Joint Arc-factored Parsing of Syntactic and Semantic Dependencies

Xavier Lluís and **Xavier Carreras** and **Lluís Màrquez**
TALP Research Center
Universitat Politècnica de Catalunya
Jordi Girona 1–3, 08034 Barcelona
{xlluis,carreras,lluism}@lsi.upc.edu

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00222/1566653/tacl_a_00222.pdf by guest on 30 May 2024

Abstract

In this paper we introduce a joint arc-factored model for syntactic and semantic dependency parsing. The semantic role labeler predicts the full syntactic paths that connect predicates with their arguments. This process is framed as a linear assignment task, which allows to control some well-formedness constraints. For the syntactic part, we define a standard arc-factored dependency model that predicts the full syntactic tree. Finally, we employ dual decomposition techniques to produce consistent syntactic and predicate-argument structures while searching over a large space of syntactic configurations. In experiments on the CoNLL-2009 English benchmark we observe very competitive results.

1 Introduction

Semantic role labeling (SRL) is the task of identifying the arguments of lexical predicates in a sentence and labeling them with semantic roles (Gildea and Jurafsky, 2002; Màrquez et al., 2008). SRL is an important shallow semantic task in NLP since predicate-argument relations directly represent semantic properties of the type "who" did "what" to "whom", "how", and "why" for events expressed by predicates (typically verbs and nouns).

Predicate-argument relations are strongly related to the syntactic structure of the sentence: the majority of predicate arguments correspond to some syntactic constituent, and the syntactic structure that connects an argument with the predicate is a strong indicator of its semantic role. Actually, semantic roles represent an abstraction of the syntactic form of a predicative event. While syntactic functions of arguments change with the form of the event (e.g., active vs. passive forms), the semantic roles of arguments remain invariant to their syntactic realization.

Consequently, since the first works, SRL systems have assumed access to the syntactic structure of the sentence (Gildea and Jurafsky, 2002; Carreras and Màrquez, 2005). A simple approach is to obtain the parse trees as a pre-process to the SRL system, which allows the use of unrestricted features of the syntax. However, as in other pipeline approaches in NLP, it has been shown that the errors of the syntactic parser severely degrade the predictions of the SRL model (Gildea and Palmer, 2002). A common approach to alleviate this problem is to work with multiple alternative syntactic trees and let the SRL system optimize over any input tree or part of it (Toutanova et al., 2008; Punyakanok et al., 2008). As a step further, more recent work has proposed parsing models that predict syntactic structure augmented with semantic predicate-argument relations (Surdeanu et al., 2008; Hajič et al., 2009; Johansson, 2009; Titov et al., 2009; Lluís et al., 2009), which is the focus of this paper. These joint models should favor the syntactic structure that is most consistent with the semantic predicate-argument structures of a sentence. In principle, these models can exploit syntactic and semantic features simultaneously, and could potentially improve the accuracy for both syntactic and semantic relations.

One difficulty in the design of joint syntactic-semantic parsing models is that there exist important structural divergences between the two layers.

Transactions of the Association for Computational Linguistics, 1 (2013) 219–230. Action Editor: Brian Roark.
Submitted 1/2013; Revised 3/2013; Published 5/2013. ©2013 Association for Computational Linguistics.

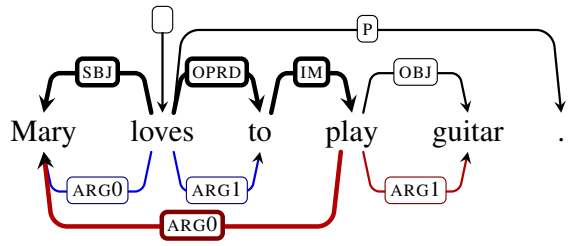

Figure 1: A sentence with syntactic dependencies (top) and semantic dependencies for the predicates "loves" and "play" (bottom). The thick arcs illustrate a structural divergence where the argument "Mary" is linked to "play" with a path involving three syntactic dependencies.

This is clearly seen in dependency-based representations of syntax and semantic roles (Surdeanu et al., 2008), such as in the example in Figure 1: the construct "loves to" causes the argument "Mary" to be syntactically distant from the predicate "play". Linguistic phenomena such as auxiliary verbs, control and raising, typically result in syntactic structures where semantic arguments are not among the direct dependants of their predicate —e.g., about 25% of arguments are *distant* in the English development set of the CoNLL-2009 shared task. Besides, standard models for dependency parsing crucially depend on arc factorizations of the dependency structure (McDonald et al., 2005; Nivre and Nilsson, 2005), otherwise their computational properties break. Hence, it is challenging to define efficient methods for syntactic and semantic dependency parsing that can exploit features of both layers simultaneously. In this paper we propose a method for this joint task.

In our method we define predicate-centric semantic models that, rather than predicting just the argument that realizes each semantic role, they predict the full syntactic path that connects the predicate with the argument. We show how efficient predictions with these models can be made using assignment algorithms in bipartite graphs. Simultaneously, we use a standard arc-factored dependency model that predicts the full syntactic tree of the sentence. Finally, we employ dual decomposition techniques (Koo et al., 2010; Rush et al., 2010; Sontag et al., 2010) to find agreement between the full dependency tree and the partial syntactic trees linking each predicate with its arguments. In summary, the main contributions of this paper are:

- We frame SRL as a weighted assignment problem in a bipartite graph. Under this framework we can control assignment constraints between roles and arguments. Key to our method, we can efficiently search over a large space of syntactic realizations of semantic arguments.
- We solve joint inference of syntactic and semantic dependencies with a dual decomposition method, similar to that of Koo et al. (2010). Our system produces consistent syntactic and predicate-argument structures while searching over a large space of syntactic configurations.

In the experimental section we compare joint and pipeline models. The final results of our joint syntactic-semantic system are competitive with the state-of-the-art and improve over the best results published by a joint method on the CoNLL-2009 English dataset.

2 A Syntactic-Semantic Dependency Model

We first describe how we represent structures of syntactic and semantic dependencies like the one in Figure 1. Throughout the paper, we will assume a fixed input sentence $\mathbf{x}$ with n tokens where lexical predicates are marked. We will also assume fixed sets of syntactic functions $\mathcal{R}_{\text{syn}}$ and semantic roles $\mathcal{R}_{\text{sem}}$. We will represent depencency structures using vectors of binary variables. A variable $y_{h,m,l}$ will indicate the presence of a syntactic dependency from head token h to dependant token m labeled with syntactic function l. Then, a syntactic tree will be denoted as a vector $\mathbf{y}$ of variables indexed by syntactic dependencies. Similarly, a variable $z_{p,a,r}$ will indicate the presence of a semantic dependency between predicate token p and argument token a labeled with semantic role r. We will represent a semantic role structure as a vector $\mathbf{z}$ indexed by semantic dependencies. Whenever we enumerate syntactic dependencies $\langle h, m, l\rangle$ we will assume that they are in the valid range for $\mathbf{x}$, i.e. $0 \leq h \leq n$, $1 \leq m \leq n$, $h \neq m$ and $l \in \mathcal{R}_{\text{syn}}$, where $h = 0$ stands for a special *root* token. Similarly, for semantic dependencies $\langle p, a, r\rangle$ we will assume that p points to a predicate of $\mathbf{x}$, $1 \leq a \leq n$ and $r \in \mathcal{R}_{\text{sem}}$.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00222/1566653/tacl_a_00222.pdf by guest on 30 May 2024

A joint model for syntactic and semantic dependency parsing could be defined as:

$$\underset{\mathbf{y},\mathbf{z}}{\text{argmax}}\ \text{s\_syn}(\mathbf{x},\mathbf{y}) + \text{s\_srl}(\mathbf{x},\mathbf{z},\mathbf{y}) \quad . \qquad (1)$$

In the equation, $\text{s\_syn}(\mathbf{x},\mathbf{y})$ gives a score for the syntactic tree $\mathbf{y}$. In the literature, it is standard to use arc-factored models defined as

$$\text{s\_syn}(\mathbf{x},\mathbf{y}) = \sum_{y_{h,m,l}=1} \text{s\_syn}(\mathbf{x},h,m,l) \quad , \qquad (2)$$

where we overload s_syn to be a function that computes scores for individual syntactic dependencies. In linear discriminative models one has $\text{s\_syn}(\mathbf{x},h,m,l) = \mathbf{w}_{\text{syn}} \cdot \mathbf{f}_{\text{syn}}(\mathbf{x},h,m,l)$, where $\mathbf{f}_{\text{syn}}$ is a feature vector for a syntactic dependency and $\mathbf{w}_{\text{syn}}$ is a vector of parameters (McDonald et al., 2005). In Section 6 we describe how we trained score functions with discriminative methods.

The other term in Eq. 1, $\text{s\_srl}(\mathbf{x},\mathbf{z},\mathbf{y})$, gives a score for a semantic dependency structure $\mathbf{z}$ using features of the syntactic structure $\mathbf{y}$. Previous work has empirically proved the importance of exploiting syntactic features in the semantic component (Gildea and Jurafsky, 2002; Xue and Palmer, 2004; Punyakanok et al., 2008). However, without further assumptions, this property makes the optimization problem computationally hard. One simple approximation is to use a pipeline model: first compute the optimal syntactic tree $\mathbf{y}$, and then optimize for the best semantic structure $\mathbf{z}$ given $\mathbf{y}$. In the rest of the paper we describe a method that searches over syntactic and semantic dependency structures jointly.

We first note that for a fixed semantic dependency, the semantic component will typically restrict the syntactic features representing the dependency to a specific subtree of $\mathbf{y}$. For example, previous work has restricted such features to the syntactic path that links a predicate with an argument (Moschitti, 2004; Johansson, 2009), and in this paper we employ this restriction. Figure 1 gives an example of a subtree, where we highlight the syntactic path that connects the semantic dependency between "play" and "Mary" with role ARG0.

Formally, for a predicate p, argument a and role r we define a *local* syntactic subtree $\boldsymbol{\pi}^{p,a,r}$ represented as a vector: $\pi^{p,a,r}_{h,m,l}$ indicates if a dependency $\langle h,m,l\rangle$ is part of the syntactic path that links predicate p with token a and role r.[1] Given full syntactic and semantic structures $\mathbf{y}$ and $\mathbf{z}$ it is trivial to construct a vector $\boldsymbol{\pi}$ that concatenates vectors $\boldsymbol{\pi}^{p,a,r}$ for all $\langle p,a,r\rangle$ in $\mathbf{z}$. The semantic model becomes

$$\text{s\_srl}(\mathbf{x},\mathbf{z},\boldsymbol{\pi}) = \sum_{z_{p,a,r}=1} \text{s\_srl}(\mathbf{x},p,a,r,\boldsymbol{\pi}^{p,a,r}) \quad , \qquad (3)$$

where s_srl computes a score for a semantic dependency $\langle p,a,r\rangle$ *together* with its syntactic path $\boldsymbol{\pi}^{p,a,r}$. As in the syntactic component, this function is typically defined as a linear function over a set of features of the semantic dependency and its path.

The inference problem of our joint model is:

$$\underset{\mathbf{y},\mathbf{z},\boldsymbol{\pi}}{\text{argmax}}\ \text{s\_syn}(\mathbf{x},\mathbf{y}) + \text{s\_srl}(\mathbf{x},\mathbf{z},\boldsymbol{\pi}) \qquad (4)$$

subject to

cTree : $\mathbf{y}$ is a valid dependency tree

cRole : $\forall p,r : \sum_a z_{p,a,r} \leq 1$

cArg : $\forall p,a : \sum_r z_{p,a,r} \leq 1$

cPath : $\forall p,a,r$: if $z_{p,a,r} = 1$ then $\boldsymbol{\pi}^{p,a,r}$ is a path from p to a, otherwise $\boldsymbol{\pi}^{p,a,r} = \mathbf{0}$

cSubtree : $\forall p,a,r : \boldsymbol{\pi}^{p,a,r}$ is a subtree of $\mathbf{y}$

Constraint **cTree** dictates that $\mathbf{y}$ is a valid dependency tree; see (Martins et al., 2009) for a detailed specification. The next two sets of constraints concern the semantic structure only. **cRole** imposes that each semantic role is realized at most once.[2] Conversely, **cArg** dictates that an argument can realize at most one semantic role in a predicate. The final two sets of constraints model the syntactic-semantic interdependencies. **cPath** imposes that each $\boldsymbol{\pi}^{p,a,r}$ represents a syntactic path between p and a whenever there exists a semantic relation. Finally, **cSubtree** imposes that the paths in $\boldsymbol{\pi}$ are consistent with the full syntactic structure, i.e. they are subtrees.

[1] In this paper we say that structures $\boldsymbol{\pi}^{p,a,r}$ are paths from predicates to arguments, but they could be more general subtrees. The condition to build a joint system is that these subtrees must be parseable in the way we describe in Section 3.1.

[2] In general a semantic role can be realized with more than one argument, though it is rare. It is not hard to modify our framework to allow for a maximum number of occurrences of a semantic role.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00222/1566653/tacl_a_00222.pdf by guest on 30 May 2024

In Section 3 we define a process that optimizes the semantic structure ignoring constraint **cSubtree**. Then in Section 4 we describe a dual decomposition method that uses the first process repeatedly to solve the joint problem.

3 SRL as Assignment

In this section we frame the problem of finding semantic dependencies as a linear assignment task. The problem we optimize is:

$$\underset{\mathbf{z},\boldsymbol{\pi}}{\text{argmax}} \ \text{s\_srl}(\mathbf{x}, \mathbf{z}, \boldsymbol{\pi}) \quad (5)$$
$$\text{subject to} \quad \textbf{cRole}, \textbf{cArg}, \textbf{cPath}$$

In this case we dropped the full syntactic structure $\mathbf{y}$ from the optimization in Eq. 4, as well as the corresponding constraints **cTree** and **cSubtree**. As a consequence, we note that the syntactic paths $\boldsymbol{\pi}$ are not tied to any consistency constraint other than each of the paths being a well-formed sequence of dependencies linking the predicate to the argument. In other words, the optimal solution in this case does not guarantee that the set of paths from a predicate to all of its arguments satisfies tree constraints. We first describe how these paths can be optimized locally. Then we show how to find a solution $\mathbf{z}$ satisfying **cRole** and **cArg** using an assignment algorithm.

3.1 Local Optimization of Syntactic Paths

Let $\widehat{\mathbf{z}}$ and $\widehat{\boldsymbol{\pi}}$ be the optimal values of Eq. 5. For any $\langle p, a, r\rangle$, let

$$\tilde{\boldsymbol{\pi}}^{p,a,r} = \underset{\boldsymbol{\pi}^{p,a,r}}{\text{argmax}} \ \text{s\_srl}(\mathbf{x}, p, a, r, \boldsymbol{\pi}^{p,a,r}) \ . \quad (6)$$

For any $\langle p, a, r\rangle$ such that $\widehat{z}_{p,a,r} = 1$ it has to be that $\widehat{\boldsymbol{\pi}}^{p,a,r} = \tilde{\boldsymbol{\pi}}^{p,a,r}$. If this was not true, replacing $\widehat{\boldsymbol{\pi}}^{p,a,r}$ with $\tilde{\boldsymbol{\pi}}^{p,a,r}$ would improve the objective of Eq. 5 without violating the constraints, thus contradicting the hypothesis about optimality of $\widehat{\boldsymbol{\pi}}$. Therefore, for each $\langle p, a, r\rangle$ we can optimize its best syntactic path locally as defined in Eq. 6.

In this paper, we will assume access to a list of likely syntactic paths for each predicate p and argument candidate a, such that the optimization in Eq. 6 can be solved explicitly by looping over each path in the list. The main advantage of this method is that, since paths are precomputed, our model can make unrestricted use of syntactic path features.

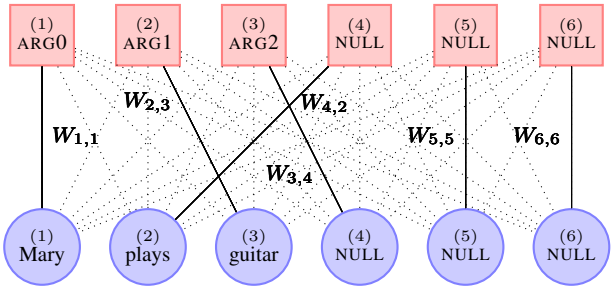

Figure 2: Illustration of the assignment graph for the sentence "Mary plays guitar", where the predicate "plays" can have up to three roles: ARG0 (agent), ARG1 (theme) and ARG2 (benefactor). Nodes labeled NULL represent a null role or token. Highlighted edges are the correct assignment.

It is simple to employ a probabilistic syntactic dependency model to create the list of likely paths for each predicate-argument pair. In the experiments we explore this approach and show that with an average of 44 paths per predicate we can recover 86.2% of the correct paths.

We leave for future work the development of efficient methods to recover the most likely syntactic structure linking an argument with its predicate.

3.2 The Assignment Algorithm

Coming back to solving Eq. 5, it is easy to see that an optimal solution satisfying constraints **cRole** and **cArg** can be found with a linear assignment algorithm. The process we describe determines the predicate-argument relations separately for each predicate. Assume a bipartite graph of size N with *role* nodes $r_1 \dots r_N$ on one side and *argument* nodes $a_1 \dots a_N$ on the other side. Assume also a matrix of non-negative scores $W_{i,j}$ corresponding to assigning argument a_j to role r_i. A linear assignment algorithm finds a bijection $f : i \to j$ from roles to arguments that maximizes $\sum_{i=1}^{N} W_{i,f(i)}$. The Hungarian algorithm finds the exact solution to this problem in $\mathcal{O}(N^3)$ time (Kuhn, 1955; Burkard et al., 2009).

All that is left is to construct a bipartite graph representing predicate roles and sentence tokens, such that some roles and tokens can be left unassigned, which is a common setting for assignment tasks. Algorithm 1 describes a procedure for constructing a weighted bipartite graph for SRL, and Figure 2 illustrates an example of a bipartite graph. We then

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00222/1566653/tacl_a_00222.pdf by guest on 30 May 2024

Algorithm 1 Construction of an Assignment Graph for Semantic Role Labeling

Let p be a predicate with k possible roles. Let n be the number of argument candidates in the sentence. This algorithm creates a bipartite graph with $N = n+k$ vertices on each side.

1. Create *role* vertices r_i for $i = 1 \dots N$, where
   - for $1 \leq i \leq k$, r_i is the i-th role,
   - for $1 \leq i \leq n$, r_{k+i} is a special NULL role.
2. Create *argument* vertices a_j for $j = 1 \dots N$, where
   - for $1 \leq j \leq n$, a_j is the j-th argument candidate,
   - for $1 \leq j \leq k$, a_{n+j} is a special NULL argument.
3. Define a matrix of model scores $S \in \mathbb{R}^{(k+1)\times n}$:
   (a) Optimization of syntactic paths:
   For $1 \leq i \leq k, 1 \leq j \leq n$
   $S_{i,j} = \max_{\boldsymbol{\pi}^{p,a_j,r_i}} \text{s\_srl}(\mathbf{x}, p, a_j, r_i, \boldsymbol{\pi}^{p,a_j,r_i})$
   (b) Scores of NULL assignments[3]:
   For $1 \leq j \leq n$
   $S_{k+1,j} = 0$
4. Let $S_0 = \min_{i,j} S_{i,j}$, the minimum of any score in S. Define a matrix of *non-negative* scores $W \in \mathbb{R}^{N\times N}$ as follows:
   (a) for $1 \leq i \leq k,\ 1 \leq j \leq n$
   $W_{i,j} = S_{i,j} - S_0$
   (b) for $k < i \leq N,\ 1 \leq j \leq n$
   $W_{i,j} = S_{k+1,j} - S_0$
   (c) for $1 < i \leq N,\ n < j \leq N$
   $W_{i,j} = 0$

run the Hungarian algorithm on the weighted graph and obtain a bijection $f : r_i \rightarrow a_j$, from which it is trivial to recover the optimal solution of Eq. 5. Finally, we note that it is simple to allow for multiple instances of a semantic role by adding more role nodes in step 1; it would be straightforward to add penalties in step 3 for multiple instances of roles.

4 A Dual Decomposition Algorithm

We now present a dual decomposition method to optimize Eq. 4, that uses the assignment algorithm presented above as a subroutine. Our method is similar to that of Koo et al. (2010), in the sense that our joint optimization can be decomposed into two sub-problems that need to agree on the syntactic dependencies they predict. For a detailed description of dual decomposition methods applied to NLP see (Sontag et al., 2010; Rush et al., 2010).

We note that in Eq. 4 the constraint **cSubtree** ties the syntactic and semantic structures, imposing that any path $\boldsymbol{\pi}^{p,a,r}$ that links a predicate p with an argument a must be a subtree of the full syntactic structure $\mathbf{y}$. Formally the set of constraints is:

$$y_{h,m,l} \geq \pi^{p,a,r}_{h,m,l} \quad \forall\, p, a, r, h, m, l \ .$$

These constraints can be compactly written as

$$c \cdot y_{h,m,l} \geq \sum_{p,a,r} \pi^{p,a,r}_{h,m,l} \quad \forall\, h, m, l \ ,$$

where c is a constant equal to the number of distinct semantic dependencies $\langle p, a, r\rangle$. In addition, we can introduce a vector non-negative slack variables $\boldsymbol{\xi}$ with a component for each syntactic dependency $\xi_{h,m,l}$, turning the constraints into:

$$c \cdot y_{h,m,l} - \sum_{p,a,r} \pi^{p,a,r}_{h,m,l} - \xi_{h,m,l} = 0 \quad \forall\, h, m, l$$

We can now rewrite Eq. 4 as:

$$\begin{aligned} &\underset{\mathbf{y},\mathbf{z},\boldsymbol{\pi},\boldsymbol{\xi}\geq\mathbf{0}}{\text{argmax}} \quad \text{s\_syn}(\mathbf{x}, \mathbf{y}) + \text{s\_srl}(\mathbf{x}, \mathbf{z}, \boldsymbol{\pi}) \qquad (7)\\ &\quad \text{subject to}\\ &\quad \textbf{cTree}, \textbf{cRole}, \textbf{cArg}, \textbf{cPath}\\ &\quad \forall h, m, l : c \cdot y_{h,m,l} - \sum_{p,a,r} \pi^{p,a,r}_{h,m,l} - \xi_{h,m,l} = 0 \end{aligned}$$

As in Koo et al. (2010), we will relax subtree constraints by introducing a vector of Lagrange multipliers $\boldsymbol{\lambda}$ indexed by syntactic dependencies, i.e. each coordinate $\lambda_{h,m,l}$ is a Lagrange multiplier for the constraint associated with $\langle h, m, l\rangle$. The Lagrangian of the problem is:

$$\begin{aligned} L(\mathbf{y}, \mathbf{z}, \boldsymbol{\pi}, \boldsymbol{\xi}, \boldsymbol{\lambda}) = &\ \text{s\_syn}(\mathbf{x}, \mathbf{y}) + \text{s\_srl}(\mathbf{x}, \mathbf{z}, \boldsymbol{\pi})\\ &+ \boldsymbol{\lambda} \cdot \left(c \cdot \mathbf{y} - \sum_{p,a,r} \boldsymbol{\pi}^{p,a,r} - \boldsymbol{\xi} \right) \qquad (8) \end{aligned}$$

We can now formulate Eq. 7 as:

$$\begin{aligned} &\max_{\substack{\mathbf{y},\mathbf{z},\boldsymbol{\pi},\boldsymbol{\xi}\geq\mathbf{0}\\ \text{s.t. } \textbf{cTree},\textbf{cRole},\textbf{cArg},\textbf{cPath}\\ c\cdot\mathbf{y}-\sum_{p,a,r}\boldsymbol{\pi}^{p,a,r}-\boldsymbol{\xi}=\mathbf{0}}} L(\mathbf{y}, \mathbf{z}, \boldsymbol{\pi}, \boldsymbol{\xi}, \boldsymbol{\lambda}) \qquad (9) \end{aligned}$$

[3] In our model we fix the score of null assignments to 0. It is straightforward to compute a discriminative score instead.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00222/1566653/tacl_a_00222.pdf by guest on 30 May 2024

This optimization problem has the property that its optimum value is the same as the optimum of Eq. 7 for any value of $\boldsymbol{\lambda}$. This is because whenever the constraints are satisfied, the terms in the Lagrangian involving $\boldsymbol{\lambda}$ are zero. If we remove the subtree constraints from Eq. 9 we obtain the dual objective:

$$D(\boldsymbol{\lambda}) = \max_{\substack{\mathbf{y},\mathbf{z},\boldsymbol{\pi},\boldsymbol{\xi}\geq\mathbf{0} \\ \text{s.t. } \mathbf{cTree},\mathbf{cRole},\mathbf{cArg},\mathbf{cPath}}} L(\mathbf{y},\mathbf{z},\boldsymbol{\pi},\boldsymbol{\xi},\boldsymbol{\lambda}) \quad (10)$$

$$= \max_{\mathbf{y} \text{ s.t. } \mathbf{cTree}} \left(\text{s\_syn}(\mathbf{x},\mathbf{y}) + c \cdot \mathbf{y} \cdot \boldsymbol{\lambda}\right) + \max_{\substack{\mathbf{z},\boldsymbol{\pi} \\ \text{s.t. } \mathbf{cRole},\mathbf{cArg},\mathbf{cPath}}} \left(\text{s\_srl}(\mathbf{x},\mathbf{z},\boldsymbol{\pi}) - \boldsymbol{\lambda} \cdot \sum_{p,a,r} \boldsymbol{\pi}^{p,a,r}\right) + \max_{\boldsymbol{\xi}\geq\mathbf{0}} \left(-\boldsymbol{\lambda} \cdot \boldsymbol{\xi}\right) \quad (11)$$

The dual objective is an upper bound to the optimal value of primal objective of Eq. 7. Thus, we are interested in finding the minimum of the dual in order to tighten the upper-bound. We will solve

$$\min_{\boldsymbol{\lambda}} D(\boldsymbol{\lambda}) \quad (12)$$

using a subgradient method. Algorithm 2 presents pseudo-code. The algorithm takes advantage of the *decomposed* form of the dual in Eq. 11, where we have rewritten the Lagrangian such that syntactic and semantic structures appear in separate terms. This will allow to compute subgradients efficiently. In particular, the subgradient of D at a point $\boldsymbol{\lambda}$ is:

$$\Delta(\boldsymbol{\lambda}) = c \cdot \widehat{\mathbf{y}} - \sum_{p,a,r} \widehat{\boldsymbol{\pi}}^{p,a,r} - \widehat{\boldsymbol{\xi}} \quad (13)$$

where

$$\widehat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y} \text{ s.t. } \mathbf{cTree}} \left(\text{s\_syn}(\mathbf{x},\mathbf{y}) + c \cdot \mathbf{y} \cdot \boldsymbol{\lambda}\right) \quad (14)$$

$$\widehat{\mathbf{z}},\widehat{\boldsymbol{\pi}} = \operatorname*{argmax}_{\substack{\mathbf{z},\boldsymbol{\pi} \text{ s.t.} \\ \mathbf{cRole},\mathbf{cArg},\mathbf{cPath}}} \text{s\_srl}(\mathbf{x},\mathbf{z},\boldsymbol{\pi}) - \boldsymbol{\lambda} \cdot \sum_{p,a,r} \boldsymbol{\pi}^{p,a,r} \quad (15)$$

$$\widehat{\boldsymbol{\xi}} = \operatorname*{argmax}_{\boldsymbol{\xi}\geq\mathbf{0}} -\boldsymbol{\lambda} \cdot \boldsymbol{\xi} \quad (16)$$

Whenever $\widehat{\boldsymbol{\pi}}$ is consistent with $\widehat{\mathbf{y}}$ the subgradient will be zero and the method will converge. When paths $\widehat{\boldsymbol{\pi}}$ contain a dependency $\langle h,m,l\rangle$ that is inconsistent with $\widehat{\mathbf{y}}$, the associated dual $\lambda_{h,m,l}$ will increase, hence lowering the score of all paths that use $\langle h,m,l\rangle$ at the next iteration; at same time, the total score for that dependency will increase, favoring syntactic dependency structures alternative to $\widehat{\mathbf{y}}$. As

Algorithm 2 A dual-decomposition algorithm for syntactic-semantic dependency parsing

Input: $\mathbf{x}$, a sentence; T, number of iterations;
Output: syntactic and semantic structures $\widehat{\mathbf{y}}$ and $\widehat{\mathbf{z}}$
Notation: we use **cSem**= **cRole** $\wedge$ **cArg** $\wedge$ **cPath**

$\boldsymbol{\lambda}^1 = \mathbf{0}$ # initialize dual variables
c =number of distinct $\langle h,m,l\rangle$ in $\mathbf{x}$
for $t = 1 \ldots T$ **do**
  $\widehat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y} \text{ s.t. } \mathbf{cTree}} \left(\text{s\_syn}(\mathbf{x},\mathbf{y}) + c \cdot \boldsymbol{\lambda}^t \cdot \mathbf{y}\right)$
  $\widehat{\mathbf{z}},\widehat{\boldsymbol{\pi}} = \operatorname{argmax}_{\substack{\mathbf{z},\boldsymbol{\pi} \\ \text{s.t. } \mathbf{cSem}}} \left(\text{s\_srl}(\mathbf{x},\mathbf{z},\boldsymbol{\pi}) - \boldsymbol{\lambda}^t \cdot \sum_{p,a,r} \boldsymbol{\pi}^{p,a,r}\right)$
  $\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t$ # dual variables for the next iteration
  Set α_t, the step size of the current iteration
  for each $\langle h,m,l\rangle$ **do**
    $q = \sum_{p,a,r} \widehat{\pi}^{p,a,r}_{h,m,l}$ # num. paths using $\langle h,m,l\rangle$
    if $q > 0$ **and** $\widehat{y}_{h,m,l} = 0$ **then**
      $\lambda^{t+1}_{h,m,l} = \lambda^{t+1}_{h,m,l} + \alpha_t q$
  break if $\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t$ # convergence
return $\widehat{\mathbf{y}},\widehat{\mathbf{z}}$

in previous work, in the algorithm a parameter α_t controls the size of subgradient steps at iteration t.

The key point of the method is that solutions to Eq. 14 and 15 can be computed efficiently using separate processes. In particular, Eq. 14 corresponds to a standard dependency parsing problem, where for each dependency $\langle h,m,l\rangle$ we have an additional score term $c \cdot \lambda_{h,m,l}$ —in our experiments we use the projected dependency parsing algorithm by (Eisner, 2000). To calculate Eq. 15 we use the assignment method described in Section 3, where it is straightforward to introduce additional score terms $-\lambda_{h,m,l}$ to every factor $\pi^{p,a,r}_{h,m,l}$. It can be shown that whenever the subgradient method converges, the solutions $\widehat{\mathbf{y}}$ and $\widehat{\mathbf{z}}$ are the optimal solutions to our original problem in Eq. 4 (see (Koo et al., 2010) for a justification). In practice we run the subgradient method for a maximum number of iterations, and return the solutions of the last iteration if it does not converge.

5 Related Work

Recently, there have been a number of approaches to joint parsing of syntactic and semantic dependencies, partly because of the availability of treebanks in this format popularized by the CoNLL shared tasks (Surdeanu et al., 2008; Hajič et al., 2009).

Like in our method, Johansson (2009) defined a model that exploits features of a semantic depen-

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00222/1566653/tacl_a_00222.pdf by guest on 30 May 2024

dency together with the syntactic path connecting the predicate and the argument. That method uses an approximate parsing algorithm that employs k-best inference and beam search. Similarly, Lluís et al. (2009) defined a joint model that forces the predicate structure to be represented in the syntactic dependency tree, by enriching arcs with semantic information. The semantic component uses features of pre-computed syntactic structures that may diverge from the joint structure. In contrast, our joint parsing method is exact whenever the dual decomposition algorithm converges.

Titov et al. (2009) augmented a transition-based dependency parser with operations that produce synchronous derivations of syntactic and semantic structures. Instead of explicitly representing semantic dependencies together with a syntactic path, they induce latent representations of the interactions between syntactic and semantic layers.

In all works mentioned the model has no control of assignment constraints that disallow labeling multiple arguments with the same semantic role. Punyakanok et al. (2008) first introduced a system that explicitly controls these constraints, as well as other constraints that look at pairwise assignments which we can not model. They solve SRL using general-purpose Integer Linear Programming (ILP) methods. In similar spirit, Riedel and McCallum (2011) presented a model for extracting structured events that controls interactions between predicate-argument assignments. They take into account pairwise assignments and solve the optimization problem with dual decomposition. More recently, Das et al. (2012) proposed a dual decomposition method that deals with several assignment constraints for predicate-argument relations. Their method is an alternative to general ILP methods. To our knowledge, our work is the first that frames SRL as a linear assignment task, for which simple and exact algorithms exist. We should note that these works model predicate-argument relations with assignment constraints, but none of them predicts the underlying syntactic structure.

Our dual decomposition method follows from that of Koo et al. (2010). In both cases two separate processes predict syntactic dependency structures, and the dual decomposition algorithm seeks agreement at the level of individual dependencies. One difference is that our semantic process predicts partial syntax (restricted to syntactic paths connecting predicates and arguments), while in their case each of the two processes predicts the full set of dependencies.

6 Experiments

We present experiments using our syntactic-semantic parser on the CoNLL-2009 Shared Task English benchmark (Hajič et al., 2009). It consists of the usual WSJ training/development/test sections mapped to dependency trees, augmented with semantic predicate-argument relations from PropBank (Palmer et al., 2005) and NomBank (Meyers et al., 2004) also represented as dependencies. It also contains a PropBanked portion of the Brown corpus as an out-of-domain test set.

Our goal was to evaluate the contributions of parsing algorithms in the following configurations:

Base Pipeline Runs a syntactic parser and then runs an SRL parser constrained to paths of the best syntactic tree. In the SRL it only enforces constraint **cArg**, by simply classifying the candidate argument in each path into one of the possible semantic roles or as NULL.

Pipeline with Assignment Runs the assignment algorithm for SRL, enforcing constraints **cRole** and **cArg**, but constrained to paths of the best syntactic tree.

Forest Runs the assignment algorithm for SRL on a large set of precomputed syntactic paths, described below. This configuration corresponds to running Dual Decomposition for a single iteration, and is not guaranteed to predict consistent syntactic and semantic structures.

Dual Decomposition (DD) Runs dual decomposition using the assignment algorithm on the set of precomputed paths. Syntactic and semantic structures are consistent when it reaches convergence.

All four systems used the same type of discriminative scorers and features. Next we provide details about these systems. Then we present the results.

6.1 Implementation

Syntactic model We used two discriminative arc-factored models for labeled dependency parsing: a

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00222/1566653/tacl_a_00222.pdf by guest on 30 May 2024

first-order model, and a second-order model with grandchildren interactions, both reimplementations of the parsers by McDonald et al. (2005) and Carreras (2007) respectively. In both cases we used projective dependency parsing algorithms based on (Eisner, 2000).[4] To learn the models, we used a log-linear loss function following Koo et al. (2007), which trains probabilistic discriminative parsers. At test time, we used the probabilistic parsers to compute marginal probabilities $p(h, m, l \mid \mathbf{x})$, using inside-outside algorithms for first/second-order models. Hence, for either of the parsing models, we always obtain a table of first-order marginal scores, with one score per labeled dependency. Then we run first-order inference with these marginals to obtain the best tree. We found that the higher-order parser performed equally well on development using this method as using second-order inference to predict trees: since we run the parser multiple times within *Dual Decomposition*, our strategy results in faster parsing times.

Precomputed Paths Both *Forest* and *Dual Decomposition* run assignment on a set of precomputed paths, and here we explain how we build it. We first observed that 98.4% of the correct arguments in development data are either direct descendants of the predicate, direct descendants of an ancestor of the predicate, or an ancestor of the predicate.[5] All methods we test are restricted to this syntactic scope. To generate a list of paths, we did as follows:

- Calculate marginals of *unlabeled* dependencies using the first-order parser: $p(h, m \mid \mathbf{x}) = \sum_l p(h, m, l \mid \mathbf{x})$. Note that for each m, the probabilities $p(h, m|\mathbf{x})$ for all h form a distribution (i.e. they sum to one). Then, for each m, keep the most-likely dependencies that cover at least 90% of the mass, and prune the rest.
- Starting from a predicate p, generate a path by taking any number of dependencies that ascend, and optionally adding one dependency that descends. We constrained paths to be projective, and to have a maximum number of 6 ascendant dependencies.
- Label each unlabeled edge $\langle h, m \rangle$ in the paths with $l = \text{argmax}_l \, p(h, m, l \mid \mathbf{x})$.

On development data, this procedure generated an average of 43.8 paths per predicate that cover 86.2% of the correct paths. In contrast, enumerating paths of the single-best tree covers 79.4% of correct paths for the first-order parser, and 82.2% for the second-order parser.[6]

SRL model We used a discriminative model with similar features to those in the system of Johansson (2009). In addition, we included the following:

- Unigram/bigram/trigram path features. For all n-grams in the syntactic path, patterns of words and POS tags (e.g., mary+loves+to, mary+VB+to).
- Voice features. The predicate voice together with the word/POS of the argument (e.g., passive+mary).
- Path continuity. Count of non-consecutive tokens in a predicate-argument path.

To train SRL models we used the averaged perceptron (Collins, 2002). For the base pipeline we trained standard SRL classifiers. For the rest of models we used the structured Perceptron running the assignment algorithm as inference routine. In this case, we generated a large set of syntactic paths for training using the procedure described above, and we set the loss function to penalize mistakes in predicting the semantic role of arguments and their syntactic path.

Dual Decomposition We added a parameter β weighting the syntactic and semantic components of the model as follows:

$$(1 - \beta)\, \text{s\_syn}(\mathbf{x}, \mathbf{y}) + \beta\, \text{s\_srl}(\mathbf{x}, \mathbf{z}, \boldsymbol{\pi}) \quad .$$

As syntactic scores we used normalized marginal probabilities of dependencies, either from the first or the higher-order parser. The scores of all factors of the SRL model were normalized at every sentence to be between -1 and 1. The rest of details

[4] Our method allows to use non-projective dependency parsing methods seamlessly.

[5] This is specific to CoNLL-2009 data for English. In general, for other languages the coverage of these rules may be lower. We leave this question to future work.

[6] One can evaluate the maximum recall on correct arguments that can be obtained, irrespective of whether the syntactic path is correct: for the set of paths it is 98.3%, while for single-best trees it is 91.9% and 92.7% for first and second-order models.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00222/1566653/tacl_a_00222.pdf by guest on 30 May 2024

	o	LAS	UAS	sem_p	sem_r	sem_{F_1}	sem_pp
Pipeline	1	85.32	88.86	**86.23**	67.67	75.83	45.64
w. Assig.	1	85.32	88.86	84.08	71.82	**77.47**	51.17
Forest	-	-	-	80.67	**73.60**	76.97	**51.33**
Pipeline	2	87.77	90.96	**87.07**	68.65	76.77	47.07
w. Assig.	2	87.77	90.96	85.21	73.41	**78.87**	**53.80**

Table 1: Results on development for the baseline and assignment pipelines, running first and second-order syntactic parsers, and the Forest method. o indicates the order of syntactic inference.

of the method were implemented following Koo et al. (2010), including the strategy for decreasing the step size α_t. We ran the algorithm for up to 500 iterations, with initial step size of 0.001.

6.2 Results

To evaluate syntactic dependencies we use *unlabeled attachment score* (UAS), i.e., the percentage of words with the correct head, and *labeled attachment scores* (LAS), i.e., the percentage of words with the correct head and syntactic label. Semantic predicate-argument relations are evaluated with precision (sem_p), recall (sem_r) and F_1 measure (sem_{F_1}) at the level of labeled semantic dependencies. In addition, we measure the percentage of perfectly predicted predicate structures (sem_{pp}).[7]

Table 1 shows the results on the development set for our three first methods. We can see that the pipeline methods running assignment improve over the baseline pipelines in semantic F_1 by about 2 points, due to the application of the **cRole** constraint. The *Forest* method also shows an improvement in recall of semantic roles with respect to the pipeline methods. Presumably, the set of paths available in the *Forest* model allows to recognize a higher number of arguments at an expense of a lower precision. Regarding the percentage of perfect predicate-argument structures there is a remarkable improvement in the systems that apply the full set of constraints using the assignment algorithm. We believe that the **cRole** constraint that ensures no repeated roles for a given predicate is a key factor to predict the full set of arguments of a predicate.

[7] Our evaluation metrics differ slightly from the official metric at CoNLL-2009. That metric considers predicate senses as special semantic dependencies and, thus, it includes them in the calculation of the evaluation metrics. In this paper, we are not addressing predicate sense disambiguation and, consequently, we ignore predicate senses when presenting evaluation results. When we report the performance of CoNLL systems, their scores will be noticeably lower than the scores reported at the shared task. This is because predicate disambiguation is a reasonably simple task with a very high baseline around 90%.

o	β	LAS	UAS	sem_p	sem_r	sem_{F_1}	sem_pp	%conv
1	0.1	85.32	88.86	84.09	71.84	77.48	51.77	100
1	0.4	85.36	88.91	84.07	71.94	77.53	51.85	100
1	0.5	85.38	88.93	84.08	72.03	77.59	51.96	100
1	0.6	85.41	88.95	84.05	72.19	77.67	52.03	99.8
1	0.7	85.44	89.00	84.10	72.42	77.82	52.24	99.7
1	0.8	**85.48**	**89.02**	**83.99**	72.69	**77.94**	**52.57**	99.5
1	0.9	85.39	88.93	83.68	**72.82**	77.88	52.49	99.8
2	0.1	87.78	90.96	85.20	73.11	78.69	53.74	100
2	0.4	87.78	90.96	**85.21**	73.12	78.70	53.74	100
2	0.5	87.78	90.96	85.19	73.12	78.70	53.72	100
2	0.6	87.78	90.96	85.20	73.13	78.70	53.72	99.9
2	0.7	87.78	90.96	85.19	73.13	78.70	53.72	99.8
2	0.8	87.80	90.98	85.20	73.18	78.74	53.77	99.8
2	0.9	**87.84**	**91.02**	85.20	**73.23**	**78.76**	**53.82**	100

Table 2: Results of the dual decomposition method on development data, for different values of the β parameter. o is the order of the syntactic parser. %conv is the percentage of examples that converged.

The *Forest* configuration is the starting point to run the dual decomposition algorithm. We ran experiments for various values of the β parameter. Table 2 shows the results. We see that as we increase β, the SRL component has more relative weight, and the syntactic structure changes. The *DD* methods are always able to improve over the *Forest* methods, and find convergence in more than 99.5% of sentences. Compared to the pipeline running assignment, *DD* improves semantic F_1 for first-order inference, but not for higher-order inference, suggesting that 2^{nd} order predictions of paths are quite accurate. We also observe slight benefits in syntactic accuracy.

Table 3 presents results of our system on the test sets, where we run *Pipeline with Assignment* and *Dual Decomposition* with our best configuration ($\beta = 0.8/0.9$ for $1^{\text{st}}/2^{\text{nd}}$ order syntax). For comparison, the table also reports the results of the best CoNLL–2009 joint system, *Merlo09* (Gesmundo et al., 2009), which proved to be very competitive ranking third in the closed challenge. We also include *Lluís09* (Lluís et al., 2009), which is another joint syntactic-semantic system from CoNLL–2009.[8] In the WSJ test *DD* obtains the best syntactic accuracies, while the *Pipeline* obtains the best se-

[8] Another system to compare to is the joint system by Johansson (2009). Unfortunately, a direct comparison is not possible because it is evaluated on the CoNLL-2008 datasets, which

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00222/1566653/tacl_a_00222.pdf by guest on 30 May 2024

WSJ	LAS	UAS	sem_p	sem_r	sem_{F_1}	sem_pp
Lluís09	87.48	89.91	73.87	67.40	70.49	39.68
Merlo09	88.79	91.26	81.00	**76.45**	78.66	54.80
Pipe-Assig 1st	86.85	89.68	85.12	73.78	79.05	54.12
DD 1st	87.04	89.89	85.03	74.56	79.45	54.92
Pipe-Assig 2nd	89.19	91.62	**86.11**	75.16	**80.26**	**55.96**
DD 2nd	**89.21**	**91.64**	86.01	74.84	80.04	55.73

Brown	LAS	UAS	sem_p	sem_r	sem_{F_1}	sem_pp
Lluís09	80.92	85.96	62.29	59.22	60.71	29.79
Merlo09	80.84	86.32	68.97	**63.06**	65.89	38.92
Pipe-Assig 1st	80.96	86.58	72.91	60.16	65.93	38.44
DD 1st	81.18	86.86	72.53	60.76	66.12	38.13
Pipe-Assig 2nd	82.56	87.98	73.94	61.63	67.23	**38.99**
DD 2nd	**82.61**	**88.04**	**74.12**	61.59	**67.28**	38.92

Table 3: Comparative results on the CoNLL–2009 English test sets, namely the WSJ test (top table) and the out of domain test from the Brown corpus (bottom table).

mantic F_1. The bottom part of Table 3 presents results on the out-of-domain Brown test corpus. In this case, *DD* obtains slightly better results than the rest, both in terms of syntactic accuracy and semantic F_1.

Table 4 shows statistical significance tests for the syntactic LAS and semantic F_1 scores of Table 3. We have applied the sign test (Wackerly et al., 2007) and approximate randomization tests (Yeh, 2000) to all pairs of systems outputs. The differences between systems in the WSJ test can be considered significant in almost all cases with $p = 0.05$. In the Brown test set, results are more unstable and differences are not significant in general, probably because of the relatively small size of that test.

Regarding running times, our implementation of the baseline pipeline with 2nd order inference parses the development set (1,334 sentences) in less than 7 minutes. Running assignment in the pipeline increases parsing time by ∼8% due to the overhead from the assignment algorithm. The *Forest* method, with an average of 61.3 paths per predicate, is ∼13% slower than the pipeline due to the exploration of the space of precomputed paths. Finally, *Dual Decomposition* with 2nd order inference converges in 36.6 iterations per sentence on average. The first iteration of *DD* has to perform roughly the same work as *Forest*, while subsequent iterations only need to re-parse the sentence with respect to the dual updates, which are extremely sparse. Our current implementation did not take advantage of the sparsity of updates, and overall, *DD* was on average 13 times slower than the pipeline running assignment and 15 times slower than the baseline pipeline.

	WSJ					**Brown**				
	ME	PA1	DD1	PA2	DD2	ME	PA1	DD1	PA2	DD2
LL	○●□■	○●□■	●□■	○●□■	○●□■	□■	□■	□■	○●□■	○●□■
ME		○●	○●□■	○●□■	○●□■				●	●
PA1			○●□■	○●□■	○●□■			●	○●■	○●□■
DD1				○●□■	○●□■				○●■	○●□■
PA2					●□■					

Table 4: Statistical tests of significance for LAS and sem_{F_1} differences between pairs of systems from Table 3. ○/● = LAS difference is significant by the sign/ approximate randomization tests at 0.05 level. □/■ = same meaning for sem_{F_1}. The legend for systems is: LL: *Lluís09*, ME: *Merlo09*, PA1/2: *Pipeline with Assignment, 1st/2nd order*, DD1/2: *Dual Decomposition, 1st/2nd order*.

7 Conclusion

We have introduced efficient methods to parse syntactic dependency structures augmented with predicate-argument relations, with two key ideas. One is to predict the local syntactic structure that links a predicate with its arguments, and seek agreement with the full syntactic structure using dual decomposition techniques. The second is to control linear assignment constraints in the predicate-argument structure.

In experiments we observe large improvements resulting from the assignment constraints. As for the dual decomposition technique for joint parsing, it does improve over the pipelines when we use a first order parser. This means that in this configuration the explicit semantic features help to find a solution that is better in both layers. To some extent, this empirically validates the research objective of joint models. However, when we move to second-order parsers the differences with respect to the pipeline are insignificant. It is to be expected that as syntactic parsers improve, the need of joint methods is less critical. It remains an open question to validate if large improvements can be achieved by integrating syntactic-semantic features. To study this question, it is necessary to have efficient parsing algorithms for joint dependency structures. This paper contributes with a method that has optimality

are slightly different. However, note that *Merlo09* is an application of the system by Titov et al. (2009). In that paper authors report results on the CoNLL-2008 datasets, and they are comparable to Johansson's.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00222/1566653/tacl_a_00222.pdf by guest on 30 May 2024

guarantees whenever it converges.

Our method can incorporate richer families of features. It is straightforward to incorporate better semantic representations of predicates and arguments than just plain words, e.g. by exploiting WordNet or distributional representations as in (Zapirain et al., 2013). Potentially, this could result in larger improvements in the performance of syntactic and semantic parsing.

It is also necessary to experiment with different languages, where the performance of syntactic parsers is lower than in English, and hence there is potential for improvement. Our treatment of local syntactic structure that links predicates with arguments, based on explicit enumeration of likely paths, was simplistic. Future work should explore methods that model the syntactic structure linking predicates with arguments: whenever this structure can be parsed efficiently, our dual decomposition algorithm can be employed to define an efficient joint system.

Acknowledgments

We thank the editor and the anonymous reviewers for their valuable feedback. This work was financed by the European Commission for the XLike project (FP7-288342); and by the Spanish Government for project OpenMT-2 (TIN2009-14675-C03-01), project Skater (TIN2012-38584-C06-01), and a Ramón y Cajal contract for Xavier Carreras (RYC-2008-02223).

References

Rainer Burkard, Mario Dell'Amico, and Silvano Martello. 2009. *Assignment Problems*. Society for Industrial and Applied Mathematics.

Xavier Carreras 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 (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan, June.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961, Prague, Czech Republic, June.

Michael Collins. 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, July.

Dipanjan Das, André F. T. Martins, and Noah A. Smith. 2012. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 209–217, Stroudsburg, PA, USA.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In Harry Bunt and Anton Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers, October.

Andrea Gesmundo, 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 (CoNLL 2009): Shared Task*, pages 37–42, Boulder, Colorado, June.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, September.

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 239–246, Philadelphia, Pennsylvania, USA, July.

Jan Hajič, 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): Shared Task*, pages 1–18, Boulder, Colorado, USA, June.

Richard Johansson. 2009. Statistical bistratal dependency parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 561–569, Singapore, August.

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, Prague, Czech Republic, June.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298, Cambridge, MA, October.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00222/1566653/tacl_a_00222.pdf by guest on 30 May 2024

Harold W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.

Xavier Lluís, Stefan Bott, and Lluís Màrquez. 2009. A second-order joint eisner model for syntactic and semantic dependency parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 79–84, Boulder, Colorado, June.

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

André Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 342–350, Suntec, Singapore, August.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98, Ann Arbor, Michigan, June.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank Project: An interim report. In A. Meyers, editor, *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 24–31, Boston, Massachusetts, USA, May.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow statistic parsing. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 335–342, Barcelona, Spain, July.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 99–106, Ann Arbor, Michigan, June.

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

Vasin Punyakanok, Dan Roth, and Wen tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(3):257–287, June.

Sebastian Riedel and Andrew McCallum. 2011. Fast and robust joint models for biomedical event extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1–12, Edinburgh, Scotland, UK., July.

Alexander M Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Cambridge, MA, October.

David Sontag, Amir Globerson, and Tommi Jaakkola. 2010. Introduction to dual decomposition for inference. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*. MIT Press.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England, August.

Ivan Titov, 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*, IJCAI'09, pages 1562–1567.

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

Dennis D. Wackerly, William Mendenhall, and Richard L. Scheaffer, 2007. *Mathematical Statistics with Applications*, chapter 15: Nonparametric statistics. Duxbury Press.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 88–94, Barcelona, Spain, July.

Alexander S. Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics*, pages 947–953.

Beñat Zapirain, Eneko Agirre, Lluís Màrquez, and Mihai Surdeanu. 2013. Selectional preferences for semantic role classification. *Computational Linguistics*, 39(3).

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00222/1566653/tacl_a_00222.pdf by guest on 30 May 2024