lexicon entries: (a) and (b) initial trees, (c) auxiliary tree, (d) prediction tree.

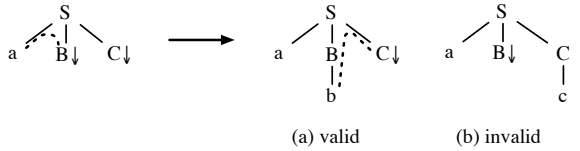


Figure 3: The current fringe (dashed line) indicates where valid substitutions can occur. Other substitutions result in an invalid prefix tree.

of non-predictive elementary trees. An example of a PLTAG derivation is given in Figure 2. In step 1, a prediction tree is introduced through substitution, which then allows the adjunction of an adverb in step 2. Step 3 involves the verification of the marker introduced by the prediction tree against the elementary tree for *open*.

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 3 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 3b, which is not a valid prefix tree (i.e., it represents a non-incremental derivation).

The parsing algorithm proposed by Demberg et al. (2013) exploits fringes to tabulate intermediate results. It manipulates a chart in which each cell (i, f) contains all the prefix trees whose first i leaves are the first i words and whose current fringe is f. To extend the prefix trees for i to the prefix trees for $i+1$, the algorithm retrieves all current fringes f such that the chart has entries in the cell (i, f). For each such fringe, it needs to determine the elementary trees in the lexicon that can be combined with f using substitution or adjunction. In spite of the large size of a typical TAG lexicon, this can be done efficiently, as it only requires matching the current fringes. For each match, the parser then computes the new prefix trees and its new current fringe f' and enters it into cell $(i+1, f')$.

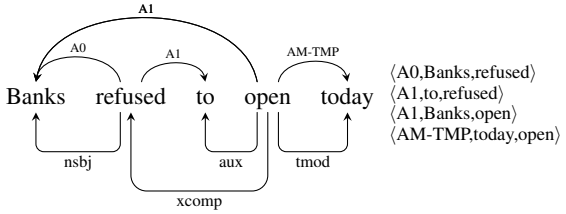


Figure 4: Syntactic dependency graph with semantic role annotation and the accompanying semantic triples, for *Banks refused to open today*.

Demberg et al. (2013) convert the Penn Treebank (Marcus et al., 1993) into TAG format by enriching it with head information and argument/modifier information from Propbank (Palmer et al., 2005). This makes it possible to decompose the Treebank trees into elementary trees as proposed by Xia et al. (2000). Prediction trees can be learned from the converted Treebank by calculating the connection path (Mazzei et al., 2007) at each word in a tree. Intuitively, a prediction tree for word w_n contains the structure that is necessary to connect w_n to the prefix tree $w_1 \ldots w_{n-1}$, but is not part of any of the elementary trees of $w_1 \ldots w_{n-1}$. Using this lexicon, a probabilistic model over PLTAG operations can be estimated following Chiang (2000).

4 Model

4.1 Problem Formulation

In a typical semantic role labeling scenario, the goal is to first identify words that are predicates in the sentence and then identify and label all the arguments for each predicate. This translates into spotting specific words in a sentence that represent the predicate's arguments, and assigning predefined semantic role labels to them. Note that in this work we focus on verb predicates only. The output of a semantic role labeler is a set of semantic dependency triples $\langle l, a, p \rangle$, with $l \in \mathcal{R}$, and $a, p \in \mathbf{w}$, where $\mathcal{R}$ is a set of semantic role labels denoting a specific relationship between a predicate and an argument (e.g., ARG0, ARG1, ARGM in Propbank), $\mathbf{w}$ is the list of words in the sentence, l denotes a specific role label, a the argument, and p the predicate. An example is shown in Figure 4.

As discussed in the introduction, standard semantic role labelers make their decisions based on evidence from the whole sentence. In contrast, our aim is to assign semantic roles incrementally, i.e.,

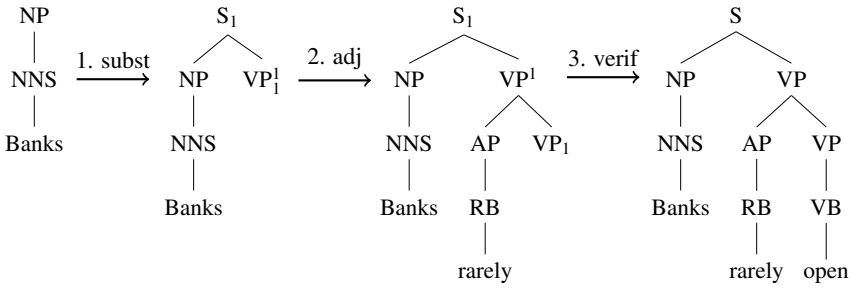


Figure 2: Incremental parse for *Banks rarely open* using the operations substitution (with a prediction tree), adjunction, and verification.

we want to produce a set of (potentially incomplete) semantic dependency triples for each 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. Furthermore, the triples themselves may be incomplete, as either the predicate or the argument may not have been observed yet (**predicate-incomplete** or **argument-incomplete** triples).

Our iSRL system relies on PLTAG, using a semantically augmented lexicon. We parse an input sentence incrementally, applying a novel incremental role propagation algorithm (IRPA) that creates or updates existing semantic triple candidates whenever an elementary (or prediction) tree containing role information is attached to the existing prefix tree. As soon as a triple is completed we apply a two-stage classification process, that first identifies whether the predicate/argument pair is a good candidate, and then disambiguates role labels in case there is more than one candidate.

4.2 Semantic Role Lexicon

Recall that Propbank is used to construct the PLTAG treebank, in order to distinguish between arguments and modifiers, which result in elementary trees with substitution nodes, and auxiliary trees, i.e., trees with a foot node, respectively (see Figure 1). Conveniently, we can use the same information to also enrich the extracted lexicon with the semantic role annotations, following the process described by Sayeed and Demberg (2013).[1] For arguments, annotations are retained on the substitution node in the parental tree, while for modifiers, the role annotation is displayed on the foot node of the auxiliary tree. Note that we display role annotation on traces that are leaf nodes, which enables us to recover long-range dependencies (third and fifth tree in Figure 5a). Likewise, we annotate prediction trees with semantic roles, which enables our system to predict upcoming incomplete triples.

Our annotation procedure unavoidably introduces some role ambiguity, especially for frequently occurring trees. This can give rise to two problems when we generate semantic triples incrementally: IRPA tends to create many spurious candidate semantic triples for elementary trees that correspond to high frequency words (e.g., prepositions or modals). Secondly, a semantic triple may be identified correctly but is assigned several role labels. (See the elementary tree for *refuse* in Figure 5a.) We address these issues by applying classifiers for role label disambiguation at every parsing operation (substitution, adjunction, or verification), as detailed in Section 4.4.

4.3 Incremental Role Propagation Algorithm

The main idea behind IRPA is to create or update existing semantic triples as soon as there is available role information during parsing. Our algorithm (lines 1–6 in Algorithm 1) is applied after every PLTAG parsing operation, i.e., when an elementary or prediction tree $\mathcal{T}$ is adjoined to a particular integration point node π_{ip} of the prefix tree of the sentence, via substitution or adjunction (lines 3–4).[2] In case an elementary tree $\mathcal{T}_v$ verifies a prediction tree $\mathcal{T}_{pr}$ (lines 5–6), the same methodology applies, the only difference being that we have to tackle multiple integration point nodes $\mathcal{T}_{pr,ip}$, one for each prediction marker of $\mathcal{T}_{pr}$ that matches the corresponding nodes in $\mathcal{T}_v$.

For simplicity of presentation, we will use a concrete example, see Figure 5. Figure 5a shows the lexicon entries for the words of the sentence

[1] Contrary to Sayeed and Demberg (2013) we put role label annotations for PPs on the preposition rather than their NP child, following of the CoNLL 2005 shared task (Carreras and Màrquez, 2005).

[2] Prediction tree $\mathcal{T}_{pr}$ in our algorithm is only used during verification, so it set to nil for substitution and adjunction operations.

Banks refused to open. Naturally, some nodes in the lexicon trees might have multiple candidate role labels. For example, the substitution NP node of the second tree takes two labels, namely A0 and A1. These stem from different role *signatures* when the same elementary tree occurs in different contexts during training (A1 only on the NP; A0 on the NP and A1 on S). For simplicity's sake, we collapse different signatures, and let a classifier labeller to disambiguate such cases (see Section 4.4).

Algorithm 1 Incremental Role Propagation Alg.

```
1:  procedure IRPA(π_ip, T, T_pr)
2:      Σ ← ∅        ▷ Σ is a dictionary of (π_ip, ⟨l,a,p⟩) pairs
3:      if parser operation is substitution or adjunction then
4:          CREATE-TRIPLES(π_ip, T)
5:      else if parser operation is verification then
6:          CREATE-TRIPLES-VERIF(π_ip, T, T_pr)
        return set of triples ⟨l,a,p⟩ for prefix tree π
7:  procedure CREATE-TRIPLES(π_ip, T)
8:      if HAS-ROLES(π_ip) then
9:          UPDATE-TRIPLE(π_ip, T)
10:     else if HAS-ROLES(T) then
11:         T_ip ← substitution or foot node of T
12:         ADD-TRIPLE(π_ip, T_ip, T)
13:     for all remaining nodes n ∈ T with roles do
14:         ADD-TRIPLE(π_ip, n, T)        ▷ incomplete triples
15: procedure CREATE-TRIPLES-VERIF(π_ip, T_v, T_pr)
16:     if HAS-ROLES(T_v) then
17:         anchor ← lexeme of T_v
18:         for all T_ip ← node in T_v with role do
19:             T_pr,ip ← matching node of T_ip in T_pr
20:             CREATE-TRIPLES(T_pr,ip, T_v)
        ▷ Process the rest of covered nodes in T_pr with roles
21:     for all remaining T_pr,ip ← node in T_pr with role do
22:         UPDATE-TRIPLE(T_pr,ip, T_pr)
23: function UPDATE-TRIPLE(π_ip, T)
24:     dep ← FIND-INCOMPLETE(Σ, T_ip)
25:     anchor ← lexeme of T
26:     if anchor of T is predicate then
27:         SET-PREDICATE(dep, anchor)
28:     else if anchor of T is argument then
29:         SET-ARGUMENT(dep, anchor)
        return dep
30: procedure ADD-TRIPLE(π_ip, T_ip, T)
31:     dep ← ⟨[roles of T_ip], nil, nil⟩
32:     anchor ← lexeme of T
33:     if anchor of T is predicate then
34:         SET-PREDICATE(dep, anchor)
35:         SET-ARGUMENT(dep, head of π_ip)
36:     else if anchor of T is argument then
37:         if T is auxiliary then                ▷ adjunction
38:             SET-ARGUMENT(dep, anchor)
39:         else        ▷ substitution: arg is head of prefix tree
40:             SET-ARGUMENT(dep, head of T_ip)
41:             pred ← find dep ∈ Σ with matching π_ip
42:             SET-PREDICATE(dep, pred)
43:     Σ ← (π_ip, dep)
```

Once we process *Banks*, the prefix tree becomes the lexical entry for this word, see the first column of Figure 5b. Next, we process *refused*: the parser substitutes the prefix tree into the elementary tree $\mathcal{T}$ of *refused*;[3] the integration point π_{ip} on the prefix tree is the topmost NP. Since the operation is a substitution (line 3), we create triples between $\mathcal{T}$ and π_{ip} via CREATE-TRIPLES (lines 7–12). π_{ip} does not have any role information (line 8), so we proceed to add a new semantic triple between the role-labeled integration point $\mathcal{T}_{ip}$, i.e., substitution NP node of $\mathcal{T}$, and π_{ip}, via ADD-TRIPLE (lines 30–43). First, we create an incomplete semantic triple with all roles from $\mathcal{T}_{ip}$ (line 31). Then we set the predicate to the anchor of $\mathcal{T}$ to be the word *refused*, and the argument to be the head word of the prefix tree, *Banks* (lines 34–35). Note that predicate identification is a trivial task based on part-of-speech information in the elementary tree.[4]

Then, we add the pair (NP $\rightarrow$ ⟨{A0,A1},Banks, refused⟩) to a dictionary (line 43). Storing the integration point along with the semantic triple is essential, to be able to recover incomplete triples in later stages of the algorithm. Finally, we repeat this process for all remaining nodes on $\mathcal{T}$ that have roles, in our example the substitution node S (lines 13–14). This outputs an incomplete triple, ⟨{A1},nil,refused⟩.

Next, the parser decides to substitute a prediction tree (third tree in Figure 5a) into the substitution node S of the prefix tree. Since the integration point is on the prefix tree and has role information (line 8), the corresponding triple should already be present in our dictionary. Upon retrieving it, we set the nil argument to the anchor of the incoming tree. Since it is a prediction tree, we set it to the root of the tree, namely S_2 (phrase labels in triples are denoted by italics), but mark the triple as yet incomplete. This distinction allows us to fill in the correct lexical information once it becomes available, i.e, when the tree gets verified. We also add an incomplete triple for the trace t in the subject position of the prediction tree, as described above. Note that this triple contains multiple roles; this is expected given that prediction trees are unlexicalized and occur in a wide variety of contexts.

When the next verb arrives, the parser successfully verifies it against the embedded prediction

[3] PLTAG parsing operations can occur in two ways: An elementary tree can be substituted into the substitution node of the prefix tree, or the prefix tree can be substituted into a node of an elementary tree. The same holds for adjunction.

[4] Most predicates can be identified as anchors of non-modifier auxiliary trees. However, there are exceptions to this rule, i.e., modifier auxiliary trees and non-modifier non-auxiliary trees being also verbs in our lexicon, hence the use of the more reliable POS tags.

	IRPA	MaltParser
Banks	–	–
refused	⟨{A0,A1},Banks,refused⟩, ⟨A1,S_2,refused⟩, ⟨{A0,A1,A2},*t*,nil⟩	⟨A0,Banks,refused⟩
to	–	–
open	⟨A1,to,refused⟩, ⟨A1,Banks,open⟩	⟨A1,to,refused⟩, ⟨A0,Banks,open⟩
today	⟨AM-TMP,today,open⟩	⟨AM-TMP,today,open⟩

Table 1: Complete and incomplete semantic triple generation, comparing IRPA and a system that maps gold-standard role labels onto MaltParser incremental dependencies for Figure 4.

tree within the prefix tree (last step of Figure 5b). Our algorithm first cycles through all nodes that match between the verification tree $\mathcal{T}_v$ and the prediction tree $\mathcal{T}_{pr}$ and will complete or create new triples via CREATE-TRIPLES (lines 18–20). In our example, the second semantic triple gets completed by replacing S_2 with the head of the subtree rooted in S. Normally, this would be the verb *open*, but in this case the verb is followed by the infinitive marker *to*, hence we heuristically set it to be the argument of the triple instead, following Carreras and Màrquez (2005). For the last triple, we set the predicate to the anchor of $\mathcal{T}_v$ *open*, and now are able to remove the excess role labels A0 and A2. This illustrated how the lexicalized verification tree disambiguates the semantic information stored in the prediction tree. Finally, trace *t* is set to the closest NP head that is below the same phrase subtree, in this case *Banks*. Note that *Banks* is part of two triples as shown in the last tree of Figure 5b: it is either an A0 or an A1 for *refused* and an A1 for *open*.

We are able to create incomplete semantic triples after the prediction of the upcoming verb at step 2, as shown in Figure 5b. This is not possible using an incremental dependency parser such as MaltParser (Nivre et al., 2007) that lacks a predictive component. Table 1 illustrates this by comparing the output of IRPA for Figure 5b with the output of a baseline system that maps role labels onto the syntactic dependencies in Figure 4, generated incrementally by MaltParser (see Section 5.3 for a description of the MaltParser baseline). MaltParser has to wait for the verb *open* before outputting the relevant semantic triples. In contrast, IRPA outputs incomplete triples as soon as the information is available, and later on updates its decision. (MaltParser also incorrectly assigns A0 for the *Banks–open* pair.)

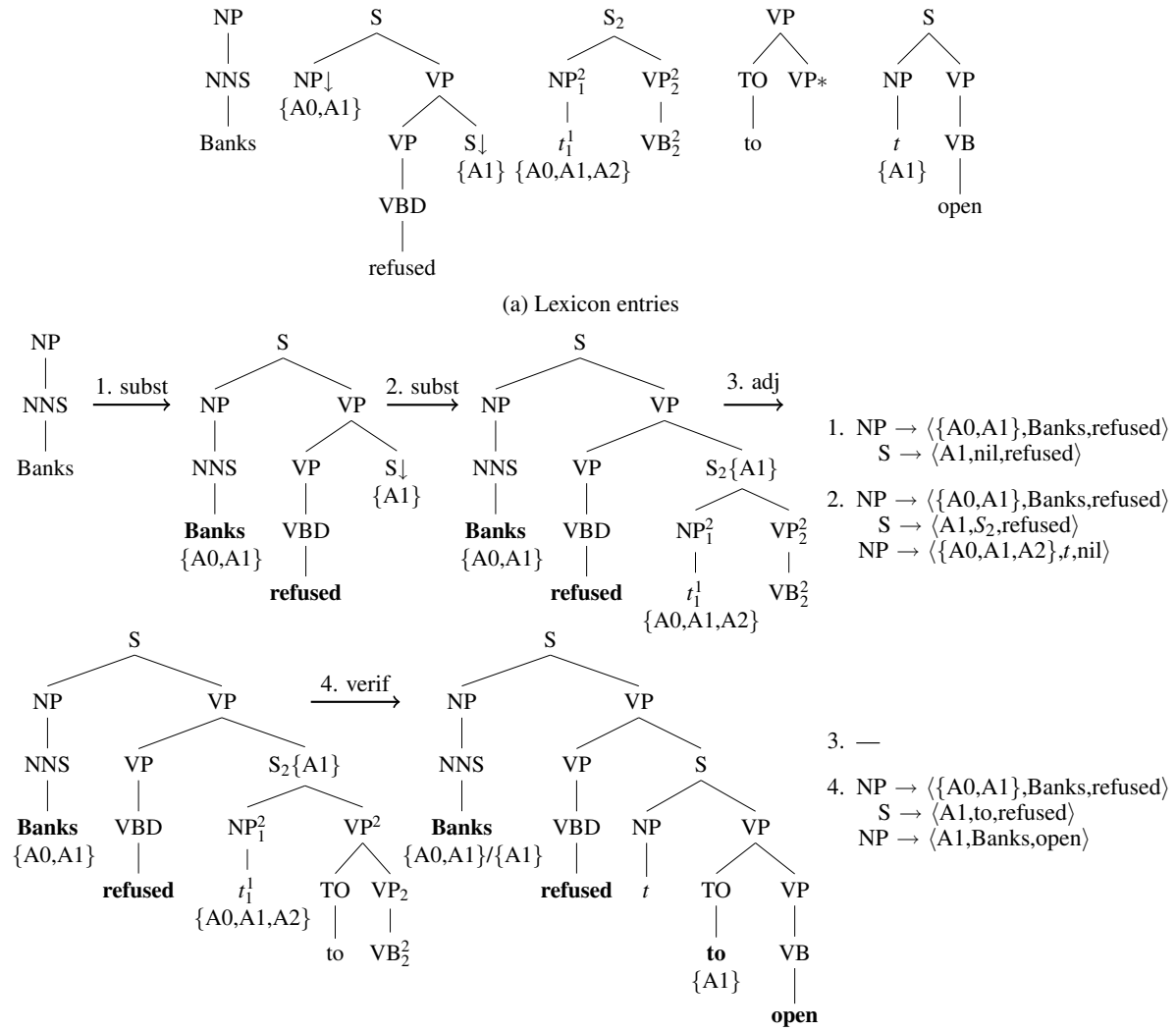
4.4 Argument Identification and Role Label Disambiguation

IRPA produces semantic triples for every role annotation present in the lexicon entries, which will often overgenerate role information. Furthermore, some triples have more than one role label attached to them. During verification, we are able to filter out the majority of labels in the corresponding prediction trees; However, most triples are created via substitution and adjunction.

In order to address these problems we adhere to the following classification and ranking strategy: after each semantic triple gets completed, we perform a binary classification that evaluates its suitability as a whole, given bilexical and syntactic information. If the triple is identified as a good candidate, then we perform multi-class classification over role labels: we feed the same bilexical and syntactic information to a logistic classifier, and get a ranked list of labels. We then use this list to re-rank the existing ambiguous role labels in the semantic triple, and output the top scoring ones.

The identifier is a binary L2-loss support vector classifier, and the role disambiguator an L2-regularized logistic regression classifier, both implemented using the efficient LIBLINEAR framework of Fan et al. (2008). The features used are based on Björkelund et al. (2009) and Liu and Sarkar (2007), and are listed in Table 2.

The bilexical features are: predicate POS tag, predicate lemma, argument word form, argument POS tag, and position. The latter indicates the position of the argument relative to the predicate, i.e., before, on, or after. The syntactic features are: the predicate and argument elementary trees without the anchors (to avoid sparsity), the category of the integration point node on the prefix tree where the elementary tree of the argument attaches to, an alphabetically ordered set of the categories of the fringe nodes of the prefix tree after attaching the argument tree, and the path of PLTAG operations applied between the argument and the predicate. Note that most of the original features used by Björkelund et al. (2009) and others are not applicable in our context, as they exploit information that is not accessible incrementally. For example, sibling information to the right of the word is not available. Furthermore, our PLTAG parser does not compute syntactic dependencies, hence these cannot serve as features (and in any case not all dependencies are available incrementally, see Figure 4). To counterbalance this, we use local syntactic information stored in the fringe of the pre-



(a) Lexicon entries

(b) Incremental parsing using PLTAG and incremental propagation of roles

Figure 5: Incremental Role Propagation Algorithm application for the sentence *Banks refused to open*.

Bilexical	Syntactic
PredPOS	PredElemTree
PredLemma	ArgElemTree
ArgWord	IntegrationPoint
ArgPOS	PrefixFringe
Position	OperationPath

Table 2: Features for argument identification and role label disambiguation.

fix tree. We also store the series of operations applied by our parser between argument and predicate, in an effort to emulate the effect of recovering longer-range patterns.

5 Experimental Design

5.1 PLTAG and Classifier Training

We extracted the semantically-enriched lexicon and trained the PLTAG parser by converting the Wall Street Journal part of Penn Treebank to PLTAG format. We used Propbank to retrieve semantic role annotation, as described in Section 4.2. We trained the PLTAG parser according to Demberg et al. (2013) and evaluated the parser on section 23, on sentences with 40 words or less, given gold POS tags for each word, and achieved a labeled bracket F_1 score of 79.41.

In order to train the argument identification and role label disambiguation classifiers, we used the English portion of the CoNLL 2009 Shared Task (Hajič et al., 2009; Surdeanu et al., 2008). It consists of the Penn Treebank, automatically converted to dependencies following Johansson and

Nugues (2007), accompanied by semantic role label annotation for every argument pair. The latter is converted from Propbank based on Carreras and Màrquez (2005). We extracted the bilexical features for the classifiers directly from the gold standard annotation of the training set. The syntactic features were obtained as follows: for every sentence in the training set we applied IRPA using the trained PLTAG parser, with gold standard lexicon entries for each word of the input sentence. This ensures near perfect parsing accuracy. Then for each semantic triple predicted incrementally, we extracted the relevant syntactic information in order to construct training vectors. If the identified predicate-argument pair was in the gold standard then we assigned a positive label for the identification classifier, otherwise we flagged it as negative. For those pairs that are not identified by IRPA but exist in the gold standard (false negatives), we extracted syntactic information from already identified similar triples, as follows: We first look for correctly identified arguments, wrongly attached to a different predicate and re-create the triple with correct predicate/argument information. If no argument is found, we then pick the argument in the list of identified arguments for a correct predicate with the same POS-tag as the gold-standard argument. In the case of the role label disambiguation classifier we just assign the gold label for every correctly identified pair, and ignore the (possibly ambiguous) predicted one. After tuning on the development set, the argument identifier achieved an accuracy of 92.18%, and the role label disambiguation classifier, 82.37%.

5.2 Evaluation

The focus of this paper is to build a system that is able to output semantic role labels for predicate-argument pairs incrementally, as soon as they become available. In order to properly evaluate such a system, we need to measure its performance incrementally. We propose two different cumulative scores for assessing the (possibly incomplete) semantic triples that have been created so far, as the input is processed from left to right, *per word.* The first metric is called Unlabeled Prediction Score (UPS) and gets updated for every identified argument or predicate, even if the corresponding semantic triple is incomplete. Note that UPS does not take into account the role label, it only measures predicate and argument identification. In this respect it is analogous to unlabeled dependency accuracy reported in the parsing literature. We expect a model that is able to predict semantic roles to achieve an improved UPS result compared to a system that does not do prediction, as illustrated in Table 1. Our second score, Combined Incremental SRL Score (CISS), measures the identification of complete semantic role triples (i.e., correct predicate, predicate sense, argument, and role label) per word; by the end of the sentence, CISS coincides with standard combined SRL accuracy, as reported in CoNLL 2009 SRL-only task. This score is analogous to labeled dependency accuracy in parsing.

Note that conventional SRL systems such as Björkelund et al. (2009) typically assume gold standard syntactic information. In order to emulate this, we give our parser gold standard lexicon entries for each word in the test set; these contain all possible roles observed in the training set for a given elementary tree (and all possible senses for each predicate). This way the parser achieves a syntactic parsing F_1 score of 94.24, thus ensuring the errors of our system can be attributed to IRPA and the classifiers. Also note that we evaluate on verb predicates only, therefore trivially reducing the task of predicate identification to the simple heuristic of looking for words in the sentence with a verb-related POS tag and excluding auxiliaries and modals. Likewise, predicate sense disambiguation on verbs presumably is trivial, as we observed almost no ambiguity of senses among lexicon entries of the same verb (we adhered to a simple majority baseline, by picking the most frequent sense, given the lexeme of the verb, in the few ambiguous cases). It seems that the syntactic information held in the elementary trees discriminates well among different senses.

5.3 System Comparison

We evaluated three configurations of our system. The first configuration (iSRL) uses all semantic roles for each PLTAG lexicon entry, applies the PLTAG parser, IRPA, and both classifiers to perform identification and disambiguation, as described in Section 4. The second one (Majority-Baseline), solves the problem of argument identification and role disambiguation without the classifiers. For the former we employ a set of heuristics according to Lang and Lapata (2014), that rely on gold syntactic dependency information, sourced from CoNLL input. For the latter, we choose the most frequent role given the gold standard dependency relation label for the particular argument. Note that dependencies have been produced in view of the whole sentence and not incrementally.

System	Prec	Rec	F1
iSRL-Oracle	91.00	80.26	85.29
iSRL	81.48	75.51	78.38
Majority-Baseline	71.05	58.10	63.92
Malt-Baseline	60.90	46.14	52.50

Table 3: Full-sentence combined SRL score

This gives the baseline a considerable advantage especially in case of longer range dependencies. The third configuration (iSRL-Oracle), is identical to iSRL, but uses the gold standard roles for each PLTAG lexicon entry, and thus provides an upper-bound for our methodology. Finally, we evaluated against Malt-Baseline, a variant of Majority-Baseline that uses the MaltParser of Nivre et al. (2007) to provide labeled syntactic dependencies MaltParser is a state-of-the-art shift-reduce dependency parser which uses an incremental algorithm. Following Beuck et al. (2011), we modified the parser to provide intermediate output at each word by emitting the current state of the dependency graph before each shift step. We trained MaltParser using the arc-eager algorithm (which outperformed the other parsing algorithms available with MaltParser) on the CoNLL dataset, achieving a labeled dependency accuracy of 89.66% on section 23.

6 Results

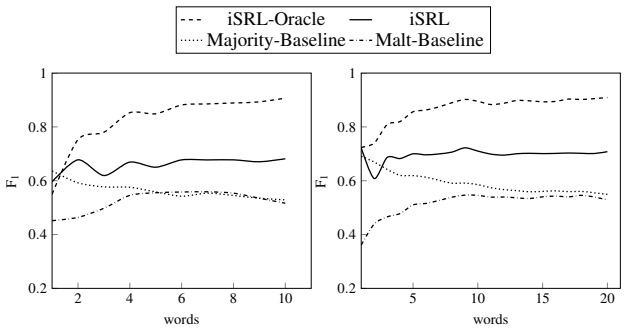
Figures 6 and 7 show the results on the incremental SRL task. We plot the F_1 for Unlabeled Prediction Score (UPS) and Combined Incremental SRL Score (CISS) per word, separately for sentences of lengths 10, 20, 30, and 40 words. The task gets harder with increasing sentence length, hence we can only meaningfully compare the average scores for sentence of the same length. (This approach was proposed by Sangati and Keller 2013 for evaluating the performance of incremental parsers.)

The UPS results in Figure 6 clearly show that our system (iSRL) outperforms both baselines on unlabeled argument and predicate prediction, across all four sentence lengths. Furthermore, we note that the iSRL system achieves a near-constant performance for all sentence prefixes. Our PLTAG-based prediction/verification architecture allows us to correctly predict incomplete semantic role triples, even at the beginning of the sentence. Both baselines perform worse than the iSRL system in general. Moreover, the Malt-Baseline performs badly on the initial sentence prefixes (up to word 10), presumably as it does not benefit from syntactic prediction, and thus cannot generate incomplete triples early in the sentence, as illustrated in Table 1. The Majority-Baseline also does not do prediction, but it has access to gold-standard syntactic dependencies, and thus outperforms the Malt-Baseline on initial sentence prefixes. Note that due to prediction, our system tends to over-generate incomplete triples in the beginning of sentences, compared to non-incremental output, which may inflate UPS for the first words. However, this cancels out later in the sentence if triples are correctly completed; failure to do so would decrease UPS. The near-constant performance of our output illustrates this phenomenon. Finally, the iSRL-Oracle outperforms all other systems, as it benefits from correct role labels and correct PLTAG syntax, thus providing an upper limit on performance.

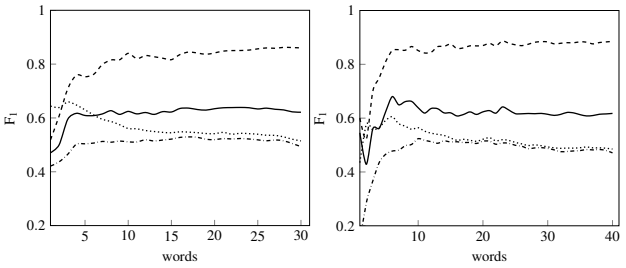
The CISS results in Figure 7 present a similar picture. Again, the iSRL system outperforms both baselines at all sentence lengths. In addition, it shows particularly strong performance (almost at the level of the iSRL-Oracle) at the beginning of the sentence. This presumably is due to the fact that our system uses prediction and is able to identify correct semantic role triples earlier in the sentence. The baselines also show higher performance early in the sentence, but to a lesser degree.

Table 3 reports traditional combined SRL scores for full sentences over all sentence lengths, as defined for the CoNLL task. Our iSRL system outperforms the Majority-Baseline by almost 15 points, and the Malt-Baseline by 25 points. It remains seven points below the iSRL-Oracle upper limit.

Finally, in order to test the effect of syntactic parsing on our system, we also experimented with a variant of our iSRL system that utilizes all lexicon entries for each word in the test set. This is similar to performing the CoNLL 2009 joint task, which is designed for systems that carry out both syntactic parsing and semantic role labeling. This variant achieved a full sentence F-score of 68.0%, i.e., around 10 points lower than our iSRL system. This drop in score correlates with the difference in syntactic parsing F-score between the two versions of PLTAG parser (94.24 versus 79.41), and is expected given the high ambiguity of the lexicon entries for each word. Note, however, that the full-parsing version of our system still outperforms Malt-Baseline by 15 points.

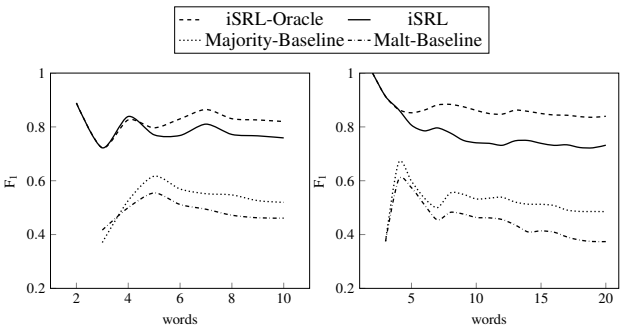


(a) 10 words (b) 20 words

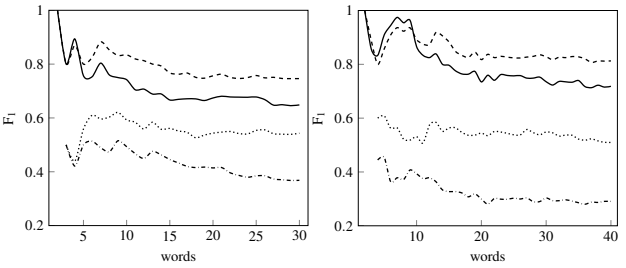


(c) 30 words (d) 40 words

Figure 6: Unlabeled Prediction Score (UPS)



(a) 10 words (b) 20 words



(c) 30 words (d) 40 words

Figure 7: Combined iSRL Score (CISS)

7 Conclusions

In this paper, we introduced the new task of incremental semantic role labeling and proposed a system that solves this task by combining an incremental TAG parser with a semantically enriched lexicon, a role propagation algorithm, and a cascade of classifiers. This system achieved a full-sentence SRL F-score of 78.38% on the standard CoNLL dataset. Not only is the full-sentence score considerably higher than the Majority-Baseline (which is a strong baseline, as it uses gold-standard syntactic dependencies), but we also observe that our iSRL system performs well incrementally, i.e., it predicts both complete and incomplete semantic role triples correctly early on in the sentence. We attributed this to the fact that our TAG-based architecture makes it possible to predict upcoming syntactic structure together with the corresponding semantic roles.

Acknowledgments

EPSRC support through grant EP/I032916/1 “An integrated model of syntactic and semantic prediction in human language processing” to FK and ML is gratefully acknowledged.

References

Beuck, Niels, Arne Khn, and Wolfgang Menzel. 2011. Incremental parsing and the evaluation of partial dependency analyses. In *Proceedings of the 1st International Conference on Dependency Linguistics*. Depling 2011.

Björkelund, Anders, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, Stroudsburg, PA, USA, CoNLL ’09, pages 43–48.

Carreras, Xavier and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL ’05, pages 152–164.

Chiang, David. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. pages 456–463.

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

Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Li-

blinear: A library for large linear classification. *Journal of Machine Learning Research* 9:1871–1874.

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 (CoNLL-2009), June 4-5*. Boulder, Colorado, USA.

Johansson, Richard and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for english. In Joakim Nivre, Heiki-Jaan Kalep, Kadri Muischnek, and Mare Koit, editors, *NODALIDA 2007 Proceedings*. University of Tartu, pages 105–112.

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.

Keller, Frank. 2010. Cognitively plausible models of human language processing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Companion Volume: Short Papers*. Uppsala, pages 60–67.

Lang, Joel and Mirella Lapata. 2014. Similarity-driven semantic role induction via graph partitioning. *Computational Linguistics Accepted* pages 1–62. To appear.

Liu, Yudong and Anoop Sarkar. 2007. Experimental evaluation of LTAG-based features for semantic role labeling. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic, pages 590–599.

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.

Màrquez, Lluís, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. 2008. Semantic Role Labeling: An Introduction to the Special Issue. *Computational Linguistics* 34(2):145–159.

Mazzei, Alessandro, Vincenzo Lombardo, and Patrick Sturt. 2007. Dynamic TAG and lexical dependencies. *Research on Language and Computation* 5:309–332.

Nivre, Joakim, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering* 13:95–135.

Padó, Ulrike, Matthew W. Crocker, and Frank Keller. 2009. A probabilistic model of semantic plausibility in sentence processing. *Cognitive Science* 33(5):794–838.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics* 31(1):71–106.

Pickering, Martin J., Matthew J. Traxler, and Matthew W. Crocker. 2000. Ambiguity resolution in sentence processing: Evidence against frequency-based accounts. *Journal of Memory and Language* 43(3):447–475.

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.

Sayeed, Asad and Vera Demberg. 2013. The semantic augmentation of a psycholinguistically-motivated syntactic formalism. In *Proceedings of the Fourth Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 57–65.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*.

Witten, Ian H. and Timothy C. Bell. 1991. The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. *Information Theory, IEEE Transactions on* 37(4):1085–1094.

Xia, Fei, Martha Palmer, and Aravind Joshi. 2000. A uniform method of grammar extraction and its applications. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in*

Natural Language Processing and Very Large Corpora. pages 53–62.