Universitat Politècnica de Catalunya

Ph.D. Thesis

Joint Parsing of Syntactic and Semantic Dependencies

Xavier Lluís

advisors
Doctor Xavier Carreras
Doctor Lluís Màrquez

Departament de Ciències de la Computació
Programa de Doctorat en Intel·ligència Artificial

CONTENTS

Agraïments

No hauria pogut escriure aquesta tesi sense el suport i l'ajuda dels meus directors de tesi, la meva família, amics i companys de despatx. Gràcies a tots!

Acknowledgments

This work has been partially funded by the European Commission for the XLike project (FP7-288342); and by the Spanish Government for project KNOW (TIN2006-15049-C03), project OpenMT-2 (TIN2009-14675-C03-01) and project Skater (TIN2012-38584-C06-01).

Abstract

Syntactic Dependency Parsing and Semantic Role Labeling (SRL) are two main problems in Natural Language Understanding. Both tasks are closely related and can be regarded as parsing on top of a given sequence. In the data-driven approach context, these tasks are typically addressed sequentially by a pipeline of classifiers. A syntactic parser is run in the first stage, and then given the predicates, the semantic roles are identified and classified (Gildea and Jurafsky, 2002).

An appealing and largely unexplored idea is to jointly process syntactic dependencies and semantic roles. A joint process could capture some interactions that pipeline systems are unable to model. We expect joint models to improve on syntax based on semantic cues and also the reverse. Despite this potential advantage and the interest in joint processing stimulated by the CoNLL-2008 and 2009 Shared Tasks (Surdeanu et al., 2008; Hajič et al., 2009), very few joint models have been proposed to date, few have achieved attention and fewer have obtained competitive results.

This thesis presents three contributions on this topic. The first contribution is to frame semantic role labeling as a linear assignment task. Under this framework we avoid assigning repeated roles to the arguments of a predicate. Our proposal follows previous work on enforcing constraints on the SRL analysis (Punyakanok et al., 2004; Surdeanu et al., 2007). But in our case, we enforce only a relevant subset of these constraints. We solve this problem with the efficient $O(n^3)$ Hungarian algorithm. Our next contributions will rely on this assignment framework.

The second contribution of this thesis is a joint model that combines syntactic parsing and SRL (Lluís et al., 2013). We solve it by using dual-decomposition techniques. A strong point of our model is that it generates a joint solution relying on largely unmodified syntactic and SRL parsers. We train each component independently and the dual-decomposition method finds the optimal joint

solution at decoding time. Our model has some optimality and efficiency guarantees. We show experiments comparing the pipeline and joint approaches on different test sets extracted from the CoNLL-2009 Shared Task. We observe some improvements both in syntax and semantics when our syntactic component is a first-order parser. Our results for the English language are competitive with respect to other state-of-the-art joint proposals such as Henderson et al. (2013).

The third contribution of this thesis is a model that finds semantic roles together with syntactic paths linking predicates and arguments (Lluís et al., 2014). We frame SRL as a shortest-path problem. Our method instead of conditioning over complete syntactic paths is based on the assumption that paths can be factorized. We rely on this factorization to efficiently solve our problem. The approach represents a novel way of exploiting syntactic variability in SRL. In experiments we observe improvements in the robustness of classifiers.

1

INTRODUCTION

Natural Language Understanding (NLU) is a complex topic still far from being solved. Syntactic dependency parsing and Semantic Role Labeling (SRL) are two key tasks that could provide relevant insights regarding NLU. Progress in these tasks may also benefit a wide range of Natural Language Processing (NLP) applications. In the last years, there have been many proposals of systems that use the output of dependency parsing and SRL analysis for several applications, including automatic summarization (Melli et al., 2006), question answering (Narayanan and Harabagiu, 2004), information extraction (Surdeanu et al., 2003), co-reference resolution (Kong et al., 2008; Màrquez et al., 2013) and machine translation (Boas, 2002).

In the last decade, significant progress has been made towards a deep understanding of syntactic parsing and semantic role labeling partially summarized in the CoNLL-2008 and 2009 Shared Tasks (Surdeanu et al., 2008; Hajič et al., 2009). These shared tasks and also this thesis are framed under the data-driven approach. The availability of large-sale hand-annotated resources such as FrameNet (Fillmore et al., 2004), PropBank (Palmer et al., 2005) and NomBank (Meyers et al., 2004) boosted this data-driven research.

This work is focused on these following fundamental problems in natural language understanding: syntactic dependency parsing and semantic role la-

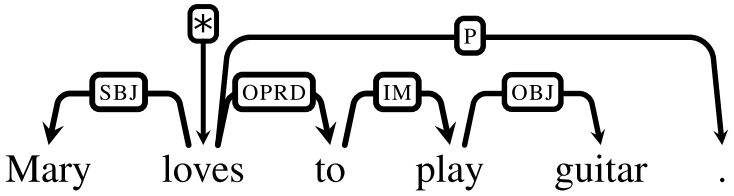

Figure 1.1: A sentence with syntactic dependencies

beling. But more specifically, we focus on the joint processing of both tasks.

Syntactic Dependency Parsing Parsing is the process of building a structure from a sequence of symbols (i.e., a sentence) with respect to a grammatical formalism. Many NLP tasks can be regarded as finding an underlying structure or parse tree. Syntactic dependency parsing is the analysis of the syntactic structure of a sentence under a formalism that defines a set of lexical elements linked by dependencies (Mel'čuk, 1998). For example, figure 1.1 shows a syntactic parse of the sentence *Mary loves to play guitar*. In the figure, we place a dependency between *loves* and *Mary* in order to indicate a *verb-subject* relationship between the two. This relationship is labeled with the syntactic tag *SBJ*. A dependency links a modifier with their head. E.g., *loves* is the head of *Mary*. The token * is the root of the sentence. Formally, a dependency structure is commonly a weakly-connected directed acyclic graph with a fictitious root node. Each node in the graph has exactly one head except the root. Other constraints may be applied depending on the formalism or framework.

The dependency parsing framework contrasts with the constituent approach, the latter consists of decomposing the sentence into a set of hierarchically-related constituents. Syntactic dependency parsing may offer some advantages over the latter:

- Dependency graphs can better capture the structure of free-order languages. The constituent formalism forces a less expressive projective parse. Projectivity is the planarity of the dependency graph restricted to the upper

half of the graph. In other words, a projective tree is one in which dependencies do not cross.

- Many NLP applications such as information extraction, question answering and co-reference resolution could benefit from directly exploiting the dependency-based representation.

- Dependencies are direct relations between lexical items, allowing for a straightforward exploitation of lexical features. These lexical features are important in order to obtain high accuracies with statistical methods. Constituent-based models must be extended in order to incorporate such features (Collins, 1999).

The most common algorithms used to solve data-driven dependency parsing are the the Eisner bottom-up parsers (Eisner, 2000) and the shift-reduce parsers (Covington, 2001; Nivre et al., 2007b). A more detailed review of these algorithms is presented in chapter 2.

Semantic Role Labeling Semantic role labeling is the semantic processing of a sentence where the arguments for a given predicate are identified and classified (Gildea and Jurafsky, 2002). The arguments capture semantic properties that answer questions of the type *who* did *what* to *whom*, *how*, and *why*.

Figure 1.2 shows a sentence with the predicates *loves* and *play*. And for example, *guitar* is an argument of the predicate *play*. Arguments are classified by the semantic role that they play with respect to the predicate. In the figure, *guitar* fills the *instrument* or *ARG1* role for the predicate *play*.

Semantic roles provide an interesting level of abstraction with respect to syntax. For example regarding the sentences: *Mary plays the guitar* and *The guitar is played by Mary*, in both sentences *Mary* and *guitar* will realize the same semantic role for the main predicate but with different syntactic function.

In this thesis we view SRL as a form of semantic parsing. Semantic parsing has a long tradition in NLP, with many proposals for representing the semantics

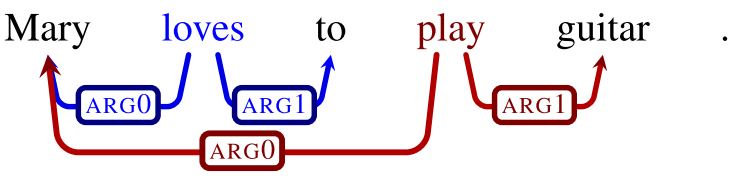

Figure 1.2: A sentence with semantic dependencies for the predicates *loves* and *play*.

of the sentence that are much richer than SRL structures, e.g., meaning-text theory (Mel'čuk, 1981), lambda calculus (Wong and Mooney, 2007), CCG grammars (Bos et al., 2004) or dependency-based compositional semantics (Liang et al., 2011).

SRL is often described as a *shallow* representation of the semantics of the sentence. This thesis is focused on dependency-based SRL, and we will refer to it as SRL, shallow semantic parsing or simply semantic parsing.

We take a data-driven approach to semantic role labeling. Our experimental datasets are based on the CoNLL-2009 Shared Task data (Hajič et al., 2009). For the English language, the datasets were extracted from the hand-annotated PropBank and NomBank (Palmer et al., 2006; Meyers et al., 2004). These datasets were converted to a dependency representation by a process described in Surdeanu et al. (2008). In some of these datasets, the semantic core roles are labeled with agnostic tags of the type *ARGn* that are not necessarily consistent across different predicates.

SRL is typically addressed by a three-stage process: first pruning/filtering of the candidates (Xue and Palmer, 2004), then identifying and classifying the arguments and finally a global post-processing may be applied such as in Toutanova et al. (2005). A more in-depth summary of the state of the art of SRL can be found on chapter 2.

Syntactic-semantic parsing We consider syntactic-semantic parsing as performing both the syntactic dependency analysis and the semantic role labeling. Figure 1.3 shows these two parses for our example sentence. In the example, *Mary* is a direct semantic dependent of *play*, but both words are syntactically

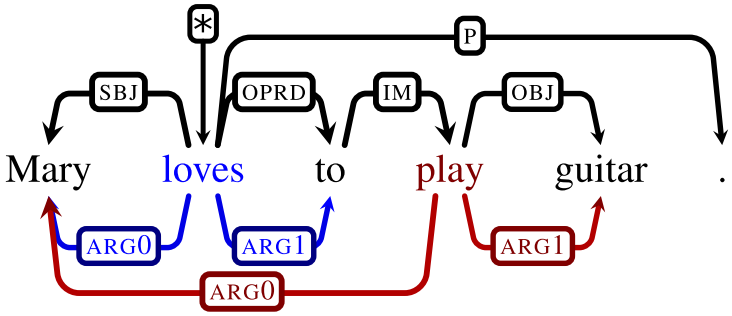

Figure 1.3: A sentence with syntactic and semantic dependencies

distant as they are separated by 3 syntactic dependencies. Syntax and semantics are assumed to be related, but these relations appear to be non-trivial and not governed by a set of simple rules. How these *relations* could be statistically modeled is a challenging question that we will discuss along this dissertation.

Syntactic and shallow semantic parsing are typically addressed as a sequence of tasks (Hajič et al., 2009). Usually, complex natural language understanding problems such as those discussed here are typically solved by using a pipeline architecture. In a pipeline approach the complex problem is decomposed into a sequence of affordable subtasks. Unfortunately, propagated errors through the pipeline are hard to recover and severe performance degradation is observed in the latter stages (Gildea and Palmer, 2002). Furthermore, these models are unable to exploit some of the potential interactions between syntax and semantics. This thesis is focused in modeling joint systems.

1.1 Goals and contributions

This dissertation has as its main goal the exploration of joint models for syntax and shallow semantics. There have been a small number of joint proposals such as the ones introduced by Sutton and McCallum (2005), Yi and Palmer (2005) and Collobert and Weston (2008). Furthermore, the CoNLL-2008 and 2009 Shared Tasks were intended to boost the research in this topic. However, only few models were proposed such as the Gesmundo et al. (2009) and Morante et al. (2009b) systems. In general, there was a lack of controlled experiments,

thus the results of those shared tasks were mostly inconclusive (Surdeanu et al., 2008; Hajič et al., 2009). A more in-depth review of joint proposals will be found in chapter 2.

In this thesis our goal is to make progress towards understanding the following two motivating questions. Our first question is *whether a joint system can outperform the pipeline approach*. The relevance of this question is justified by the current limitations of the pipeline architectures. A joint approach could potentially improve the results on both tasks by capturing more interactions between the syntactic and semantic layers. Furthermore, a better understanding of the interactions between complex NLP tasks could represent a significant advancement in the field.

A major drawback of a joint system is the potential increase in computational cost with respect to a pipeline approach. Our second question is *how to build an efficient joint system*. Joint systems are expected to search in a much larger space to allow to reconsider some decisions previously made. We expect to pay *some* computational penalty by exploring this larger space in a joint model.

To at least partially answer these previous questions and to explore joint syntactic and semantic parsing we summarize here the main contributions of this thesis:

- Our first contribution is to frame semantic role labeling as a weighted linear assignment task. This framework provides an efficient way of enforcing unique roles for a given predicate. We exploit this contribution in our latter joint proposals to generate the semantic parse.

- Our second contribution is to define a joint syntactic-semantic model solved by using dual-decomposition techniques. We include an experimental section to evaluate our proposal with respect to an equivalent pipeline system. Regarding our first motivating question of *whether a joint system can out-*

perform the pipeline approach we give some results to at least partially answer this question for our particular proposal.

- Our third contribution is to frame semantic role labeling as a shortest-path problem. We jointly find the semantic roles together with the syntactic paths linking predicates with their arguments. This latter contribution allows us to efficiently process datasets with complex predicate-argument syntactic paths and complements our previous joint proposal.

1.2 Overview of this Document

The rest of this dissertation is organized as follows. We first review the state of the art and then each chapter describes each one of our contributions.

Chapter 2: State of the Art. This chapter gives an overview of the previous work directly relevant to the discussed topics and categorizes the current approaches for joint syntactic and semantic parsing.

Chapter 3: SRL as Assignment. We frame the SRL problem as finding a weighted assignment in a bipartite graph. Under this framework we can efficiently control for non-repeated constraints on SRL. The optimization method introduced here will be applied in our joint systems described on chapters 4 and 5.

Chapter 4: Dual-decomposition Joint Syntactic-semantic Parsing. Here, we describe our proposal for a joint system. We solve the joint optimization problem by using dual-decomposition techniques. Our method finds the optimal joint parse whenever it converges.

Chapter 5: Shortest-path Syntactic-semantic Parsing. We describe a semantic role labeling approach that also builds the syntactic paths linking predicates with their arguments. Our proposal is based on finding shortest-paths and it is flexible enough to capture complex predicate-argument paths.

Chapter 6: Conclusions and Future Work. We conclude this dissertation drawing our main conclusions and pointing out possible future research directions. We summarize here the discussions from the previous chapters.

The appendices will give supplementary material not included in the body of this document for sake of clarity.

Appendix A: Algorithms. We describe our basic algorithms and related technical details.

Appendix B: The CoNLL-2006 and 2007 Shared Tasks. Here are outlined the main results of these shared tasks.

Appendix C: Datasets, Treebanks and Measures. We sum up the main scoring measures and details of our training, development and test datasets.

1.3 Published Work

We list here all our publications. The core of the chapters 3 and 4 was first described in:
Joint Arc-factored Parsing of Syntactic and Semantic Dependencies
Xavier Lluís and Xavier Carreras and Lluís Màrquez
Transactions of the Association for Computational Linguistics (TACL), vol 1:219–230, May 2013.

The shortest-path approach for syntactic-semantic parsing was presented in:
A Shortest-path Method for Arc-factored Semantic Role Labeling
Xavier Lluís and Xavier Carreras and Lluís Màrquez
Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, October 2014.

Our initial proposal for a joint syntactic-semantic system based on a first-order graph-based parser was described in:

A Joint Model for Parsing Syntactic and Semantic Dependencies
Xavier Lluís and Lluís Màrquez
Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL 2008): Shared Task, August 2008.

A more detailed description of this first proposal including additional material can be found at:
Joint Learning of Syntactic and Semantic Dependencies
Master's Thesis. Universitat Politècnica de Catalunya. September 2008.

The multilingual and second-order extension to the previous proposal including improved results was published at:
A Second-Order Joint Eisner Model for Syntactic and Semantic Dependency Parsing
Xavier Lluís and Stefan Bott and Lluís Màrquez
Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task, June 2009

2

State of the Art

This chapter gives an overview of the state of the art of syntactic and semantic dependency parsing. We start by reviewing syntactic dependency parsing in the first section. Next section describes from our perspective the most relevant semantic role labeling approaches. Finally, we give an overview of the main proposals that generate both the syntactic and semantic analysis. We classify these proposals starting with pipeline approaches and ending with systems that simultaneously process both syntax and semantics.

2.1 Syntactic Dependency Parsing

Dependency parsing is the process of analyzing a string. Dependencies are syntactic relations between words or tokens (Mel'čuk, 1998).

We first introduce our basic notation for dependency parsing. Let $\mathbf{x}$ be a string or the input sentence. We denote the words or tokens of this sentence as $x_1, \ldots, x_n$. For convenience, we add a special *root* token x_0. A dependency tree is a directed graph T defined as follows: $V(T)$ are the vertices representing the sentence words including the *root* token. $E(T)$ corresponds to the set of labeled directed arcs, i.e., the dependencies between the words. A dependency $\langle h, m \rangle$ is a relation between a head h and a modifier or dependent m. In the context of syntactic dependency parsing we typically add a label l that represents the

syntactic function between m and h as in $\langle h, m, l \rangle$. A dependency tree T is well-formed if and only if (Nivre, 2006):

- *connectedness*: T is weakly connected, i.e., if we replace the arcs of T with undirected edges the resulting graph is connected.
- *single-root*: T has a single root node x_0 with no incoming edges.

Typically tighter constraints are enforced:

- *single-head*: every node except the root has a single head.
- *acyclicity*: there is no non-empty directed path that starts and ends on the same node.
- *projectivity*: given a dependency $\langle h, m \rangle$ in any directed path from h to k, k must be in the range $h \leq k \leq m$ or $m \leq k \leq h$. Projectivity is planarity on the upper half of the graph.

The latter *projectivity* property is relevant as an important family of parsing algorithms are only able to parse projective structures.

2.1.1 Parsing algorithms

Some authors (McDonald and Nivre, 2007; Kübler et al., 2009) made a distinction on dependency parsing algorithms between *graph-based* and *transition-based* parsers. The Maximum Spanning Tree (Chu and Liu, 1965; Edmonds, 1967) and the Eisner (2000) algorithms are considered graph-based algorithms. In contrast, shift-reduce parsers introduced by Covington (2001) and Nivre (2003) in the context of dependency parsing are categorized as transition-based algorithms. In this thesis we define joint models based on graph-based algorithms. Therefore along this chapter we will mainly focus on graph-based approaches and we will only briefly review other families of parsers.

Dependency parsing consists of computing the best parse tree $\mathbf{y}^*$ for an input sentence $\mathbf{x}$:

$$\mathbf{y}^* = \underset{\mathbf{y} \in \mathcal{Y}(\mathbf{x})}{\operatorname{argmax}}\, s(\mathbf{x}, \mathbf{y}) \quad ,$$

where $\mathcal{Y}(\mathbf{x})$ is the set of all dependency trees for $\mathbf{x}$ and and the function $s(\mathbf{x}, \mathbf{y})$ assigns a weight, score or probability to each parse tree $\mathbf{y}$ given the input sentence.

The feasibility of parsing algorithms (i.e., the computation of the argmax) crucially relies on a *factorization* of the scores of the trees. We start by defining an arc-based factorization. The key idea of this factorization is the fragmentation of the score of the tree by the sum of independent scores:

$$\text{score}(\mathbf{x}, \mathbf{y}) = \sum_{f \in \mathbf{y}} \text{score}(\mathbf{x}, f) \quad ,$$

where f is a factor of the dependency tree. In first-order models, factors are single arcs. I.e., $f = \langle h, m, l \rangle$, the head, the modifier and the syntactic label. The score of a factor is typically computed by a linear function:

$$\text{score}(\mathbf{x}, f) = \mathbf{w} \cdot \phi(\mathbf{x}, f) \quad ,$$

where $\mathbf{w}$ is a weight vector parameterizing the model and ϕ is a feature extraction function. Note that usually and in addition to the factor features, the input sentence $\mathbf{x}$ is also considered, and therefore passed to the function ϕ.

First-order Parsing Algorithms The Eisner algorithm performs the required first-order inference in $O(n^3)$. The algorithm details are in appendix A. The inference is restricted to projective trees, i.e., $\mathcal{Y}(\mathbf{x})$ is in this case the set of projective parse trees for $\mathbf{x}$.

Alternatively, the Maximum Spanning Tree (MST) model solves first-order dependency parsing without being restricted to projective parses. A maximum spanning tree is a tree in a graph with the highest scored edges. The Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967) efficiently solves the MST problem in $O(n^2)$. This algorithm is also described in the appendix A. This approach was introduced in the context of dependency parsing by McDonald et al. (2005b). Unfortunately, it has not been successfully extended. Much

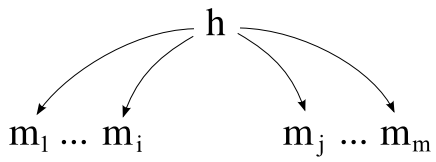

Figure 2.1: Second order sibling dependencies.

more attention had received Eisner-based algorithms (McDonald and Pereira, 2006) we believe that the reason is a higher degree of flexibility offered by this latter family of parsers.

2.1.1.1 Higher-order Parsing

McDonald and Pereira (2006), Carreras (2007) and Koo and Collins (2010) extended first-order models with second and third-order dependencies considering siblings and grandchildren in the factorizations.

Second-order Siblings Second-order factorizations include some degree of sibling information. The score of a parse is equally defined by the sum of its factors scores:

$$\text{score}(\mathbf{x}, \mathbf{y}) = \sum_{f \in \mathbf{y}} \text{score}(f) \quad .$$

But now a factor is considered as the tuple $\langle h, s, m, l \rangle$:

$$s(\mathbf{x}, \mathbf{y}) = \sum_{\langle h,s,m,l \rangle \in y} s(\mathbf{x}, h, s, m, l) \quad .$$

The score function evaluates pairs of adjacent dependencies. The adjacency is only relative to the left or right of the head, i.e., a left/right independence is assumed. The function $s(\mathbf{x}, h, s, m)$ is defined for the head h, the sibling s, the modifier m and the syntactic label l. A null sibling, i.e., no left or right modifier is denoted as "-". For example, considering l_i as the syntactic label assigned to the i-th modifier, the second-order score for the set of dependencies shown in figure 2.1 is

$$\sum_{k=1}^{i-1} s(\mathbf{x}, h, m_{k+1}, m_k, l_k) + s(\mathbf{x}, h, -, m_i, l_i) +$$
$$s(\mathbf{x}, h, -, m_j, l_j) + \sum_{k=j}^{m-1} s(\mathbf{x}, h, m_k, m_{k+1}, l_{k+1}) \quad .$$

Eisner (2000) introduced an efficient $O(n^3)$ algorithm to compute the best second-order parse in a projective tree. The algorithm was latter reintroduced by McDonald and Pereira (2006) and it is detailed in appendix A, see algorithm 6. A latter publication by McDonald and Satta (2007) proved that the non-projective version of this model is NP-hard to solve.

Second-order Grandchildren A further second-order improvement was introduced by Carreras (2007) allowing some grandchildren information to be captured. Improvements were reported at an expense of a higher computational cost of $O(n^4)$. As it was in the case of the previous second-order model, this extension is restricted to sibling and grandchildren information relative to the span concept and left/right independence is assumed. The second-order model considers factors as:

$$f = \langle h, m, l, c_h, c_{mo}, c_{mi} \rangle$$

where h, m, l are the head, modifier and label as defined in first-order models. c_h is the child of h in $[h \ldots m]$ closest to m, i.e., the sibling as defined in the second-order siblings model. c_{mi} is the child of m in $[h \ldots m]$ furthest from m and c_{mo} is the child of m outside $[h \ldots m]$ furthest from m. Thus some grandchildren of h are considered in the scoring function. This function is defined as follows:

$$s(\mathbf{x}, \langle h, m, l, c_h, c_{mo}, c_{mi} \rangle) = s(\mathbf{x}, h, m, l) + s(\mathbf{x}, h, m, l, c_h) +$$
$$s(\mathbf{x}, h, m, l, c_{mi}) + s(\mathbf{x}, h, m, l, c_{mo}) \quad .$$

The algorithm is described in appendix A, see algorithm 7.

Third-order Parsing Koo and Collins (2010) introduced an algorithm to capture higher third-order dependencies. Dependencies involving up to 4 nodes are modeled. Their algorithm extended the previous second-order models with also an $O(n^4)$ time complexity. At a practical level it is crucial to prune the search space to achieve reasonable computation times. The authors showed improvements specially when parsing non-projective languages.

Shift-Reduce and Transition-based Parsers Shift-reduce parsers are bottom-up parsers that can be defined as parsing a sentence from left to right and using a stack of symbols and a queue (Aho et al., 1986).

A *transition* is a function between parser configurations. A configuration can be defined by the *stack*, the *input queue* and the *partially constructed tree*. In the context of dependency parsing a classifier (e.g., SVM (Joachims, 1999)) selects the best transition given the current configuration also called the *history*. The parsing algorithm is described in appendix A, see algorithm 8. The algorithm is linear assuming a constant-time *oracle* function that is approximated by the trained classifier. We will not provide here a more detailed review of these family of parsers as this thesis is mainly focused on graph-based parsers. A further discussion of the transition-based models can be found in Kübler et al. (2009).

Other approaches Syntactic dependency parsing is still a very active research topic. An overview of the results of the CoNLL-2006 and 2007 Shared task can be found in the appendix B. As an example of recent work, Rush and Petrov (2012) applied coarse-to-fine inference up to third-order parsing. They showed significant speed-ups in computation time. Intuitively, given a set of parsers, say first, second and third-order parsers they prune the search space starting from the faster first-order parser. Each parser filters the search space of the next parser and in some cases they allow the possibility to amend the pruning. An analogous approach was presented by Zhang and McDonald (2012) also based on pruning.

McDonald and Nivre (2011) combined an Eisner parser with a shift-reduce parser by each parser using features extracted from the other. The best gains were obtained in Eisner-based parsing when features from the shift-reduce model are included. Martins et al. (2009) made a remarkable contribution by compactly formalizing first and second-order dependency parsing as a set of linear equations. An integer linear programming solver is then applied.

Koo et al. (2010) trained head-automata models and dependency parsers. They applied dual-decomposition methods to take advantage of the head-automata high accuracies and also from the tree constraints enforced by dependency parsers. This approach decomposes the problem into two sub-problems that need to agree on the syntactic dependencies they predict.

Martins et al. (2013) presented a non-projective third-order model based on head automata and sequential bigram models combined with a dual-decomposition optimization method. Lei et al. (2014) parsed using a low-dimensional representation of the feature space achieving remarkable results.

2.1.1.2 Remarks and Future Work

A key idea in syntactic dependency parsing is the factorization of the score or probability of the parse tree. It implies that the factors are unrelated or conditionally independent with respect to each other. This arc-based factorization is probably an unrealistic assumption that higher-order models try to alleviate. In these latter models, however, still only local information is mainly considered (e.g., just immediately adjacent dependencies not crossing the position of the head).

A common strategy to alleviate these previous limitations is to capture some additional context information by extracting features from the surrounding tokens. Another approach is to use re-ranking or systems combination to improve the final results. In that case, global properties of the output can be used as features.

All these strategies have shown significant performance improvements. In general, recent parsing proposals have increased the parsing time complexity (e.g., third-order parsers, hybrid parsers) and are commonly handled with aggressive pruning strategies. However, these latter approaches still appear not to be flexible enough to capture long distance dependencies.

2.2 Semantic Role Labeling

Semantic role labeling (SRL) consists of identifying in a sentence the arguments of the predicates and labeling them with semantic roles (Gildea and Jurafsky, 2002). SRL is a relevant semantic task in NLP since predicate-argument relations directly represent semantic properties of the type *who* did *what* to *whom*, *how*, and *why* for the events expressed by the predicates.

Predicate-argument relations are strongly related to the syntactic structure of the sentence. Most 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. Semantic roles provide an interesting level of abstraction with respect to syntax. While syntactic functions of arguments change with the form of the event (e.g., active vs. passive forms), the semantic roles of the arguments remain invariant to their syntactic realization.

SRL comprises the annotation of semantic core roles such as agent and patient, but also the adjunct arguments of the predicate, e.g., locative, temporal, manner, cause.

The CoNLL-2008 Shared Task popularized a framework were arguments are semantic *dependents* of their predicates. Under the dependency framework we consider that predicates are linked to their arguments in a semantic dependency graph. If we merge the semantic graphs from all predicates a directed graph is conformed, without any further constraint.

However, in general, the introduction of this dependency formalism in SRL did not change the consideration of SRL mainly as a classification task. The

task is typically addressed by the following three-stage architecture:

Pruning or filtering. The first step is intended to reduce the amount of tokens that will be considered as potential arguments. Any sentence token could be identified as an argument of a predicate but most tokens are non-arguments. This renders an unbalanced distribution that may degrade the performance of some classifiers. Filtering alleviates this problem with almost no penalty regarding the argument recall. In addition, it significantly speeds-up the computation times. Xue and Palmer (2004) introduced a widely applied pruning rule for the English language datasets.

Argument identification and classification. Each candidate is labeled with a role possibly including a null tag either in a single step or more commonly in a two-stage process. In this latter case, the candidates are initially classify as arguments or non-arguments. Then semantic roles are assigned to each argument.

Global scoring or post-processing. In this latter step a wide variety of techniques are applied to improve the final results. These techniques may combine a bag of candidates from the previous steps and extract global features from all candidates. An example of these techniques is the re-ranking approach of Toutanova et al. (2005). In addition, domain constraints can be enforced over the complete SRL parse of a sentence (Punyakanok et al., 2004).

Besides the 3-stage classification approach, semantic role labeling can also be addressed by BIO tagging (Màrquez et al., 2005) or by CRF on tree structures (Cohn and Blunsom, 2005; Moreau and Tellier, 2009). A summary of SRL approaches is found in the CoNLL-2004 and 2005 Shared Tasks (Carreras and Màrquez, 2004; Carreras and Màrquez, 2005) that evaluated SRL approaches under a framework of predicates already identified and syntactic constituent trees and chunks provided.

team	ml	synt	pre	arch	glob	post	comb	prec (%)	rec (%)	F_1
punyakanok	SNoW	n-best	prun	i+c	yes	no	yes	81.18	74.92	77.92
haghighi	ME	n-best	?	i+c	yes	no	yes	81.87	73.21	77.30
màrquez	AB	2 parses	no	BIO	no	no	yes	78.34	75.78	77.04
pradhan	SVM	3 parses	?	BIO	no	no	yes	78.44	74.83	76.59
surdeanu	AB	1	prun	c	no	yes	no	79.35	71.17	75.04

Table 2.1: Summary of top performing CoNLL-2005 Shared Task systems

Table 2.1[1] shows the top 5 systems of the CoNLL-2005 Shared Task that were among the 19 submitted results. It shows also a summary of the approaches. The column *ml* contains the machine learning method. The *synt* column indicates if multiple syntactic trees were used to extract features. The next column *pre* indicates if preprocessing was a first stage. *Arch* column contains if the architecture performs identification and classification ($i + c$), only classification (c) or it is a *BIO* tagging process. The *glob* column flags if global information is used through the annotation process. *Post* contains if the system applied a simple post-processing that could be a set of rules. The final column *comb* indicates if the result is build from a system combination.

The most common approaches were based either in BIO tagging or in a identification and classification process. System combination was a widely used technique to improve results in the top performing systems. These systems build the SRL parse from different syntactic trees or from a set semantic role classifiers trained with different feature sets. Almost all systems extracted features from the syntactic path and the phrase structure around the predicate parent. This intensive use of syntactic features reinforces the idea that these are crucial for SRL in that task.

More recently, Punyakanok et al. (2008) introduced an approach that explicitly controls semantic role labeling constraints, as well as other constraints

[1]See http://www.lsi.upc.edu/ srlconll/st05/st05.html for detailed scores. This table is extracted from Carreras and Màrquez (2005). ME: maximum entrophy, AB: AdaBoost, prun: pruning, i+c: identification and classification.

that look at expressive pairwise assignments. They solve SRL using a general-purpose integer linear programming method. In similar spirit, Riedel and McCallum (2011) presented a model for the extraction of structured events that controls interactions between predicates and arguments. They take into account pairwise assignments and solve the optimization problem by using dual- decomposition techniques. Das et al. (2012) have proposed a dual-decomposition method that deals with several assignment constraints for predicate-argument relations. Täckström et al. (2015) presented a dynamic programming method that enforces some SRL domain constraints. These last two approaches were presented as alternatives to general ILP methods.

Naradowsky et al. (2012) presented a method based on Markov random fields that avoids the need of syntactically annotated data or automatic parses. Their method is able to infer an intermediate syntactic-like structure while jointly finding semantic roles. In the experiments, they have shown that for some datasets such as the CoNLL-2009 Japanese data, this intermediate structure helps improving unlabeled SRL scores. In those datasets where there were strong divergences between the syntactic tree and the predicate-argument dependencies the inferred syntactic-like layer better correlates with the predicate-argument induced graph.

Recently, the SemEval 2014 Task 8 was devoted to semantic dependency parsing (Oepen et al., 2014). One of the main goals of the task was to stimulate parsing approaches beyond tree structures. However, many proposals only transformed the original semantic dependency graph into a tree structure to then apply conventional dependency parsers. There were some exceptions such as Kuhlmann (2014) that extended the Eisner algorithm to parse a restricted class of directed graphs. Martins and Almeida (2014) modeled the task by scoring parts of the graph and allowing for multiple parents. Decoding is performed by using a a dual-decomposition strategy. Ribeyre et al. (2014) used a transition-based model extended with operation that allow to process a more generic acyclic graph structure.

Remarks and Conclusions SRL is still mainly approached by the 3-step process of filtering/pruning, argument identification and classification and global scoring. The first stage is a simple and widely performed process to discard improbable argument candidates.

The second stage, intended to identify and classify the arguments is mainly addressed by trained classifiers based on features defined in works such as Xue and Palmer (2004). Different learning algorithm are used in different proposals suggesting that there is no single-best method.

The final stage of SRL systems covers a wide variety of techniques. Global features and constraints regarding all argument candidates may be considered. A re-ranking method or a domain constraint satisfaction approach can be applied over the set of previously generated candidates.

Some interesting alternatives to the common 3-step architecture are the extended parsers of Kuhlmann (2014) and Ribeyre et al. (2014) and also the CRF approach of Cohn and Blunsom (2005). Naradowsky et al. (2012) presented also a graphical model that avoids the need of a syntactic parse by inferring a hidden syntactic-like structure.

Almost all reviewed SRL systems strongly rely on features extracted from the input syntax. State-of-the-art proposals exploit syntax from different syntactic parsers in order to avoid the limitations of a pipeline approach. However, the possibilities of this last post-processing are still limited to the set of previous generated candidates.

2.3 Syntactic-Semantic Parsing

In this section we give an overview of syntactic and semantic parsing. We consider syntactic-semantic parsing as the analysis under a shared dependency representation of syntactic dependencies and semantic roles (Surdeanu et al., 2008). We include in this definition any model that generates both layers either

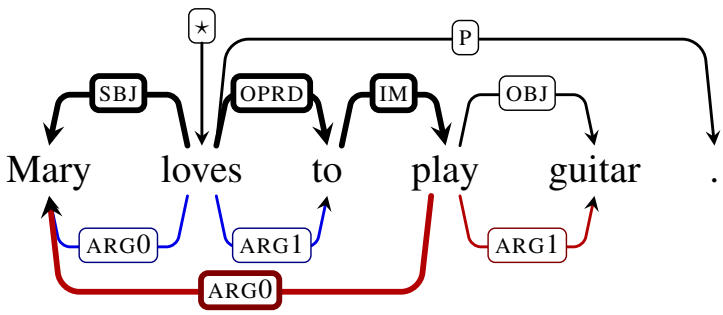

Figure 2.2: A sentence with syntactic dependencies (top) and semantic dependencies for the predicates *loves* and *play* (bottom). The thick arcs illustrate a structural divergence where the argument *Mary* is linked to *play* with a path involving three syntactic dependencies.

independently or jointly. Figure 2.2 shows a sentence with its syntactic and SRL parses.

If we look at the dependency-based representation of the figure, we can observe important structural divergences between the syntactic and semantic layers. For example, 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. This phenomena –the *distant arguments*– is not uncommon as it occurs in about the 25% of the arguments in the English development set of the CoNLL-2009 Shared Task.

In syntactic-semantic parsing, given an input sentence $\mathbf{x}$ we are interested in finding the best syntactic tree and semantic analysis pair $\mathbf{y}^*, \mathbf{z}^*$. Our objective function can be written as:

$$\langle \mathbf{y}^*, \mathbf{z}^* \rangle = \underset{\mathbf{y} \in \mathcal{Y}, \mathbf{z} \in \mathcal{Z}}{\operatorname{argmax}}\, s(\mathbf{x}, \mathbf{y}, \mathbf{z}) \quad ,$$

where $\mathcal{Y}$ is the set of all parse trees and $\mathcal{Z}$ is the set of all semantic role analysis. We assume that the previous function can be rewritten as:

$$\langle \mathbf{y}^*, \mathbf{z}^* \rangle = \underset{\mathbf{y} \in \mathcal{Y}, \mathbf{z} \in \mathcal{Z}}{\operatorname{argmax}} \left(s_{\text{syn}}(\mathbf{x}, \mathbf{y}) + s_{\text{srl}}(\mathbf{x}, \mathbf{y}, \mathbf{z}) \right) \quad . \tag{2.1}$$

Without any further assumption or simplification this problem remains in general unfeasible. To compute the best pair $\mathbf{y}^*, \mathbf{z}^*$ we have to explore all possible combinations of parse trees $\mathbf{y}$ and semantic parses $\mathbf{z}$. The functions s_{syn} and s_{srl} compute the score of a syntactic and SRL parse respectively.

A common simplification that makes the problem feasible is to restrict the search space $\mathcal{Y}$ to the k-best syntactic trees. Thus for any finite small k we can find the optimal solution to the Eq. (2.1) by computing the s_{syn} and s_{srl} functions only for all SRL parses over every k-best syntactic tree. Then we choose the highest scored combination. A pipeline system is a particular case where $k = 1$. These k-best trees must be previously computed by a syntactic parser assumed to be of polynomial cost.

We next briefly review the *pipeline* and *k-best* approaches to later discuss what we consider *joint models*.

2.3.1 Pipeline Models

A first approach to syntactic-semantic parsing is to use a pipeline architecture. First the best syntactic parse tree is computed. And then an SRL system chooses the optimal semantic roles for this first syntactic parse tree. Pipeline models consistently obtained remarkable results across different test sets. E.g., Che et al. (2009) presented a pipeline system that ranked first in the multilingual CoNLL-2009 Shared Task. That system chained a second-order syntactic dependency parser with an SRL system. SRL in that system was based on Maximum Entropy SRL classifiers combined with an integer linear programming post-processing.

2.3.2 k-best Approaches

A k-best model is intended to alleviate some of the limitations of the pipeline approach while avoiding the enumeration of an unfeasibly large space. k-best models turn the optimization problem feasible by only working with a small set of best parse trees according to some syntactic ranker.

Gildea and Jurafsky (2002) generated a number of syntactic candidates, and re-scored them according to a SRL system, then the best syntactic analysis is selected. The authors only presented the syntactic results of this strategy. However, they pointed out the possibility of working with a set syntactic trees to improve syntactic-semantic parsing. Sutton and McCallum (2005) introduced a proposal that generates the k-best parse trees and then reranks syntax considering the SRL analysis. Their solution is generated by either combining scores or by training a reranker with global features. In addition, a final SRL post-process was applied to check for argument overlaps. Unfortunately, in that system, no joint configuration outperformed a pipeline approach using the best parse tree. A similar approach was taken by Chen et al. (2008) also showing mild results. Samuelsson et al. (2008) achieved similar results working with a set of syntactic and SRL parsers, the final output was generated through a set of combination strategies.

Johansson and Nugues (2008) presented a competitive k-best system. The k-best syntactic trees were computed by a second-order syntactic parser. On top of these k-best trees, an SRL system generated the semantic role analysis. The best joint result was computed by combining the probabilities assigned by each component. The approach was indeed costly and the authors only considered at most the top 16 syntactic parses. The system evaluated in the context of the CoNLL-2008 Shared Task achieved the best overall results.

Li et al. (2010) applied a k-best approach for the Chinese language. Their model combines probabilistic scores of the syntactic parse and the SRL analysis. The authors showed improvements with respect to an equivalent pipeline system.

2.3.3 Joint Approaches

Here we consider as a joint approach any syntactic-semantic model where syntax is not restricted to the k-best syntactic parses for a small value of k. We include in this section some models that do not formalize any joint objective

function but that are allegedly able to capture syntactic and semantic interactions beyond the limitations of the k-best systems. Note that without a formal analysis we cannot guarantee that some of these approaches are in fact searching over a larger space than the previous k-best proposals.

We classify these proposals starting by the *Label Extension* models that enrich syntactic parsing labels with semantic roles. We latter describe the *Iterative Models* that sequentially run syntactic and SRL parsers for a number of iterations. The *Joint Graph-based* and *Joint Transition-based* proposals introduce extensions to these two main categories of dependency parsers to jointly process syntax and semantics. Finally, a joint system can also be built from a set of *Shared Features*.

2.3.3.1 Label Extension

Musillo and Merlo (2006) extended syntactic parsing labels with semantic roles. The system concatenates the POS or syntactic labels with semantic tags in a single label. (e.g., a constituent *SUBJ* that also is *A0* for some predicate will be labeled as *SBJ-A0*). With this new set of tags a classifier is trained and the sentences annotated. The authors only evaluated the syntactic results of the system. A post-process would be required to recover the separate semantic labels and to link the roles to the corresponding predicate. A drawback of this approach is that it generates a large amount of combined labels with a few training examples per label.

Yi and Palmer (2005) also attached semantic argument information to syntactic constituent labels to train a syntactic parser. To avoid the large number of classes generated by label combinations they only extended the syntactic labels with an argument identification tag for core and adjunct-like arguments. They showed improvements when the syntactic parser was extended to also predict argument identification tags. The best results where achieved by combining features extracted from standard syntactic parsers with their extended parsers. Also Ge and Mooney (2005) presented a similar approach.

Morante et al. (2009b) described a system based on a memory-based learning strategy. A label is the concatenation of the syntactic function and the semantic role between two tokens, e.g., *NMOD-A1*. A dependent may have as many joint labels as sentence predicates and possible heads. For example, a token with the set of labels $\{p_1$*:NMOD-A1*, p_2*:_-A2*$\}$ is a token that it is syntactically dependent to its head p_1 with the *NMOD* function. That head p_1 is also a predicate for which the token is filling the role *A1*; the same dependent token is the *A2* argument of some other predicate p_2 which is not its syntactic head. This strategy also generates a large number of labels. The architecture performs two main steps. First, a classifier is intended to generate syntactic and semantic dependencies. Then a ranker refines the predictions from the previous step. In Morante et al. (2009a) a comparison to an equivalent isolated pipeline system is presented. The results showed a slight decrease in syntactic performance for the joint system but significant improvements in SRL scores.

2.3.3.2 Iterative Models

The Dai et al. (2009) approach is to iteratively run syntactic and SRL parsers. Each iteration extracts features from the previous run. In their system, syntactic parsing is approached by a two-step process. First syntactic heads are identified. Then syntactic functions are assigned. They showed results only for the first two iterations of their method. In most cases these first and second iterations showed only slight improvements over the initial pipeline run.

2.3.3.3 Joint Transition-based Models

Henderson et al. (2008) and Titov et al. (2009) augmented a transition-based dependency parser with new operations that produce both syntactic and semantic structures synchronized at the word level. They use Incremental Sigmoid Belief Networks (Titov and Henderson, 2007a) to train a joint model, which induces latent variable representations to learn correlations between the syntactic and semantic layers. A implementation of this system was presented in the

CoNLL-2008 and CoNLL-2009 Shared Task achieving competitive results, in 2009 they ranked 3rd of 13 teams (Gesmundo et al., 2009). See table 2.3.

2.3.3.4 Graph-based extensions

The Eisner graph-based dependency parser was extended by Lluís et al. (2009), Johansson (2009) and Li et al. (2010) to generate the SRL analysis at the same time that the syntactic tree is being parsed. Sun et al. (2008) presented a different graph-based approach. First, the syntactic heads are identified using the maximum spanning tree algorithm. Then the unlabeled dependency tree is sequentialized to jointly identify semantic arguments while assigning dependency labels using a two-layer Markov model. Finally, semantic role labels are assigned and an ILP post-process is applied.

We presented a system based on the Eisner first and second-order parsers to jointly model syntactic and semantic dependencies (Lluís and Màrquez, 2008; Lluís et al., 2009). The predicate structure is forced to be represented in the same syntactic dependency tree, by enriching arcs with semantic information. A dependency has a syntactic label and as many semantic labels as predicates. The semantic component, however, uses features of pre-computed syntactic structures and thus requires a previous syntactic parse before the joint processing.

Johansson (2009) also extended the Eisner parser. His model is able to exploit features of the syntactic path connecting the predicate and the argument not requiring a previous syntactic parse. At every processed subtree, semantic dependencies between predicates and arguments contained in that subtree are annotated. The author uses an approximate parsing algorithm that employs k-best inference and beam search at a subtree level. His results matched the competitive Titov et al. (2009) results.

Li et al. (2010) compared a pipeline model, a k-best system and also an extension of the Eisner algorithm that jointly integrates SRL. In an analogous approach than Johansson (2009), they jointly process SRL at a subtree level.

Thus when the syntactic bottom-up parser had analyzed a substring then the SRL system is called. As clauses are being syntactically analyzed, the semantic roles for argument and predicates within the same clause are labeled. They showed slight improvements for their Eisner extension with respect to a baseline pipeline system.

2.3.3.5 Shared Features

Collobert and Weston (2008) trained a deep neural network to jointly parse POS tags, syntactic chunks, named entities, semantic roles and semantically related words.

The key point of their proposal is to learn a shared set of features useful for all the *related* tasks. A deep layer of the neural network is expected to automatically learn these shared features. The results of the joint approach regarding SRL were slightly better compared to a baseline SRL system. The SRL system considered as baseline was not a standard pipeline system as no feature engineering nor a set of features from a syntactic parse is used. Experiments with different number of jointly learned tasks showed that increasing the number of tasks was not always correlated with improving the SRL results.

2.3.4 Evaluation of syntactic-semantic systems

A framework to evaluate syntactic-semantic proposals was introduced in the CoNLL-2008 and 2009 Shared Tasks (Surdeanu et al., 2008; Hajič et al., 2009). These Shared Tasks made available dependency treebanks with syntactic dependencies and semantic roles annotated in a dependency format. The tasks were intended to boost the research in these topics.

The task setting was very similar on both the 2008 and 2009 tasks. The first task provided datasets with syntactic dependencies and semantic roles only for the English language. On 2009 datasets were extended for Catalan, Spanish, German, Czech, Chinese and Japanese with annotated predicates also provided for the tests sets. The English language results due to some tokenization and

team	arch	syn LAS	sem F_1	macro F_1
Johansson and Nugues (2008)	k-best	89.3	81.6	85.5
Ciaramita et al. (2008)	pipeline	87.4	78.0	82.7
Che et al. (2008)	pipeline	86.7	78.5	82.7
Zhao and Kit (2008)	pipeline	87.7	76.7	82.2
Henderson et al. (2008)	joint	87.6	73.1	80.5
Lluís and Màrquez (2008)	joint	85.8	70.3	78.1
Johansson (2009)	joint	86.6	77.1	81.1
Henderson et al. (2013)	joint	87.5	76.1	81.8

Table 2.2: Table extracted from Henderson et al. (2013) and Surdeanu et al. (2008) showing systems evaluated with the CoNLL-2008 data. The first systems show CoNLL-2008 results or post-evaluation results. The last two rows show latter publications. The systems are by: Johansson and Nugues (2008), Ciaramita et al. (2008), Che et al. (2008), Zhao and Kit (2008), Henderson et al. (2008), Lluís and Màrquez (2008), Johansson (2009) and Henderson et al. (2013).

#	team	arch	syn avg	syn eng	sem avg	sem eng	joint avg	joint eng
1	Zhao et al. (2009b)	pipeline	85.23	88.48	79.94	85.51	82.64	87.00
2	Zhao et al. (2009a)	k-best	85.04	89.19	79.96	86.15	82.52	87.69
3	Gesmundo et al. (2009)	joint	85.77	88.79	78.42	83.24	83.24	86.03
4	Bohnet (2009)	pipeline	85.65	89.98	76.00	80.39	80.85	85.14
5	Watannabe et al. (2009)	pipeline	81.16	87.70	75.65	84.26	78.43	86.40

Table 2.3: Best CoNLL-2009 systems. Results labeled as *eng* are for the English language. The *avg* columns contain the average scores for all languages. The *syn* columns show the syntactic LAS, *sem* columns show the semantic F_1 and *joint* columns report the macro syntactic-semantic F_1. Systems by: Zhao et al. (2009b), Zhao et al. (2009a), Gesmundo et al. (2009), Bohnet (2009), Watanabe et al. (2009).

codification small changes were not directly comparable between the 2008 and 2009 tasks.

Table 2.2 shows some of the most relevant CoNLL-2008 results including post-evaluation and latter publications on this same dataset. Table 2.3 summarizes the results of the top 5 teams for 2009. Most of the shared task systems approached the problem with a pipeline architecture. There were the exceptions of Gesmundo et al. (2009), Dai et al. (2009), Morante et al. (2009b) and Lluís et al. (2009). The Gesmundo et al. (2009) and Lluís et al. (2009) systems were improvements of the previous year shared task systems by Henderson et al. (2008)

and Lluís and Màrquez (2008) respectively. Dai et al. (2009) and Morante et al. (2009b) contributed with novel approaches. The best joint system of Gesmundo et al. (2009) ranked third globally in the 2009 task. Subsequently, the interest in joint parsing remained and some authors published new approaches or improved the work started at the tasks such as Johansson (2009), Titov et al. (2009) and Henderson et al. (2013). All these proposals, among others were previously classified and reviewed in section 2.3.3.

2.3.5 Remarks and Conclusions

Syntactic and semantic parsing is still mainly approached as a pipeline of tasks. To improve the parsing results, Gildea and Jurafsky (2002) pointed out the possibility of working with many syntactic trees. Latter, Johansson and Nugues (2008) confirmed that by relying on state-of-the-art components a competitive k-best system can be built.

In this thesis we focus on models that are not limited to the k-best syntax. The proposals described under the *joint models* section cover a wide variety of approaches and techniques ranging from *label extensions* to *shared features* models. A major drawback of *label extension* models is the large amount of generated labels. *Iterative models* represent a natural proposal of building a joint system. But unfortunately, the results were mild and the optimization probably could easily reach a local maximum. *Joint transition-based* systems and *joint graph-based* were extensions of common parsing algorithms. These latter approaches avoided the need to train a large number of classifiers.

The small number of joint proposals may be explained in part by some difficulties that arise when designing joint systems such as the divergence between the syntactic and semantic layers. If we look at dependency parsing, standard models crucially depend on an arc factorization of the dependency structure (McDonald et al., 2005a; Nivre and Nilsson, 2005), otherwise their computational properties break. An analogous factorization of syntactic-semantic parsing would allow us to feasibly search over this large space. However, it is challenging to define efficient methods for joint parsing that simultaneously exploit features of both layers.

We will present in chapters 4 and 5 a proposal that formalizes a joint objective function intended to overcome the limitations of the previous graph-based joint models. We solve it with dual-decomposition techniques. Johansson (2009) introduced a proposal based on an approximate search. In constrast, our optimization method gives an exact solution whenever it converges. Lluís

et al. (2009) presented a system where SRL was based on a fixed precomptued syntax. In our case, the SRL component considers many syntactic realizations between predicates and arguments. Furthermore, in chapter 5 we point out a more flexible approach that could be applied to extend our joint model to multiple languages and arbitrary predicate-argument relations.

3

SRL as Assignment

In this chapter we introduce our proposal for inference on semantic role labeling enforcing domain constraints. The method presented in this chapter is latter applied in our joint proposals of chapters 4 and 5. Here our focus is only on SRL. We frame this problem as a linear assignment task on a bipartite graph. We represent the roles to be assigned and the argument candidates as nodes in a graph. Then we find the assignment that maximizes the sum of scores of roles to candidates for a given predicate.

Under this framework we enforce two uniqueness constraints. The first is that an argument candidate receives at most a single role. The second is that no repeated roles are assigned to the arguments of a predicate.

Constraint enforcement has previously been applied in the context of semantic role labeling. E.g., figure 3.1[1] shows the constraints defined in Punyakanok et al. (2004). The labels *A0-A5* represent the core roles. *R-X* and *C-X* are arguments referring or continuing a previously labeled argument. An Integer Linear Programming (ILP) solver enforces these constraints.

Punyakanok et al. (2004) showed experiments enforcing different subsets these constraints with improvements as a larger number of constraints were added. Later, Surdeanu et al. (2007) also evaluated the contribution of different

[1]Extracted from Punyakanok et al. (2004).

1. Arguments cannot cover the predicate except those that contain only the verb or the verb and the following word.
2. Arguments cannot overlap with the clauses (they can be embedded in one another).
3. If a predicate is outside a clause, its arguments cannot be embedded in that clause.
4. No overlapping or embedding arguments.
5. No duplicate argument classes for A0-A5,V.
6. Exactly one V argument per verb.
7. If there is C-V, then there should be a sequence of consecutive V, A1, and C-V pattern. For example, when split is the verb in "split it up", the A1 argument is "it" and C-V argument is "up".
8. If there is an R-X argument, then there has to be an X argument. That is, if an argument is a reference to some other argument X, then this referenced argument must exist in the sentence.
9. If there is a C-X argument, then there has to be an X argument; in addition, the C-X argument must occur after X. This is stricter than the previous rule because the order of appearance also needs to be considered.
10. Given the predicate, some argument classes are illegal (e.g., predicate *stalk* can take only A0 or A1). This linguistic information can be found in PropBank Frames.

Figure 3.1: SRL constraints defined in Punyakanok et al. 2004.

sets of constraints for SRL. They found the best results on the WSJ test set by enforcing not all constrains but some non-overlapping constraints in addition to avoiding repeated core roles.

Punyakanok et al. (2008) confirmed that a constraint-based inference is a simple but effective setting to achieve improvements in SRL systems. In this chapter we present a contribution inspired by this proposal.

The first next section of this chapter briefly introduces the assignment problem. On the following section we describe how we frame semantic role labeling as a linear assignment task. We also review some related work. Then, the experimental section compares for many languages the application of the assignment algorithm with respect to simply choosing the best set of candidates. We conclude this chapter by giving some final remarks. Some sections of this chapter are based on the article of Lluís et al. (2013).

3.1 The Assignment Algorithm

The linear assignment problem is a classical and well-known combinatorial optimization problem (Kuhn, 1955; Burkard et al., 2009). It consists of finding a minimum cost (or maximum) perfect matching in a bipartite graph. It is typically introduced in the literature as the problem of assigning jobs to agents. Every job must be assigned to an agent and no agent should perform more than one job. More precisely, the linear assignment problem is defined as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in A} \sum_{j \in T} C_{ij} x_{ij} \\ \text{subject to} \quad & \sum_{j \in T} x_{ij} = 1 \quad \forall i \\ & \sum_{i \in A} x_{ij} = 1 \quad \forall j \quad , \end{aligned} \tag{3.1}$$

where C is a finite cost matrix typically assumed to be non-negative. Negative costs can be easily handled by simply shifting weights. A is the set of agents, T is the set of tasks, x_{ij} is a boolean variable that represents the assignment of agent i to task j. The two sets of constraints represent that each task is assigned to one agent and that one agent performs a single task.

The Hungarian algorithm (Burkard et al., 2009) solves the linear assignment problem in $O(n^4)$. An improved version runs in $O(n^3)$ (Tomizawa, 1971; Edmonds and Karp, 1972). In contrast, ILP is a NP-hard problem (Karp, 1972) even though at a practical level ILP solvers are remarkably efficient (Schrijver, 1998).

3.2 Framing SRL as an assignment task

Throughout this chapter we define **x** as the input sentence. A sentence has a set of predicates. For example, the sentence from figure 3.2 has two predicates *loves* and *play*. We assume that all predicates are given. A predicate p may

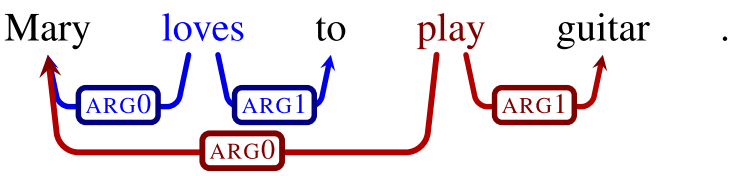

Figure 3.2: A sentence with semantic dependencies for the predicates *loves* and *play*.

have a semantic dependency to token a with role r. In our example the triplet p, a, r may be instantiated with the combination of predicate *play*, argument token *guitar* and role *ARG1*.

For a more compact notation, we will sum up the SRL parse with the indicator vector $\mathbf{z}$. We note as $\mathbf{z}_{p,a,r} = 1$ that the argument a fills the role r for predicate p. We then define the score of the SRL parse as the sum of local scores as follows,

$$s_{\text{srl}}(\mathbf{x}, \mathbf{z}) = \sum_{\mathbf{z}_{p,a,r}=1} s_{\text{srl}}(\mathbf{x}, p, a, r) \quad . \tag{3.2}$$

Here we overloaded the s_{srl} function and assumed that it decomposes in a sum of scores computed by some domain classifiers.

SRL constraints Under the dependency framework of the CoNLL-2008 Shared Task datasets (Surdeanu et al., 2008) roles are assigned directly to nodes of the dependency tree. As a consequence, this representation of role labels already enforces some constraints. Role labels are tied to syntactic nodes that represent subtrees. Any two subtrees cannot share any word except if one is the ancestor of the other. Therefore, partial non-overlapping is already guaranteed, however, embedding or complete overlapping is still possible.

Considering some of the most relevant constraints pointed out by Surdeanu et al. (2007) we enforce only:

1. **cRole**. No repeated roles for a given predicate are allowed.

2. **cArg**. At most one role is assigned to any given token.

Therefore our inference problem is stated as follows,

$$
\begin{aligned}
&\underset{\mathbf{z}}{\operatorname{argmax}} \;\; s_{\text{srl}}(\mathbf{x}, \mathbf{z}) \\
&\text{subject to} \\
&\quad \textbf{cRole}: \quad \forall p, r: \sum_{a} \mathbf{z}_{p,a,r} \leq 1 \\
&\quad \textbf{cArg}: \quad \forall p, a: \sum_{r} \mathbf{z}_{p,a,r} \leq 1 \quad .
\end{aligned}
\tag{3.3}
$$

As previously introduced, the constraint $\mathbf{cRole}$ states that the solution has no repeated roles. $\mathbf{cArg}$ enforces that an argument fills at most a role. In general, only a small percentage of the predicates have repeated roles. Table 3.1 of the experimental section shows the percentage of repeated roles in our datasets.

Syntactic Paths As usual, we assume that SRL scorers (i.e., our s_{srl} functions) have access to the syntactic dependency tree. Thus syntactic paths from predicates to arguments are implicitly available even thought we omitted them from Eq. (3.3) for clarity reasons. In the next chapter, on section 4.2 we will show that it is straightforward to extend the assignment approach in the case where we have multiple paths connecting predicates to their arguments. Furthermore, in chapter 5 a different method based on shortest-path algorithms will generate a set of syntactic paths and we will show how the assignment approach is flexible enough to accommodate a large number of syntactic paths.

3.2.1 The Assignment Algorithm

Coming back to solving Eq. (3.3), it is easy to see that an optimal solution satisfying constraints **cRole** and **cArg** can be found with a linear assignment algorithm. Our method determines the predicate-argument relations separately for each predicate. We define a bipartite graph as follows. On one side of the graph we add k nodes $r_1 \ldots r_k$ for each *role*. On the other side of the graph, we add n nodes $a_1 \ldots a_n$ for each *argument*. Let N be the sum of n plus k. We add

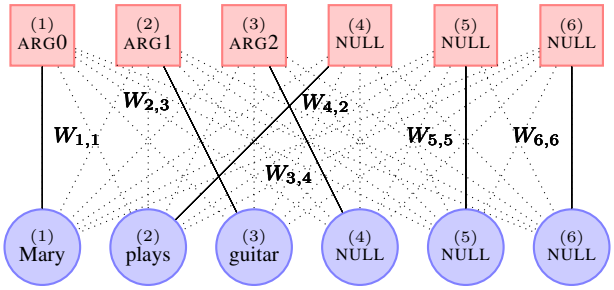

Figure 3.3: Illustration of the assignment graph for the sentence "Mary plays guitar", where the predicate *plays* can have up to three roles: ARG0 (agent), ARG1 (theme) and ARG2 (benefactor). Nodes labeled NULL represent a null role or token. Highlighted edges represent the correct assignment.

$a_{n+1}, \ldots, a_N$ additional null arguments and $r_{k+1}, \ldots, r_N$ additional null roles. The bipartite graph is of size $2N$. 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 $f : i \rightarrow j$ from roles to arguments that maximizes $\sum_{i=1}^{N} W_{i,f(i)}$.

We 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 3.3 illustrates an example of a bipartite graph. We then 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. (3.3).

We assume that we have the constraint that for a predicate there is at most a single instance of a role. It is simple to allow for a fixed number of multiple instances of a role by adding more role nodes in the step 1 of the algorithm. In addition, it would be straightforward to add penalties in step 3 for multiple instances of a role.

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

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 $1 \leq i \leq k, 1 \leq j \leq n$
   $S_{i,j} = \max_{\boldsymbol{\pi}^{p,a_j,r_i}} s_{\text{srl}}(\mathbf{x}, p, a_j, r_i, \boldsymbol{\pi}^{p,a_j,r_i})$
   (b) Scores of NULL assignments[2]:
   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 $1 < i \leq N,\ \ n < j \leq N$
   $W_{i,j} = 0$

3.3 Related Work

As previously introduced, Punyakanok et al. (2004) and Punyakanok et al. (2008) presented a system that explicitly controls semantic role constraints as well as other constraints that look at pairwise assignments which we cannot model. They solve SRL using general-purpose integer linear programming 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. Shen and Lapata (2007) find assignments between tokens and roles to improve a question answering system by measuring the semantic similarity between the query and document sentences. 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. Täckström et al. (2015) enforced non-overlapping of the arguments of a predicate and unique roles in a similar spirit that the approach that we describe in this chapter. In addition the authors pointed out the possibility of extending the model to control for continuation and reference roles. Their proposal relies on dynamic programming and the model was trained by using inside-outside methods. The work presented a probabilistic model for SRL that enforces some constraints in a more efficient way than previous ILP approaches. Our proposal, first published in Lluís et al. (2013), frames SRL as a linear assignment task, for which a simple and exact algorithm exists.

3.4 Experiments

We present experiments using the CoNLL-2009 Shared Task datasets (Hajič et al., 2009). For the case of the English language, it consists of the usual WSJ training/development/test sections mapped to dependency trees, augmented with semantic predicate-argument relations from PropBank. Further details about these datasets can be found in appendix C.

In all cases we run a syntactic parser and then our SRL system. All configurations thus represent a pipeline approach to syntactic-semantic parsing. Our goal in this experimental section is to evaluate the contribution of the assignment constraint enforcement in the following setting:

Languages. We give results for different datasets: *English* verbal predicates, *German, Czech* and *Spanish*.

Assignment. We enforce the previously defined assignment constraints for all systems from table 3.2 labeled as *yes* on the *assignment* column. Otherwise, the baseline systems labeled as *assignment no* simply select the highest ranked set of roles for the argument candidates. These inference methods apply both to training and testing.

Candidate pruning. We use the Xue and Palmer (2004) filtering rule adapted to a dependency-based representation. This rule constraints the search of argument candidates to direct descendants of the predicate or direct descendants of the predicate ancestors. This heuristic was originally devised for the English language treebanks. The configurations labeled as *ancestor* enforce this rule over all candidates. Otherwise, systems labeled as *all* explore all possible candidates thus any sentence token could potentially fill a role for a given predicate.

We provide details about the implementation in the next section. Then we present and discuss the results.

3.4.1 Implementation

All our configurations use the same type of discriminative scorers and features. We train all our SRL classifier using an averaged Perceptron (Collins, 2002). We take a structured prediction approach, i.e., we only make corrections or updates on the Perceptron after we run our assignment inference method over the output of the SRL classifiers.

Syntactic model We reimplemented the McDonald et al. (2005a) syntactic dependency parser based on the Eisner (2000) algorithm. To learn the models, we use a log-linear loss function following Koo et al. (2007), which trains probabilistic discriminative parsers. At test time, we use the probabilistic parsers to compute marginal probabilities $p(h, m, l \mid \mathbf{x})$, using inside-outside algorithms for first and second-order models. Hence, for either of the parsing models, we always obtain a table of arc-factored marginal scores, with one score per la-

beled dependency. We show experiments using first-order, second-order and also gold syntax. The syntactic LAS results are shown tables 3.3 and 3.5.

SRL model Our SRL features are based in the work of Johansson (2009). Johansson lists and categorizes a set of common features for SRL. A distinction is made between *primary*, *secondary* and *interdependency* features. *Primary* and *secondary* features can be directly computed from the input sentence $\mathbf{x}$ or from local information. In contrast, *interdependency* features extract information from both $\mathbf{x}$ and the syntactic parse tree. This distinction is relevant for joint syntactic-semantic systems as the syntactic tree may not be completely available at the time that these features are required.

Primary Features can be directly computed from the unparsed input. These features are the same features as in syntactic dependency parsing. But in this case are representing semantic arcs instead of syntactic ones.

Secondary Features are *local* features that can be computed without accessing the complete syntactic tree. These features are standard for SRL. The *label* feature is extracted from the syntactic function that is realizing the argument candidate with respect to its syntactic parent.

- Predicate word (e.g., play)
- Predicate POS (e.g., NN)
- Argument word
- Argument POS
- Pred. + arg. words (e.g., play+at)
- Predicate word + label (e.g., play+OBJ)
- Predicate POS + label (e.g., VB+OBJ)
- Argument word + label
- Argument POS + label

- Pred. + arg. words + label (e.g., play+at+OBJ)

Interdependency Features. To compute these features the syntactic path between the predicate and the candidate argument must be known. These features are hard to compute when jointly parsing syntax and semantics as the complete path may not be available at that time.

- Path
- Path + arg. POS
- Path + pred. POS
- Path + arg. word
- Path + pred. word
- Path + label
- Path + arg. POS + label
- Path + pred. POS + label
- Path + arg. word + label
- Path + pred. word + label

The *path* feature represents the syntactic relationship between the predicate and candidate argument and it is coded as the set of syntactic function labels concatenated with the direction of the relation (e.g., the path from *play* to *Mary* in figure 1.3 is IM+↑+OPRD+↑+SBJ+↓).

In addition to the previous features, we included the following:

- Unigram/bigram/trigram of the path. For all n-grams in the syntactic path, patterns of words and POS tags (e.g., from mary-loves-to-play we extract mary+loves+to, loves+to+play, 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 with respect to the sentence word order in a predicate-argument path.

3.4.2 Results

We evaluate semantic role labeling with precision recall and F_1 over the labeled semantic dependencies. Our evaluation metrics are based on the official CoNLL-2009 scorer which considers predicate senses as special semantic dependencies and, thus, it includes them in the calculation of the scores. Here, as we are not addressing predicate sense disambiguation, we ignore predicate senses when presenting the evaluation results. Therefore, when we report the performance of the CoNLL systems (e.g., Gesmundo et al. (2009), Zhao et al. (2009b) in table 3.2), their scores will be in general noticeably lower than the scores published at the shared task. This is because predicate sense disambiguation is usually a simple task with a very high baseline. E.g., around 90% of the English language predicate senses can be correctly identified by choosing the most frequent sense.

We first look at the number of repeated roles. Table 3.1 shows the most common repeated role labels for the English, German, Czech and Spanish training datasets. For each dataset we first show the total number of repeated roles per predicate. E.g., English has 0.87% of repeated roles. Then we give the ordered list of the most frequent repeated roles. For example, for the English language the *AM-TMP* role is the most repeated label representing the 31.99% of all repeated roles. We provide as additional detail the number of times (1, 2, or $\geq$3) that each role is repeated. For example, the German role label *A2* is repeated 1 time, i.e., it appears twice for a predicate, in the 47.9% of the cases of repetition.

We observe in this table that Czech exhibits the largest amount of repeated roles with an 11.08%. The distribution shows a large tail of tags and in many cases we can find 3 or more repetitions of the same role. In contrast, it is unlikely to find more than two repetitions of any given role in the Spanish datasets.

Table 3.2 shows the results on the development set for our different configurations. We again first show the percentage of repeated roles for each dataset,

% rep.	label	num. of rep. 1	2	≥3	% rep.	label	num. of rep. 1	2	≥3
(0.87%)	**English**				**(1.05%)**	**German**			
31.998	AM-TMP	29.501	2.497	0.000	59.557	A1	47.922	8.310	3.324
16.260	A1	13.240	1.916	1.103	16.620	A0	16.066	0.554	0.000
12.544	AM-ADV	11.469	0.987	0.087	16.066	A2	12.742	3.324	0.000
7.869	A2	6.243	0.929	0.697	5.263	A3	4.709	0.554	0.000
6.185	A3	5.662	0.290	0.232	1.662	A4	1.662	0.000	0.000
5.865	AM-LOC	5.168	0.348	0.348	0.831	A5	0.831	0.000	0.000
5.575	A0	5.226	0.348	0.000					
5.139	AM-MNR	4.936	0.116	0.087					
3.020	AM-DIS	3.020	0.000	0.000					
1.626	AM-DIR	1.452	0.000	0.174					
1.132	C-A1	1.132	0.000	0.000					
0.842	AM-MOD	0.842	0.000	0.000					
0.581	A4	0.436	0.000	0.145					
0.319	AM-PNC	0.319	0.000	0.000					
0.145	AM-CAU	0.145	0.000	0.000					
(11.08%)	**Czech**				**(1.74%)**	**Spanish**			
55.133	RSTR	39.093	12.260	3.780	30.805	argM-adv	27.678	3.127	0.000
12.492	PAT	7.341	2.342	2.809	26.867	argM-tmp	25.478	1.390	0.000
11.309	ACT	6.189	2.323	2.797	12.855	arg1-pat	12.623	0.232	0.000
3.259	LOC	2.303	0.499	0.457	9.728	arg2-ben	9.728	0.000	0.000
3.076	TWHEN	2.567	0.445	0.064	7.701	argM-loc	7.470	0.232	0.000
2.804	APP	1.856	0.568	0.381	3.358	arg1-tem	3.358	0.000	0.000
1.870	EFF	1.282	0.415	0.173	1.911	arg2-atr	1.911	0.000	0.000
1.297	ID	0.556	0.262	0.479	1.621	arg3-ben	1.621	0.000	0.000
0.961	MANN	0.778	0.124	0.059	1.158	argM-mnr	1.042	0.116	0.000
0.766	EXT	0.487	0.010	0.269	1.158	arg0-agt	1.158	0.000	0.000
0.734	ACMP	0.447	0.163	0.124	0.926	argM-cau	0.926	0.000	0.000
0.702	DIR3	0.413	0.148	0.141	0.753	arg2-exp	0.753	0.000	0.000
0.628	ADDR	0.405	0.119	0.104	0.290	arg2-loc	0.290	0.000	0.000
0.600	BEN	0.432	0.114	0.054	0.290	arg2-null	0.290	0.000	0.000
0.516	MEANS	0.314	0.099	0.104	0.232	argM-fin	0.232	0.000	0.000
0.482	CAUS	0.346	0.089	0.047	0.116	argL-null	0.116	0.000	0.000
0.413	MAT	0.247	0.089	0.077	0.058	argM-atr	0.058	0.000	0.000
0.413	AIM	0.269	0.104	0.040	0.058	arg3-exp	0.058	0.000	0.000
0.373	REG	0.274	0.069	0.030	0.058	arg1-ext	0.058	0.000	0.000
0.351	DIR1	0.217	0.054	0.079	0.058	argM-ext	0.058	0.000	0.000
0.331	COND	0.259	0.049	0.022					
0.175	CRIT	0.124	0.044	0.007					
0.153	COMPL	0.089	0.054	0.010					
0.148	ORIG	0.104	0.030	0.015					

Table 3.1: Percentage of repeated roles per predicate in the *English, German, Czech and Spanish* CoNLL-2009 training sets.

syntax	**candidates**	**assignment**	**English** F_1 prec/rec	**German** F_1 prec/rec	**Czech** F_1 prec/rec	**Spanish** F_1 prec/rec
% of rep.	all		1.01	1.20	11.19	2.03
gold	ancestor	yes	84.80 86.73/82.95	72.42 76.25/68.95	61.25 88.60/50.70	80.08 83.36/76.99
gold	ancestor	no	84.64 87.17/82.25	68.91 74.53/64.07	68.07 86.26/56.22	81.25 84.76/78.02
gold	all	yes	85.15 85.81/84.49	77.18 80.06/74.51	66.52 71.25/62.39	80.08 83.36/76.99
gold	all	no	84.93 87.05/82.91	71.78 78.01/66.47	72.98 83.50/64.81	81.25 84.76/78.02
first order						
predicted	ancestor	yes	78.69 82.51/75.21	68.64 72.28/65.36	60.72 76.98/50.13	71.87 74.22/69.66
predicted	ancestor	no	78.44 82.79/74.53	66.64 71.88/62.10	68.78 86.10/57.26	72.54 74.65/70.53
predicted	all	yes	78.97 81.45/76.63	68.86 72.65/65.44	65.18 70.16/60.86	71.87 74.22/69.66
predicted	all	no	78.94 83.34/74.98	66.67 71.28/62.62	71.94 81.84/64.17	72.54 74.65/70.53
second or.						
predicted	ancestor	yes	80.97 84.41/77.79	70.40 73.15/67.84	60.59 75.44/50.62	72.94 73.81/72.09
predicted	ancestor	no	80.67 85.04/76.73	68.87 73.10/65.10	67.64 84.21/56.51	74.05 75.06/73.06
predicted	all	yes	81.51 83.69/79.44	69.71 73.08/66.64	65.66 68.37/63.15	72.94 73.81/72.09
predicted	all	no	81.03 85.24/77.21	68.58 73.44/64.33	73.16 79.98/67.42	74.05 73.81/72.09

Table 3.2: Results for semantic precision, recall and F_1 on the development set for *English verbal, German, Czech* and *Spanish* for assignment and non-assignment configuration and for ancestor and no pruning. As *Spanish* arguments are all syntactic direct dependants of their predicates results are identical for *all* and *ancestor configurations.*

syntax	English	German	Czech	Spanish
first order	86.18	83.64	79.51	85.58
second order	88.55	85.58	81.31	88.25

Table 3.3: Syntactic LAS results for the previous table 3.2 on the development set for *English, German, Czech* and *Spanish*.

syntax	assignment	English F_1 prec/rec	German F_1 prec/rec	Czech F_1 prec/rec	Spanish F_1 prec/rec
Zhao et al.		82.60	74.46	77.59	77.68
(2009b)		86.21/79.29	76.32/72.69	83.88/72.18	81.42/74.26
Zhao et al.		83.97	74.65	71.83	77.33
(2009a)		86.91/81.22	77.14/72.32	73.82/69.93	80.68/74.26
Gesmundo		81.07	69.55	75.77	73.62
et al. (2009)		83.45/78.83	75.46/64.49	78.56/73.18	76.80/70.70
second or.	yes	82.84	71.76	65.65	73.25
		84.97/80.81	74.03/69.62	68.41/63.10	74.21/72.31
second or.	no	82.60	70.13	75.70	74.46
		86.43/79.09	73.23/67.29	80.92/71.11	75.39/73.56
candidates		all	ancestor	all	all

Table 3.4: Results for semantic precision, recall and F_1 on test set for *English verbal, German, Czech* and *Spanish*. We evaluate enforcing assignment and non-assignment constraints for the best configuration from the development set. These best configurations use second-order syntax and the candidate filtering rule is shown in the last row. We compare to the Gestmundo et al. 2009, Zhao et al. 2009a and Zhao et al. 2009b systems.

system	English	German	Czech	Spanish
Zhao et al. (2009b)	85.50	85.93	78.46	86.20
Zhao et al. (2009a)	89.19	86.24	79.70	86.29
Gesmundo et al. (2009)	88.79	87.29	80.38	87.64
second order	90.21	86.54	80.86	88.21

Table 3.5: Syntactic LAS results for the systems presented in the previous table 3.4 for the test set for *English verbal, German, Czech* and *Spanish*. The Zhao et al. (2009b) system used syntax provided by the CoNLL-2009 organizers.

that is the recall upper bound. Note that here we are enforcing unique roles for all kind of role labels. In contrast, Punyakanok et al. (2004) only enforced unique roles regarding the core arguments.

Results are grouped by the syntax used. As expected, we observe in general better SRL results as we move from first-order syntax to second-order. Results further improve when we switch to gold syntax. In the case of the *Czech* datasets, probably due to the recall penalization of the assignment algorithm, a better syntax is not always correlated with better results.

The second column of the table shows the filtering strategy that we applied. We observe that the pruning rule of Xue and Palmer (2004) penalizes the recall specially in the Czech dataset. All arguments for the Spanish dataset are direct syntactic descendants of the predicate. Thus for this last dataset the results are identical for the *all* and *ancestor* configurations. For the English language, when we are not applying the pruning strategy we capture in some configurations a slightly larger number of arguments. This slight increase in recall comes at a cost of larger computing times.

The third column indicates whether or not we are applying the assignment approach. The *German* dataset shows consistent improvement across all tested configurations when uniqueness constraints are enforced. In the case of the English language we also observe some improvements when the assignment framework is applied. These improvements are smaller than in the previous case.

Spanish consistently suffers for about 1 point decrease in F_1 when assignment is applied. Czech is severely penalized when enforcing uniqueness constraints. These two last datasets show the largest number of repeated roles.

We finally show on table 3.4 our best development configurations for the test set. As a reference, we included the results of Zhao et al. (2009b), Zhao et al. (2009a), and Gesmundo et al. (2009). The fist was the top performing shared task system, considering the averaged SRL results. The second system achieved the best English SRL results. The third was the best joint system and

globally third of the task.

As reported in the development results, we observe that the *German* and *English* datasets benefit from our assignment approach. In contrast *Czech* and *Spanish* show a decrease in all semantic scores when unique roles are enforced.

Enforcing consistency constraints may limit the potential recall of some classifiers. In the cases where the datasets contain the largest number of repeated roles the benefits will probably be smaller. Also and as classifiers achieve better results, such as the case for the English language, we tend to observe diminishing gains when enforcing these constraints.

3.5 Remarks

We have introduced an efficient method to control linear assignment constraints in the predicate-argument structure. The Hungarian $O(n^3)$ algorithm is fast to implement and offers some flexibility to introduce few other domain constraints such as a fixed maximum number of repeated roles. Our approach is an alternative to the method from Punyakanok et al. (2004) to enforce a small subset of the most relevant constraints.

We have shown results for many CoNLL-2009 Shared Task languages. We first observe that these datasets do not contain unique roles per predicate to some extent. Even thought, in some of these case we have seen improvements in the semantic results.

Enforcing uniqueness adds a small overhead of $\sim$8% when processing the English datasets. However, in other cases such as the Czech it speeds-up the training times by reducing the number of updates in the classifiers.

We believe that when the margins for improvement in the classifiers are larger we may obtain a clearer benefit by enforcing uniqueness. In addition, in some cases such as when we work with smaller datasets, these constraints may turn learning curves steeper. These two last hypothesis are left to be proved in future work.

We applied the assignment approach in the context of pipeline systems. We will show in the following chapters how to extend this approach when we work with a larger set of argument candidates generated by our joint proposals.

4

Dual-decomposition Joint Syntactic-Semantic Parsing

One of the main difficulties in designing joint syntactic-semantic systems is that there exist important structural divergences between the syntactic and semantic layers. This can be seen in the example in figure 4.1 where *Mary* is the *ARG0* of *play*. We highlighted the syntactic and semantic dependencies between *Mary* and *play*. These two words are semantically directly related but syntactically distant.

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., 2005a; 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 chapter we introduce our proposal for a joint syntactic-semantic model. The contents of this chapter are extracted from Lluís et al. (2013). 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 syn-

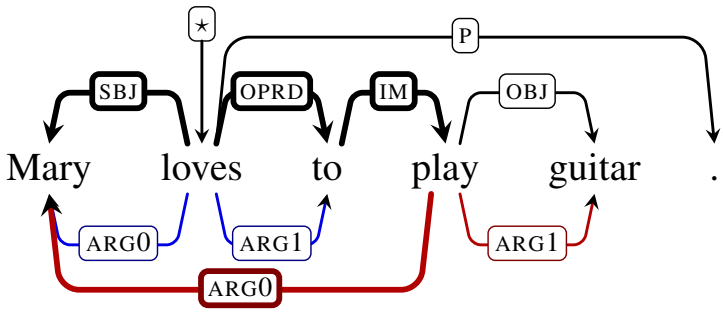

Figure 4.1: A sentence with syntactic dependencies (top) and semantic dependencies (bottom) for the predicates *loves* and *play*. The thick arcs illustrate a structural divergence where the argument *Mary* is linked to *play* with a path involving three syntactic dependencies.

tactic path that connects the predicate with the argument. 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 contribution of this chapter is to to solve the 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 the 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 results of the best joint method of the CoNLL-2009 task.

4.1 A Syntactic-Semantic Dependency Model

We first describe how we represent structures of syntactic and semantic dependencies like the one in figure 4.1. Following the notation introduced in the previous chapters, we assume a fixed input sentence $\mathbf{x}$ with n tokens where lexical predicates are marked. We further assume a fixed set of syntactic functions

$\mathcal{R}_{\text{syn}}$ and semantic roles $\mathcal{R}_{\text{sem}}$. We represent dependency structures using vectors of binary variables. The variable $\mathbf{y}_{h,m,l}$ indicates the presence of a syntactic dependency from head token h to dependant token m labeled with the syntactic function l. Then, a syntactic tree is denoted as a vector $\mathbf{y}$ indexed by syntactic dependencies. Similarly, a variable $\mathbf{z}_{p,a,r}$ indicates the presence of a semantic dependency between predicate token p and argument token a labeled with semantic role r. We 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 the 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 $r \in \mathcal{R}_{\text{sem}}$.

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

$$\underset{\mathbf{y},\mathbf{z}}{\text{argmax}} \;\; s_{\text{syn}}(\mathbf{x}, \mathbf{y}) + s_{\text{srl}}(\mathbf{x}, \mathbf{z}, \mathbf{y}) \quad . \tag{4.1}$$

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

$$s_{\text{syn}}(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{y}_{h,m,l}=1} s_{\text{syn}}(\mathbf{x}, h, m, l) \quad , \tag{4.2}$$

where we overload s_{syn} to be a function that computes scores for individual syntactic dependencies. In linear discriminative models one has $s_{\text{syn}}(\mathbf{x}, h, m, l) = \mathbf{w}_{\text{syn}} \cdot \mathbf{f}_{\text{syn}}(\mathbf{x}, h, m, l)$, 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., 2005a). In Section 4.5 we describe how we train score functions with discriminative methods.

The other term in Eq. (4.1), $s_{\text{srl}}(\mathbf{x}, \mathbf{z}, \mathbf{y})$, gives a score for a semantic dependency structure $\mathbf{z}$ using features from the syntactic tree $\mathbf{y}$. 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, as introduced in

chapter 2 this property makes the optimization problem computationally hard. In the rest of the chapter we describe a method that searches over the 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 our case we employ this restriction. Figure 4.1 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: $\boldsymbol{\pi}^{p,a,r}_{h,m,l}$ indicates if a dependency $\langle h, m, l\rangle$ is part of the syntactic path that links predicate p with token a for 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

$$s_{\text{srl}}(\mathbf{x}, \mathbf{z}, \boldsymbol{\pi}) = \sum_{\mathbf{z}_{p,a,r}=1} s_{\text{srl}}(\mathbf{x}, p, a, r, \boldsymbol{\pi}^{p,a,r}) \quad , \tag{4.3}$$

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.

[1] 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 4.2.

The inference problem of our joint model is formalized as:

$$\begin{aligned}
&\underset{\mathbf{y},\mathbf{z},\boldsymbol{\pi}}{\operatorname{argmax}} \; s_{\text{syn}}(\mathbf{x},\mathbf{y}) + s_{\text{srl}}(\mathbf{x},\mathbf{z},\boldsymbol{\pi}) \qquad (4.4)\\
&\text{subject to}\\
&\textbf{cTree}: \quad \mathbf{y} \text{ is a valid dependency tree}\\
&\textbf{cRole}: \quad \forall p, r: \sum_a \mathbf{z}_{p,a,r} \leq 1\\
&\textbf{cArg}: \quad \forall p, a: \sum_r \mathbf{z}_{p,a,r} \leq 1\\
&\textbf{cPath}: \quad \forall p, a, r: \text{if } \mathbf{z}_{p,a,r} = 1 \text{ then}\\
&\qquad \boldsymbol{\pi}^{p,a,r} \text{ is a path from } p \text{ to } a,\\
&\qquad \text{otherwise } \boldsymbol{\pi}^{p,a,r} = \mathbf{0}\\
&\textbf{cSubtree}: \quad \forall p, a, r: \boldsymbol{\pi}^{p,a,r} \text{ is a subtree of } \mathbf{y} \quad .
\end{aligned}$$

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. 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.

We first review in the next section how we find the paths between predicates and arguments. Then, section 4.3 describes a dual-decomposition method that iteratively runs a syntactic parser and an SRL analyzer to solve the joint problem under the previous constraints.

4.2 Local Optimization of Syntactic Paths

Let $\widehat{\mathbf{z}}$ and $\widehat{\boldsymbol{\pi}}$ be the optimal values of Eq. (4.3). For any $\langle p, a, r \rangle$, let

$$\tilde{\boldsymbol{\pi}}^{p,a,r} = \underset{\boldsymbol{\pi}^{p,a,r}}{\operatorname{argmax}} \; s_{\text{srl}}(\mathbf{x}, p, a, r, \boldsymbol{\pi}^{p,a,r}) \quad . \tag{4.5}$$

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. (4.3) 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. (4.5).

In this chapter, 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. (4.5) 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.

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.

4.3 A Dual-Decomposition Algorithm

Dual-decomposition methods (Rush and Collins, 2012) allow to decompose an optimization problem into subproblems that can be solved in an efficient way. Typically the problem will be of the form

$$\begin{aligned} &\text{minimize } f(x, y) + g(v, w) \\ &\text{subject to } h(y, w) = 0 \quad . \end{aligned} \tag{4.6}$$

The subproblems $f(x, y)$ and $g(v, w)$ are assumed to be efficiently solvable, but some complicating or agreement constraints $h(y, w)$ must be enforced typ-

ically by an iterative process.

We present a dual-decomposition method to optimize Eq. (4.4). We use as a subroutine the assignment approach presented in chapter 3. 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) and Rush et al. (2010).

We note that in Eq. (4.4) 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

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

These constraints can be compactly written as

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

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 $\boldsymbol{\xi}_{h,m,l}$, turning the constraints into:

$$c \cdot \mathbf{y}_{h,m,l} - \sum_{p,a,r} \boldsymbol{\pi}^{p,a,r}_{h,m,l} - \boldsymbol{\xi}_{h,m,l} = 0 \quad \forall\, h, m, l \quad .$$

We can now rewrite Eq. (4.4) as:

$$\begin{aligned} &\underset{\mathbf{y},\mathbf{z},\boldsymbol{\pi},\boldsymbol{\xi}\geq\mathbf{0}}{\operatorname{argmax}} \ \ s_{\text{syn}}(\mathbf{x},\mathbf{y}) + s_{\text{srl}}(\mathbf{x},\mathbf{z},\boldsymbol{\pi}) \\ &\quad \text{subject to} \\ &\quad \textbf{cTree}, \textbf{cRole}, \textbf{cArg}, \textbf{cPath} \\ &\quad \forall h, m, l : c \cdot \mathbf{y}_{h,m,l} - \sum_{p,a,r} \boldsymbol{\pi}^{p,a,r}_{h,m,l} - \boldsymbol{\xi}_{h,m,l} = 0 \quad . \end{aligned} \tag{4.7}$$

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 $\boldsymbol{\lambda}_{h,m,l}$ is a Lagrange multiplier for the constraint associated with $\langle h, m, l\rangle$. The Lagrangian of the problem is:

$$\begin{aligned} L(\mathbf{y}, \mathbf{z}, \boldsymbol{\pi}, \boldsymbol{\xi}, \boldsymbol{\lambda}) &= s_{\text{syn}}(\mathbf{x}, \mathbf{y}) + s_{\text{srl}}(\mathbf{x}, \mathbf{z}, \boldsymbol{\pi}) \\ &\quad + \boldsymbol{\lambda} \cdot \left(c \cdot \mathbf{y} - \sum_{p,a,r} \boldsymbol{\pi}^{p,a,r} - \boldsymbol{\xi} \right) \quad . \end{aligned} \tag{4.8}$$

We can now formulate Eq. (4.7) as:

$$\max_{\substack{\mathbf{y},\mathbf{z},\boldsymbol{\pi},\boldsymbol{\xi}\geq\mathbf{0} \\ \text{s.t. } \textbf{cTree},\textbf{cRole},\textbf{cArg},\textbf{cPath} \\ c\cdot\mathbf{y}-\sum_{p,a,r}\boldsymbol{\pi}^{p,a,r}-\boldsymbol{\xi}=\mathbf{0}}} L(\mathbf{y}, \mathbf{z}, \boldsymbol{\pi}, \boldsymbol{\xi}, \boldsymbol{\lambda}) \quad . \tag{4.9}$$

This optimization problem has the property that its optimal values are the same as the optimal of Eq. (4.7) 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. (4.9) we obtain the dual objective:

$$\begin{aligned} D(\boldsymbol{\lambda}) &= \max_{\substack{\mathbf{y},\mathbf{z},\boldsymbol{\pi},\boldsymbol{\xi}\geq\mathbf{0} \\ \text{s.t. } \textbf{cTree},\textbf{cRole},\textbf{cArg},\textbf{cPath}}} L(\mathbf{y}, \mathbf{z}, \boldsymbol{\pi}, \boldsymbol{\xi}, \boldsymbol{\lambda}) \\ &= \max_{\mathbf{y} \text{ s.t. } \textbf{cTree}} \left(s_{\text{syn}}(\mathbf{x}, \mathbf{y}) + c \cdot \mathbf{y} \cdot \boldsymbol{\lambda} \right) \\ &\quad + \max_{\substack{\mathbf{z},\boldsymbol{\pi} \\ \text{s.t. } \textbf{cRole},\textbf{cArg},\textbf{cPath}}} \left(s_{\text{srl}}(\mathbf{x}, \mathbf{z}, \boldsymbol{\pi}) - \boldsymbol{\lambda} \cdot \sum_{p,a,r} \boldsymbol{\pi}^{p,a,r} \right) \\ &\quad + \max_{\boldsymbol{\xi}\geq\mathbf{0}} \left(-\boldsymbol{\lambda} \cdot \boldsymbol{\xi} \right) \quad . \end{aligned} \tag{4.10}$$

Subgradient descent method Gradient descent algorithms are standard techniques to solve optimization problems (Rush and Collins, 2012). These iterative algorithms can be defined by the following rule:

$$x_{n+1} = x_n - \alpha \nabla F(x_n) \quad ,$$

where x_{n+1} and x_n are a converging sequence of solution points, α is the step size and $\nabla F(x)$ is the gradient of F at x. For convex differentiable functions

the method above converges to the global optimal points of F.

Sometimes the function is not differentiable but we analogously define the subgradient descent method. The subgradient v of a function f is any vector that satisfies:

$$f(x) - f(x_0) \geq v(x - x_0) \quad .$$

The dual objective is an upper bound to the optimal value of primal objective of Eq. (4.7). Thus, we are interested in finding the minimum of the dual in order to tighten the upper-bound. There, we solve

$$\min_{\boldsymbol{\lambda}} D(\boldsymbol{\lambda}) \tag{4.11}$$

using a subgradient method. Algorithm 2 presents the pseudo-code of the method. The algorithm takes advantage of the *decomposed* form of the dual in Eq. (4.10), where we have rewritten the Lagrangian such that syntactic and semantic structures appear in separate terms. This allows us to compute subgradients efficiently. In particular, the subgradient of D at a point $\boldsymbol{\lambda}$ is:

$$\Delta(\boldsymbol{\lambda}) = c \cdot \widehat{\mathbf{y}} - \sum_{p,a,r} \widehat{\boldsymbol{\pi}}^{p,a,r} - \widehat{\boldsymbol{\xi}} \quad , \tag{4.12}$$

where

$$\widehat{\mathbf{y}} = \underset{\mathbf{y} \text{ s.t. } \textbf{cTree}}{\operatorname{argmax}} \left(s_{\text{syn}}(\mathbf{x}, \mathbf{y}) + c \cdot \mathbf{y} \cdot \boldsymbol{\lambda} \right) \tag{4.13}$$

$$\widehat{\mathbf{z}}, \widehat{\boldsymbol{\pi}} = \underset{\substack{\mathbf{z},\boldsymbol{\pi} \text{ s.t.} \\ \textbf{cRole},\textbf{cArg},\textbf{cPath}}}{\operatorname{argmax}} \quad s_{\text{srl}}(\mathbf{x}, \mathbf{z}, \boldsymbol{\pi}) - \boldsymbol{\lambda} \cdot \sum_{p,a,r} \boldsymbol{\pi}^{p,a,r} \tag{4.14}$$

$$\widehat{\boldsymbol{\xi}} = \underset{\boldsymbol{\xi} \geq \mathbf{0}}{\operatorname{argmax}} \; -\boldsymbol{\lambda} \cdot \boldsymbol{\xi} \quad . \tag{4.15}$$

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 $\boldsymbol{\lambda}_{h,m,l}$ 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 in previous work, in the algorithm a parameter α_t controls the size of

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**

1: $\boldsymbol{\lambda}^1 = \mathbf{0}$ # initialize dual variables
2: c =number of distinct $\langle h, m, l\rangle$ in $\mathbf{x}$
3: **for** $t = 1 \ldots T$ **do**
4: $\quad \widehat{\mathbf{y}} = \text{argmax}_{\mathbf{y} \text{ s.t. } \textbf{cTree}} \left(s_{\text{syn}}(\mathbf{x}, \mathbf{y}) + c \cdot \boldsymbol{\lambda}^t \cdot \mathbf{y}\right)$
5: $\quad \widehat{\mathbf{z}}, \widehat{\boldsymbol{\pi}} = \text{argmax}_{\substack{\mathbf{z},\boldsymbol{\pi} \\ \text{s.t. } \textbf{cSem}}} \left(s_{\text{srl}}(\mathbf{x}, \mathbf{z}, \boldsymbol{\pi}) - \boldsymbol{\lambda}^t \cdot \sum_{p,a,r} \boldsymbol{\pi}^{p,a,r}\right)$
6: $\quad \boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t$ # dual variables for the next iteration
7: $\quad$ Set α_t, the step size of the current iteration
8: $\quad$ **for each** $\langle h, m, l\rangle$ **do**
9: $\quad\quad q = \sum_{p,a,r} \widehat{\boldsymbol{\pi}}^{p,a,r}_{h,m,l}$ # num. paths using $\langle h, m, l\rangle$
10: $\quad\quad$ **if** $q > 0$ **and** $\widehat{\mathbf{y}}_{h,m,l} = 0$ **then**
11: $\quad\quad\quad \boldsymbol{\lambda}^{t+1}_{h,m,l} = \boldsymbol{\lambda}^{t+1}_{h,m,l} + \alpha_t q$
12: $\quad\quad$ **end if**
13: $\quad$ **end for**
14: $\quad$ **break if** $\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t$ # convergence
15: **end for**
16: **return** $\widehat{\mathbf{y}}, \widehat{\mathbf{z}}$

subgradient steps at iteration t.

The key point of the method is that solutions to Eq. (4.13) and (4.14) can be computed efficiently using separate processes. In particular, Eq. (4.13) corresponds to a standard dependency parsing problem, where for each dependency $\langle h, m, l\rangle$ we have an additional score term $c \cdot \boldsymbol{\lambda}_{h,m,l}$ —in our experiments we use the projective dependency parsing algorithm by Eisner (2000). To calculate Eq. (4.14) we use the assignment method described in chapter 3, where it is straightforward to introduce additional score terms $-\boldsymbol{\lambda}_{h,m,l}$ to every factor $\boldsymbol{\pi}^{p,a,r}_{h,m,l}$. 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. (4.4), 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.

4.4 Related Work

There have been a number of approaches to joint parsing of syntactic and semantic dependencies that have been reviewed in chapter 2. Focusing more specifically in dual-decomposition works, 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. 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. 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.

4.5 Experiments

We present experiments using our syntactic-semantic parser on the CoNLL-2009 Shared Task English benchmark (Hajič et al., 2009). The dataset consists of 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. Further details can be found on appendix C.

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

Base Pipeline. Runs a syntactic parser and then an SRL system constrained to paths of the previously computed best syntactic tree. Regarding semantic constraints, we only enforce **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 the **cRole** and **cArg**constraints, but restricted 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.

4.5.1 Implementation

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

Syntactic model In a similar setting than the approach introduced in chapter 3, we used two discriminative arc-factored models for labeled dependency parsing: a first-order model, and a second-order model with grandchildren interactions, both re-implementations of the parsers by McDonald et al. (2005a) and Carreras (2007) respectively. We compute probabilistic scores based on Koo et al. (2007). Marginal probabilities are obtained by running inside-outside algorithms. As a result we get one score per dependency.

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* configuration, 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 observe 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.[2] All methods we test are restricted to this syntactic scope captured by the Xue and Palmer (2004) rules. To generate a list of paths, we proceed 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 we 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 covering 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. [3]

SRL model Our SRL features are based in the work of Johansson (2009). Johansson lists and categorizes his set of features. The features are standard for SRL but a distinction is made between *primary*, *secondary* and *interdependency* features. This distinction is relevant to the fact that some of these require a previous syntactic parse. These categories are defined in the previous chapter

[2]This is specific to CoNLL-2009 data for English. In general, for other languages the coverage of these rules may be lower. See chapters 3 and 5 for a discussion.

[3]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.

3 on section 3.4.1. To train SRL models we used the averaged Perceptron (Collins, 2002). For the base pipeline system we trained standard SRL classifiers. For the rest of models we used the structured Perceptron running the assignment algorithm as inference routine. In this latter case, we generate 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 as well as their syntactic path.

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

$$(1-\beta)\ s_{\text{syn}}(\mathbf{x},\mathbf{y}) + \beta\ s_{\text{srl}}(\mathbf{x},\mathbf{z},\boldsymbol{\pi}) \quad .$$

We used normalized marginal probabilities of dependencies for syntactic scores, 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 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 an initial step size of 0.001.

4.5.2 Results

To evaluate syntactic dependencies we use as usual, the *unlabeled attachment score* (UAS), i.e., the percentage of words with the correct head, and the *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}). Appendix C describes in further detail these metrics. [4]

[4] As a convection in this thesis and as reported in the previous chapter, our evaluation metrics slightly differ 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. Here, we are not addressing predicate

	order	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 4.1: Results on development for the baseline and assignment pipelines, running first and second-order syntactic parsers, and the Forest method.

Table 4.1 shows the results on the development set for our three first methods. We 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. [5] The *Forest* method also shows an improvement in recall of semantic roles with respect to the pipeline configuration. 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 described in chapter 3. 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, evaluated in the sem_{pp} column.

The *Forest* configuration is our starting point to run the dual-decomposition algorithm. We ran experiments for various values of the β parameter. Table 4.2 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

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 for the English dataset is a reasonably simple task with a very high baseline around 90%.

[5]The results presented in this chapter for the English datasets are not directly comparable to the assignment results from chapter 3. Here we evaluate both nominal and verbal predicates. In contrast, the previous chapter only showed results for verbal predicates. In addition, here we present experiments from the 2012 version of our parsers that were latter improved to run the assignment experiments of chapter 3. Across different chapters and configurations, we consistently observe improvements regarding the enforcement of the assignment constraints for the case of the English datasets.

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 4.2: 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.

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 2nd order predictions of paths are quite accurate. We also observe slight benefits in syntactic accuracy.

Table 4.3 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 by Gesmundo et al. (2009), which proved to be very competitive ranking third in the closed challenge. We also include the Lluís et al. (2009) system, which is another joint syntactic-semantic system from CoNLL–2009.[6] In the WSJ test *DD* obtains the best syntactic accuracies, while the *Pipeline* obtains the best semantic F_1. The bottom part of table 4.3 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.

[6]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 are slightly different. However, note that Gesmundo et al. (2009) 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.

WSJ	LAS	UAS	sem_p	sem_r	sem_{F_1}	sem_pp
Lluís et al. (2009)	87.48	89.91	73.87	67.40	70.49	39.68
Gesmundo et al. (2009)	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
Lluís et al. (2009)	80.92	85.96	62.29	59.22	60.71	29.79
Gesmundo et al. (2009)	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 4.3: 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).

Table 4.4 shows statistical significance tests for the syntactic LAS and semantic F_1 scores from table 4.3. We have applied the sign test (Wackerly et al., 2007) and the approximate randomization test (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 smaller size of that dataset.

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 $\sim$8% due to the overhead from the assignment algorithm. The *Forest* method, with an average of 61.3 paths per predicate, is $\sim$13% slower than the pipeline due to the exploration of the space of precomputed paths. Finally, *Dual Decomposition* with second 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

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

Table 4.4: Statistical tests of significance for LAS and sem_{F_1} differences between pairs of systems from table 4.3. ○/● = 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: *Lluís et al. (2009)*, ME: *Gesmundo et al. (2009)*, PA1/2: *Pipeline with Assignment, 1st/2nd order*, DD1/2: *Dual Decomposition, 1st/2nd order*.

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.

4.6 Remarks

We have introduced efficient methods to parse syntactic dependency structures augmented with predicate-argument relations. A key idea is to predict the local syntactic structure that links a predicate with its arguments, and seek agreement with the full syntactic tree using dual-decomposition techniques.

Regarding the dual-decomposition technique for joint parsing, it does improve over the pipeline systems 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 for 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 become 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 larger improvements can be achieved by integrating additional syntactic-semantic features in the joint process. To study this question, it is necessary to have efficient parsing algorithms for joint dependency structures. Here we contribute with an efficient method that has optimality 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. In the next chapter we will explore methods that model the syntactic structure linking predicates with arguments using shortest-path methods. Whenever these structures can be efficiently parsed, our dual-decomposition algorithm can be employed to define an efficient joint system.

5

Shortest-path Syntactic-semantic Parsing

In this chapter we introduce a Semantic Role Labeling parser that finds semantic roles for a predicate together with the syntactic paths linking the predicates with their arguments. Our main contribution is to formulate SRL in terms of a shortest-path inference, under the assumption that the SRL model is arc-factored. Overall and compared to chapters 3 and 4, our method for SRL is a novel way to exploit larger variability in the syntactic realizations of predicate-argument relations, moving away from pipeline architectures and from a set of precomputed paths. We show experiments pointing that our approach improves the robustness of the predictions, producing arc-factored models that perform closely to methods using unrestricted features from the syntax. The contents of this chapter are extracted from Lluís et al. (2014).

5.1 Motivation

In this chapter we take a different approach to syntactic-semantic parsing. In our scenario SRL is the end goal, and we assume that syntactic parsing is only an intermediate step as in Naradowsky et al. (2012). In contrast, in our setting, we explicitly generate the syntactic paths from which we extract features to support the SRL predictions. We define a model that, given a predicate, iden-

tifies each of the semantic roles together with the syntactic path that links the predicate with their arguments.

As in previous chapters and following the work of Moschitti (2004) and Johansson (2009), we take the syntactic path as the main source of syntactic features, but instead of just conditioning on it, we predict it together with the semantic role.

Table 5.1 shows the most frequent path patterns on CoNLL-2009 data for several languages, where a path pattern is a sequence of ascending arcs from the predicate to some ancestor, followed by some descending arcs to the argument. For English, the distribution of path patterns is rather simple and was captured as pruning heuristics by Xue and Palmer (2004). We observe in the table that the German, Czech and Chinese datasets cover over the 90% of all arguments with the first three most frequent pattern types. In contrast, Japanese exhibits much more variability and a long tail of infrequent types of patterns.

We believe that it is not feasible to capture these patterns manually, and it is not desirable that a statistical system depends on rather sparse non-factored path features. For this reason we think it's worth exploring arc-factored models.

The method presented on this chapter might be specially useful in applications were we are interested in some target semantic role, i.e., retrieving agent relations for some verb, since it processes semantic roles independently of each other. Our method might also be generalizable to other kinds of semantic relations which strongly depend on syntactic patterns such as relation extraction in information extraction or discourse parsing.

Furthermore, this method could be combined with our dual-decomposition strategy introduced in chapter 4 to allow for a more general approach to syntactic-semantic parsing considering more syntactic diversity and arbitrary predicate-argument paths.

Note that in the previous chapters we were limited to a single-best or a set of precomputed paths from predicates to arguments. Applying the Xue and Palmer (2004) rule we were able to discard all paths for the English dataset that were

English			German			Czech			Chinese			Japanese		
$\sum$%	%	path	$\sum$%	%	path	$\sum$%	%	path	$\sum$%	%	path	$\sum$%	%	path
63.63	63.6298	$\downarrow$	77.22	77.2202	$\downarrow$	63.90	63.8956	$\downarrow$	78.09	78.0949	$\downarrow$	37.20	37.1977	$\downarrow\downarrow$
73.97	10.3429	$\uparrow\downarrow$	93.51	16.2854	$\uparrow\downarrow$	86.26	22.3613	$\downarrow\downarrow$	85.36	7.26962	$\uparrow\downarrow$	51.52	14.3230	$\downarrow$
80.63	6.65915	$\circ$	97.43	3.92111	$\uparrow\uparrow\downarrow$	90.24	3.98078	$\uparrow\downarrow$	91.27	5.90333	$\uparrow\uparrow\downarrow$	60.79	9.27270	$\downarrow\downarrow\downarrow$
85.97	5.33352	$\uparrow$	98.19	0.76147	$\downarrow\downarrow$	93.95	3.71713	$\downarrow\downarrow\downarrow$	95.93	4.66039	$\uparrow\uparrow$	70.03	9.23857	$\uparrow$
90.78	4.81104	$\uparrow\uparrow\downarrow$	98.70	0.51640	$\uparrow\uparrow\uparrow\downarrow$	95.48	1.52168	$\uparrow\downarrow\downarrow$	97.53	1.60392	$\uparrow$	74.17	4.13359	$\downarrow\downarrow\downarrow\downarrow$
93.10	2.31928	$\uparrow\uparrow\uparrow\downarrow$	99.17	0.46096	$\uparrow$	96.92	1.44091	$\uparrow$	98.28	0.75086	$\uparrow\uparrow\uparrow\downarrow$	76.76	2.59117	$\uparrow\uparrow$
95.19	2.09043	$\uparrow\uparrow$	99.43	0.26841	$\uparrow\downarrow\downarrow$	97.68	0.76714	$\uparrow\uparrow\downarrow$	98.77	0.48734	$\downarrow\downarrow$	78.82	2.06111	$\uparrow\uparrow\downarrow\downarrow$
96.26	1.07468	$\uparrow\uparrow\uparrow\uparrow\downarrow$	99.56	0.12837	$\uparrow\uparrow\downarrow\downarrow$	98.28	0.59684	$\downarrow\downarrow\downarrow\downarrow$	99.13	0.36270	$\uparrow\uparrow\uparrow$	80.85	2.03381	$\downarrow\downarrow\downarrow\downarrow\downarrow$
97.19	0.92482	$\downarrow\downarrow$	99.67	0.10503	$\uparrow\uparrow\uparrow\uparrow\downarrow$	98.60	0.31759	$\uparrow\downarrow\downarrow\downarrow$	99.45	0.31699	$\uparrow\uparrow\uparrow\uparrow\downarrow$	82.66	1.80631	$\uparrow\downarrow\downarrow$
97.93	0.74041	$\uparrow\uparrow\uparrow$	99.77	0.10503	$\uparrow\uparrow$	98.88	0.28227	$\uparrow\uparrow\downarrow\downarrow$	99.72	0.27041	$\uparrow\uparrow\uparrow\uparrow$	83.71	1.05558	$\uparrow\uparrow\uparrow$
98.41	0.48565	$\uparrow\uparrow\uparrow\uparrow\uparrow\downarrow$	99.82	0.04960	$\downarrow\downarrow\downarrow$	99.15	0.26721	$\uparrow\uparrow\uparrow\downarrow$	99.82	0.10049	$\downarrow\downarrow\downarrow$	84.74	1.02828	$\uparrow\uparrow\uparrow\downarrow\downarrow$
98.71	0.29769	$\uparrow\uparrow\uparrow\uparrow$	99.87	0.04960	$\uparrow\uparrow\uparrow$	99.27	0.12430	$\downarrow\downarrow\downarrow\downarrow\downarrow$	99.86	0.03623	$\uparrow\downarrow\downarrow$	85.68	0.93500	$\uparrow\uparrow\downarrow\downarrow\downarrow$
98.94	0.22733	$\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\downarrow$	99.90	0.02626	$\circ$	99.37	0.10103	$\uparrow\uparrow\uparrow\uparrow\downarrow$	99.89	0.02890	$\uparrow\uparrow\downarrow\downarrow$	86.61	0.93273	$\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow$
99.11	0.17805	$\uparrow\downarrow\downarrow$	99.92	0.02042	$\uparrow\uparrow\uparrow\downarrow\downarrow$	99.47	0.09747	$\uparrow\uparrow$	99.92	0.02890	$\uparrow\uparrow\uparrow\uparrow\uparrow\downarrow$	87.29	0.68249	$\uparrow\uparrow\uparrow\uparrow\downarrow\downarrow$
99.27	0.15316	$\downarrow\downarrow\downarrow$	99.94	0.02042	$\uparrow\uparrow\uparrow\uparrow\uparrow\downarrow$	99.56	0.08515	$\uparrow\uparrow\downarrow\downarrow\downarrow$	99.94	0.02846	$\circ$	87.90	0.60969	$\uparrow\downarrow\downarrow\downarrow$
99.39	0.12065	$\uparrow\uparrow\uparrow\uparrow\uparrow$	99.95	0.01459	$\uparrow\uparrow\downarrow\downarrow\downarrow$	99.63	0.07419	$\uparrow\uparrow\uparrow\downarrow\downarrow$	99.96	0.02070	$\uparrow\uparrow\uparrow\uparrow\uparrow$	88.47	0.56646	$\uparrow\uparrow\downarrow\downarrow\downarrow\downarrow$
99.50	0.11024	$\uparrow\uparrow\downarrow\downarrow$	99.96	0.01167	$\downarrow\downarrow\downarrow\downarrow$	99.69	0.05667	$\uparrow\downarrow\downarrow\downarrow\downarrow$	99.97	0.00992	$\uparrow\uparrow\downarrow\downarrow\downarrow$	89.01	0.53689	$\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow$
99.60	0.09931	$\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\downarrow$	99.97	0.00875	$\uparrow\downarrow\downarrow\downarrow$	99.73	0.04216	$\uparrow\uparrow\uparrow\uparrow\uparrow\downarrow$	99.98	0.00733	$\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\downarrow$	89.49	0.48684	$\uparrow\uparrow\uparrow\downarrow\downarrow\downarrow$
99.65	0.05283	$\uparrow\downarrow\downarrow\downarrow$	99.98	0.00875	$\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\downarrow$	99.76	0.02875	$\uparrow\uparrow\uparrow\downarrow\downarrow\downarrow$	99.99	0.00431	$\uparrow\uparrow\uparrow\uparrow\downarrow\downarrow$	89.94	0.45044	$\uparrow\uparrow\uparrow\uparrow$

Table 5.1: Summary of the most frequent paths on the CoNLL-2009 Shared Task datasets. $\uparrow$ indicates that we traverse a syntactic dependency upwards from a modifier to a head. $\downarrow$ is for dependencies following a descending head to modifier edge. The symbol $\circ$ represents that the argument is the predicate itself. We exclude from this table Catalan and Spanish as predicates and arguments are always trivially related by a single syntactic dependency that descends.

not matching the pattern of a sequence of ancestors plus a single descendant. Table 5.1 shows that most of the English path patterns end in fact with zero or one descending arcs.

That enumeration strategy offered at a reasonable cost enough syntactic diversity for some particular datasets. But for example, Czech and Japanese show a larger portion of paths with two or more descending arcs. As predicate-arguments relations become more complex the pruning strategy becomes infeasible as enumerating all paths takes an impractical amount of time and space.

In the next section 5.2 we introduce an arc-factorization of SRL analogous to first-order dependency parsing. Section 5.3 frames the factorized semantic role labeling problem as a shortest-path problem. We review some alternatives to train the classifiers in section 5.4. Finally, we present and discuss our experiments in section 5.5 and we give our final remarks in section 5.6.

5.2 Arc-factored SRL

We define an SRL parsing model that retrieves predicate-argument relations based on arc-factored syntactic representations of paths connecting predicates with their arguments. We assume a fixed sentence $\mathbf{x} = x_1, \ldots, x_n$ and a fixed predicate index p. An indicator vector $\mathbf{z}$, where $\mathbf{z}_{r,a} = 1$ flags that token a is filling role r for predicate p. Our SRL parser performs the following optimization:

$$\underset{\mathbf{z} \in \mathcal{Z}(\mathbf{x},p)}{\operatorname{argmax}}\, s(\mathbf{x}, p, \mathbf{z}) \quad , \tag{5.1}$$

where $\mathcal{Z}(\mathbf{x}, p)$ defines the set of valid argument structures for p, and $s(\mathbf{x}, p, \mathbf{z})$ computes a plausibility score for $\mathbf{z}$. Our first assumption is that the score function factors over role-argument pairs:

$$s(\mathbf{x}, p, \mathbf{z}) = \sum_{z_{r,a}=1} s(\mathbf{x}, p, r, a) \quad . \tag{5.2}$$

Then we assume two components in this model, one that scores the role-argument pair alone, and another that considers the best ($\max$) syntactic dependency path $\boldsymbol{\pi}$ that connects the predicate p with the argument a:

$$s(\mathbf{x}, p, r, a) = s_0(\mathbf{x}, p, r, a) + \max_{\boldsymbol{\pi}} s_{\text{syn}}(\mathbf{x}, p, r, a, \boldsymbol{\pi}) \quad . \tag{5.3}$$

The model does not assume access to the syntactic structure of $\mathbf{x}$, hence in Eq. (5.3) we locally retrieve the maximum-scoring path for an argument-role pair. Furthermore, we assume that the syntactic component factors over labeled syntactic dependencies of the path:

$$s_{\text{syn}}(\mathbf{x}, p, r, a, \boldsymbol{\pi}) = \sum_{\langle h,m,l \rangle \in \boldsymbol{\pi}} s_{\text{syn}}(\mathbf{x}, p, r, a, \langle h, m, l \rangle) \; . \tag{5.4}$$

This will allow us to employ an efficient shortest-path inference. Note that since paths are locally retrieved per role-argument pair, there is no guarantee that the set of paths across all roles forms a (sub)tree.

As we introduced in the previous chapters we consider a constrained space of valid argument structures in $\mathcal{Z}(\mathbf{x}, p)$: where (a) each role is realized at most once, and (b) each token fills at most one role. This can be efficiently solved as a linear assignment problem as we described in chapter 3 whenever the SRL model factors over role-argument pairs, as in Eq. (5.2).

5.3 SRL as a Shortest-path Problem

We now focus on solving the maximization over syntactic paths in Eq. (5.3). Finding the path that maximizes a score or a longest path without any further assumption is an NP-hard problem (Karp, 1972). Under some simplifications, we will turn it into a minimization problem which can be solved with a polynomial algorithm, in our case a shortest-path method.

Assume a fixed argument and role, and define $\theta_{\langle h,m,l\rangle}$ to be a *non-negative penalty* for the syntactic dependency $\langle h, m, l\rangle$ to appear in the predicate-argument path. We describe a shortest-path method that finds the path of arcs with the smaller penalty:

$$\min_{\boldsymbol{\pi}} \sum_{\langle h,m,l\rangle \in \boldsymbol{\pi}} \theta_{\langle h,m,l\rangle} \quad . \tag{5.5}$$

We find these paths by appropriately constructing a weighted graph $G = (V, E)$ that represents the problem. Later, we show how to adapt the arc-factored model scores to be non-negative penalties, such that the solution to the Eq. (5.5) will be the negative of the maximizer of Eq. (5.3) under the previous factorization.

It remains only to define the graph construction where paths correspond to arc-factored edges weighted by θ penalties. We start by noting that any path from a predicate p to an argument v_i is formed by a number of *ascending* syntactic arcs followed by a number of *descending* arcs. The ascending segment connects p to some ancestor q (q might be p itself, which implies an empty ascending segment); the descending segment connects q with v_i (which again might be empty). To compactly represent all these possible paths we define the

Algorithm 3 Construction of the Graph for Shortest-path Semantic Role Labeling

1. Add node p as the *source* node of the graph.
2. Add nodes $u_1, \ldots, u_n$ for every token of the sentence except p.
3. Link every pair of these nodes u_i, u_j with a directed edge $a_{i \leftarrow j}$ weighted by the corresponding ascending arc, namely $\min_l \theta_{\langle j,i,l \rangle}$. Also add ascending edges from p to any u_i. So far we have a connected component representing all ascending paths.
4. Add nodes $v_1, \ldots, v_n$ for every token of the sentence except p, and add edges $d_{i \rightarrow j}$ between them weighted by descending arcs, namely $\min_l \theta_{\langle i,j,l \rangle}$. This adds a second strongly-connected component representing descending segments.
5. For each i, add an edge from u_i to v_i with weight 0. This ensures that ascending and descending path segments are connected consistently.
6. Add direct descending edges from p to all the v_i nodes to allow for only-descending paths, weighted by $\min_l \theta_{\langle p,i,l \rangle}$.

graph in algorithm 3. The algorithm consists in building two sets nodes. First we add all nodes involved in the ascending portion of the path. Then we add all nodes for the descending portion of the path. Ascending nodes are connected to descending nodes but we don't allow to reach an ascending node from a descending node. See figure 5.1 for a representation of this graph.

Our method builds the graph for each possible role of the predicate. We then find the shortest-paths from p to all arguments v_i that minimize Eq. (5.5). The Dijkstra's algorithm (Dijkstra, 1959) finds the optimal path from predicate p to all tokens in time $O(V^2)$, see Cormen et al. (2009) for an in-depth description.

5.4 Adapting and Training Model Scores

The shortest-path problem is undefined if a negative cycle is found in the graph as we may indefinitely decrease the cost of a path by looping over this cycle. Furthermore, Dijkstra's method requires all arc scores to be non-negative penalties. However, the model in Eq. (5.4) computes plausibility scores for dependencies, not penalties. And, if we set this model to be a standard feature-based linear predictor, it will predict unrestricted real-valued scores.

One approach to map plausibility scores to penalties is to assume a log-linear

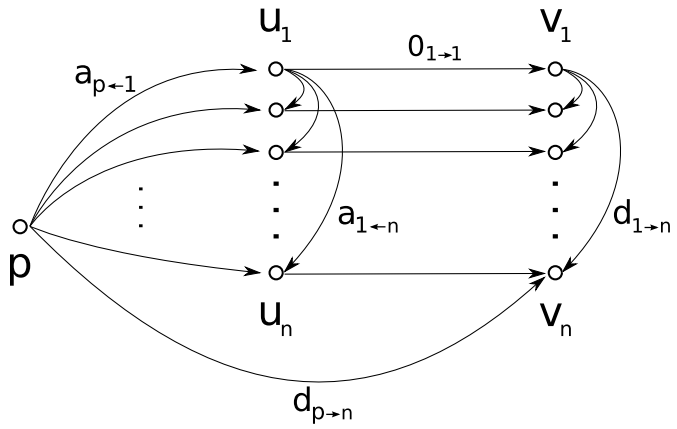

Figure 5.1: Graph representing all possible syntactic paths from a single predicate to their arguments. Algorithm 3 builds this graph. We find in this graph the best SRL using a shortest-path algorithm. Note that many edges are omitted for clarity reasons. The nodes and arcs are labeled as follows: p is the predicate and source vertex; $u_1, \ldots, u_n$ are tokens reachable by an ascending path; $v_1, \ldots, v_n$ are tokens reachable by a ascending path (possibly empty) followed by a descending path (possibly empty); $a_{j \leftarrow i}$ is an *ascending* dependency from node u_i to node u_j; $d_{j \rightarrow i}$ is a *descending* dependency from node v_i to node v_j; $0_{i \rightarrow i}$ is a 0-weighted arc that connects the ascending portion of the path ending at u_i with the descending portion of the path starting at v_i.

form for our model. Let us denote by $\bar{x}$ the tuple $\langle \mathbf{x}, p, r, a \rangle$, which we assume fixed in this section. The log-linear model predicts:

$$\Pr(\langle h, m, l \rangle \mid \bar{x}) = \frac{\exp\{\mathbf{w} \cdot \mathbf{f}(\bar{x}, \langle h, m, l \rangle)\}}{Z(\bar{x})} \quad , \tag{5.6}$$

where $\mathbf{f}(\bar{x}, \langle h, m, l \rangle)$ is a feature vector for an arc in the path, $\mathbf{w}$ are the parameters, and $Z(\bar{x})$ is the normalizer. We can turn predictions into non-negative penalties by setting $\theta_{\langle h,m,l \rangle}$ to be the negative log-probability of $\langle h, m, l \rangle$; namely $\theta_{\langle h,m,l \rangle} = -\mathbf{w} \cdot \mathbf{f}(\bar{x}, \langle h, m, l \rangle) + \log Z(\bar{x})$. Note that $\log Z(\bar{x})$ shifts all values to the non-negative side. The score of a path will be computed as a sum of negative log-probabilities, i.e., as the product of probabilities.

However, log-linear estimation of $\mathbf{w}$ is typically expensive since it requires to repeatedly compute feature expectations. Furthermore, our model as defined in Eq. (5.3) combines arc-factored path scores with path-independent scores, and it is desirable to train these two components jointly. We opt for a mistake-driven training strategy based on the Structured Averaged Perceptron (Collins, 2002), which directly employs shortest-path inference as part of the training

process.

To do so we predict plausibility scores for a dependency directly as $\mathbf{w} \cdot \mathbf{f}(\bar{x}, \langle h, m, l \rangle)$. To map scores to penalties, we define

$$\theta_0 = \max_{\langle h,m,l \rangle} \mathbf{w} \cdot \mathbf{f}(\bar{x}, \langle h, m, l \rangle) \tag{5.7}$$

and we set

$$\theta_{\langle h,m,l \rangle} = -\mathbf{w} \cdot \mathbf{f}(\bar{x}, \langle h, m, l \rangle) + \theta_0 \quad . \tag{5.8}$$

Thus, θ_0 has a similar purpose as the log-normalizer $Z(\bar{x})$ in a log-linear model, i.e., it shifts the negated scores to the positive side; but in our version the normalizer is based on the max value, not the sum of exponentiated predictions as in log-linear models. We set our model function to be

$$s_{\text{syn}}(\bar{x}, \langle h, m, l \rangle) = -\mathbf{w} \cdot \mathbf{f}(\bar{x}, \langle h, m, l \rangle) + \theta_0 \quad . \tag{5.9}$$

for which the shortest-path method is exact. Note however, that shifting by the max unfairly penalizes longer paths. If we shift every dependency weight by a value θ_0, a path of length l will be penalized by $\theta_0 \cdot l$. Despite this drawback and to prove the overall architecture, for practical reasons we perform experiments in the next section using this training strategy.

5.5 Experiments

As in previous chapters, we present experiments using the CoNLL-2009 Shared Task datasets (Hajič et al., 2009). We only run experiments here for the verbal predicates of the English datasets. Evaluation, as usual, is based on precision, recall and F_1 over correct predicate-argument relations[1]. Our system uses the feature set of the state-of-the-art system by Johansson (2009). These features are described in the previous chapter 3 on section 3.4.1 but here we ignore the features that do not factor over single arcs of the path.

[1]Unlike in the official CoNLL-2009 evaluation and following the convention of this thesis we exclude the predicate sense from the features and the evaluation. All results shown here including the Zhao et al. (2009a) results are computed ignoring predicate sense disambiguation.

We jointly find semantic roles and paths linking predicates and their arguments. But rather than searching over all possible dependencies to build the paths we only consider the most likely ones. To do so, we employ a probabilistic dependency-based model, following Koo et al. (2007), that computes the distribution over head-label pairs for a given modifier, i.e., $\Pr(h, l \mid \mathbf{x}, m)$. Specifically, for each modifier token we only consider the dependencies or heads whose probability is above a factor γ of the most likely dependency for this given modifier. Thus, $\gamma = 1$ selects only the most likely dependency similar to a pipeline system, but without enforcing tree constraints, and as γ decreases more dependencies are considered, to the point where $\gamma = 0$ would select all possible dependencies. Table 5.2 shows the ratio of dependencies included with respect to a pipeline system for the development set. As an example, if we set $\gamma = 0.5$ for a given modifier we consider the most likely dependency and also the dependencies with a probability larger than 1/2 of the probability of the most likely one. In this case the total number of dependencies is 10.3% larger than only considering the most likely one.

Table 5.3 shows results of the method on development data, when training and testing with different γ values. The general trend is that testing with the most restricted syntactic graph results in the best performance. However, we observe that as we allow for more syntactic variability during training, the results largely improve. Setting $\gamma = 1$ for both training and testing gives a semantic F_1 of 75.9. This configuration is similar to a pipeline approach but considering only factored features. If we allow to train with $\gamma = 0.1$ and we test with $\gamma = 1$ the results improve by 1.96 points to a semantic F_1 of 77.8 points. When syntactic variability is too large, e.g., $\gamma = 0.01$, no improvements are observed. The results point that some amount of variability during training could potentially make classifiers more robust.

Finally, table 5.4 shows results on the verbal English WSJ test set using our best configuration. We compare to the state-of-the art system by Zhao et al. (2009a) that was the top-performing system for the English language in SRL

Threshold γ	1	0.9	0.5	0.1	0.01
Ratio	1	1.014	1.103	1.500	2.843

Table 5.2: Ratio of additional dependencies in the graphs with respect to a single-tree pipeline model ($\gamma = 1$) on development data.

at the CoNLL-2009 Shared Task. The Gesmundo et al. (2009) system was the best joint system of that task.

We also show the results for a system trained and tested with $\gamma = 1$ thus similar to a pipeline system but using our shortest-path approach and therefore restricted to factored features. In addition, we include an equivalent pipeline system using all features, both factored and non-factored, as defined in Johansson (2009). The results of our factored model are behind the non-factored approach. We observe that by not being able to capture these non-factored features the final performance drops by 1.6 F_1 points, a result that confirms previous work such as Xue and Palmer (2004).

Our baseline non-factored system is behind the state of the art by about 2.9 points. Note however that this system represents a fairly simple approach using only features extracted from a path. We do not apply any post-processing, except our assignment inference method from chapter 3. Our approximate training strategy based on shift-normalizations could also contribute this decrease in F_1 results. This hypothesis is left to be tested as future work.

5.6 Remarks

We have formulated SRL in terms of shortest-path inference. Our model predicts semantic roles together with associated syntactic paths. A key idea is to assume an arc-factored representation of the path. This property allows for efficient shortest-path algorithms that, given a predicate and a role, retrieve the most likely arguments and their paths.

In the experimental section we observe that arc-factored models are in fact more restricted, with a drop in accuracy with respect to unrestricted models.

test threshold	prec (%)	rec (%)	F_1
training γ = 1			
1	77.91	73.97	75.89
0.9	77.23	74.17	75.67
0.5	73.30	**75.03**	74.16
0.1	58.22	68.75	63.05
0.01	32.83	53.69	40.74
training γ = 0.5			
1	81.17	73.57	77.18
0.9	80.74	73.78	77.10
0.5	78.40	74.79	76.55
0.1	65.76	71.61	68.56
0.01	42.95	57.68	49.24
training γ = 0.1			
1	**84.03**	72.52	**77.85**
0.9	83.76	72.66	77.82
0.5	82.75	73.33	77.75
0.1	77.25	72.20	74.64
0.01	63.90	65.98	64.92
training γ = 0.01			
1	81.62	69.06	74.82
0.9	81.45	69.19	74.82
0.5	80.80	69.80	74.90
0.1	77.92	68.94	73.16
0.01	74.12	65.92	69.78

Table 5.3: Results of our shortest-path system for different number of allowed dependencies showing precision, recall and F_1 on development set for the verbal predicates of the *English* language.

However, we also observe that our method largely improves the robustness of the arc-factored method when training with a degree of syntactic variability. Overall, ours is a simple strategy to bring arc-factored models close to the performance of unrestricted models.

The purpose of the experimental section is mainly to show the feasibility of the approach. Even thought the results of the shortest-path system are behind the state of the art, we think that are promising considering that we are evaluating a simpler factored model.

We performed experiments using averaged Perceptron classifiers. Our sim-

system	prec(%)	rec(%)	F_1
Zhao et al. (2009a)	86.91	81.22	83.97
Gesmundo et al. (2009)	83.45	78.83	81.07
Non-factored	86.96	75.92	81.06
Factored $\gamma = 1$	79.88	76.12	77.96
Factored best	85.26	74.41	79.46

Table 5.4: Test set results for verbal predicates of the in-domain *English* dataset. The configurations are labeled as follows. *Factored* $\gamma = 1$: our shortest-path system trained and tested with $\gamma = 1$, similar to a pipeline system but without enforcing tree constraints and restricted to arc-factored features. *Factored best*: our shortest-path system with the best results from table 3. *Non-factored*: an equivalent pipeline system that includes both factored and non-factored features.

ple transform of these scores unfairly penalizes longer paths. We left as future work to implement other probabilistic classifiers.

The approach introduced in this chapter may be combined with the dual-decomposition strategy of chapter 4 to allow for processing languages with arbitrary predicate-argument paths that are not feasible to capture by hand-written rules. The shortest-path approach will complement and extend to multiple languages our joint dual-decomposition previous proposal.

Future work should explore further approaches to parse partial syntactic structure specific to some target semantic relations such as in information extraction. Our strategy could be useful in the cases where we are only interested in semantic role labels and we view syntactic parsing only as an intermediate step.

6

Conclusions and Future Work

We conclude this thesis by summarizing the main points discussed along the previous chapters. We also give the possible future research directions. We start by reviewing the tree main contributions presented in chapters 3, 4 and 5:

- Our first contribution is to frame the SRL problem as a linear assignment task. Under this framework we can efficiently control uniqueness constraints over the arguments of a predicate.
- The second contribution is to define a joint model largely based on standard components that finds the optimal joint parse by using dual-decomposition techniques.
- Finally, we introduce a shortest-path framework to jointly generate predicate-argument relations and the syntactic paths linking the predicates with their arguments.

Our first two contributions are also described in Lluís et al. (2013) and our third contribution was initially presented in Lluís et al. (2014).

SRL as assignment In chapter 3 we set up a bipartite graph to compute an assignment of roles to arguments. In our approach we enforce unique roles per predicate and at most a single role per argument. Previous work has solved SRL applying a larger number of domain constraints (Punyakanok et al., 2004).

We presented experimental data evaluating our assignment method in multiple languages. Our experiments showed for the English and German datasets that enforcing uniqueness improves the final performance of the classifiers. In contrast, the Czech and Spanish datasets showed better results when these constraints are not enforced. These last two datasets contained a slightly larger number of repeated annotated roles per predicate. However, the experiments point out that the percentage of repeated roles may not be the most relevant factor to identify the datasets which will show improvements when the assignment approach is applied.

Enforcing uniqueness on semantic roles reduces the search space and potentially makes learning curves steeper for some classifiers. However, as a consequence the most accurate classifiers will probably be penalized when searching on this constrained space. Our hypothesis is that enforcing uniqueness could improve the final results specially in the cases were we work with smaller datasets.

Regarding the efficiency, our proposal can be solved by an exact $O(n^3)$ algorithm. Our approach is less expressive than Punyakanok et al. (2004) but it covers some of the most relevant SRL constraints identified by Surdeanu et al. (2007). In addition, the assignment algorithm is simple and straightforward to implement avoiding the need of external ILP solvers.

Our assignment proposal has two main strong points. First, it can be applied as a post-processing step in a pipeline system. Also it offers the possibility of finding the optimal solution within a large set of predicate-argument combinations extracted from a set of different syntactic parse trees.

A second relevant point is that this framework can be easily extended to allow for a finite number of repeated roles per predicate. Thus the approach can be adapted for a particular dataset with more than one argument filling the same role for a given predicate.

A drawback of our framework is that it does not control for solutions that contain inconsistent syntax. By choosing the optimal combination we may

select the best candidates generated from different syntactic trees. This problem is addressed in our joint models.

Dual-decomposition joint parsing We defined a model that finds the optimal joint parse by using dual-decomposition optimization techniques. We enforce agreement constraints between the syntactic and semantic layers to find the global solution. Our joint model incorporates the assignment framework previously described. By forcing a shared syntax we avoid the potential inconsistent syntax problems.

We set up equivalent pipeline and joint models based on components using standard features and linear classifiers for learning. Our main experimental goal was to compare the joint approach with the pipeline systems. We observed improvements in our joint model in the cases where a first-order syntactic parser was providing the syntax. We also compared our system with the state-of-the-art joint approach of Gesmundo et al. (2009). Results for the English language proved that our proposal is competitive.

Our method has two main advantages. First it generates the best joint solution to our optimization problem whenever it converges. In experiments we observe convergence in over 99% of the cases. Converge in that cases is reached in an average of $\sim$37 iterations.

A second advantage of our proposal is, as previously introduced, that it relies on standard components and features. However, as a consequence we may still suffer from limitations common to other pipeline systems, e.g., a drop in performance when testing on out-of-domain data.

Given the experimental data, we believe that our approach may be well-suited in the cases were syntactic parsers show mild results and larger margins for improvements. Experiments point that as individual components improve the benefit of a joint optimization is less clear.

Shortest-path semantic role labeling This last contribution has as its key point a factorization of the SRL scores over dependencies in an analogous way

that syntactic parsers factorize scores over syntactic dependencies. As a result we were able to use polynomial algorithms to find the optimal predicate-argument paths.

We have implemented and run experiments for the English verbal datasets. The experiments proved the feasibility of the approach that was one of our main goals. We also observed results pointing that potentially more robust classifiers are learned under this framework.

We presented experiments comparing our shortest-path model with respect to an equivalent baseline. In our shortest-path approach we discarded all features that were non-factorizable over path arcs. Thus we observed accordingly slightly better results for our baseline system that exploits both factorized and non-factorized features.

The introduced shortest-path approach represents a novel framework for semantic parsing. SRL is usually considered a classification task with some exceptions such as BIO tagging, CRF over trees or more recently, extended parsing algorithms (Màrquez et al., 2005; Cohn and Blunsom, 2005; Oepen et al., 2014). Our proposal may be appropriate in an scenario were we are interested in SRL as the end goal, assuming that syntactic parsing is only an intermediate step.

As noted before, our model may generate inconsistent syntactic paths for the arguments of a given predicate. This issue can be addressed by combining this approach with the dual-decomposition method of chapter 4.

In addition, this combination will offer the possibility to overcome the previous limitation of our dual-decomposition method to a set of precomputed paths generated following a set of rules. Datasets such as the Japanese and Czech present syntactic paths that are hard to capture by a simple set of rules. The shortest-path approach will allow to semantically parse arbitrary predicate-argument relations under our joint approach.

Final remarks This thesis started with two motivating questions. Our first question was whether a joint system will outperform the pipeline approach. Our particular proposal described in chapter 4 showed promising results when syntax was computed by first-order parsers. Unfortunately, we have not seen consistent improvements as we replaced first-order parsers with better second-order analyzers.

To at least partially answer this question for a particular proposal it is important to set up controlled experiments. These experiments should compare equivalent pipeline and joint models. We implemented pipeline and joint systems sharing features, learning and parsing algorithms.

We have shown results for the English datasets. It remains to evaluate our system for multiple languages. The combination of the shortest-path method of chapter 5 with the dual-decomposition joint model from chapter 4 will allow us to process these datasets. The engineering effort to put all these components together is left as future work.

The second initial motivating question regards the efficiency of joint systems. If we look at the case of syntactic dependency parsing, a factorization of the search space allows to efficiently search over this large space. However, no analogous factorization is defined for joint syntactic-semantic parsing. A main difficulty in formalizing such factorization is the divergence between the syntactic and semantic layers.

We opted for a strategy that independently computes syntactic trees and semantic roles. Consistency between both layers is enforced by forcing a shared syntax. We solve the optimization problem by using dual-decomposition techniques. It is implemented as a process that iteratively calls standard components and updates typically few agreement variables. Our model thus feasibly computes the optimal joint parse whenever it converges that is in over the 99% of the sentences of our datasets.

Future Work

There are a number of research directions started in thesis that can be further explored. The combination of the shortest-path model with our assignment and dual-decomposition proposal will finally offer a flexible and efficient framework to jointly parse syntax and semantics for multiple languages. This generic parsing method will allow us to present a widely applicable joint model.

Another extension of our dual-decomposition model would be incorporate other related tasks in our optimization function, e.g., word sense disambiguation. In addition, richer sets of features could be incorporated to our classifiers. E.g., distributional representations such as in Zapirain et al. (2013) extracted from predicates and arguments. These features could potentially improve the performance of syntactic and semantic parsing.

Experimentally, it remains to be tested the hypothesis that our dual-decomposition system may improve the final results specially in the cases were there are larger margins for gains. We could run experiments with datasets of different sizes that will complement a multilingual evaluation of our joint system.

Our shortest-path method defined in chapter 3 is based on probability distributions over edges. However, for simplicity we trained unbounded linear classifiers. A first simple strategy to implement would be to use a softmax transformation on these trained classifiers. Our current projection strategy unfairly penalizes longer paths. It will be interesting to complement the experimental section implementing a probabilistic approach. In addition, our proposal could be evaluated in the context of other semantic analysis tasks such as information extraction or the broad-coverage semantic parsing SemEval 2014 task 8 (Oepen et al., 2014).

These are some of the many research directions and hypothesis to be confirmed regarding joint approaches. Given the potential of joint approaches and the limitations of current pipeline models we believe that it is worth continuing the research in this area.

A

ALGORITHMS

We describe in this appendix the basic algorithms and we summarize the relevant technical details for the main models introduced in chapter 2.

A.1 Chu-Liu-Edmonds

The Chu-Liu-Edmods algorithm (Chu and Liu, 1965; Edmonds, 1967; Tarjan, 1977) solves the Maximum Spanning Tree problem in $O(n^2)$. For clarity reasons we show here the $O(n^3)$ implementation, see algorithm 4.

The sketch of the algorithm is as follows. We select the highest scoring incoming edge for each vertex. If a tree results, it is the MST. Otherwise the graph contains a cycle. We remove every cycle by contracting it to a single node, we then recompute the weight of the incoming edges to this new node, and select the new maximum scoring edges. In contrast to the Eisner family of parsers, this algorithm is not restricted to projective trees.

A.2 Eisner First-order

The Eisner algorithm (Eisner, 1996) was adapted to the dependency parsing framework by McDonald et al. (2005a), see algorithm 5.

Algorithm 4 Chu-Liu-Edmonds

Input: vertices of the graph (sentence tokens) and score(i, j) function
$M \leftarrow$ the highest scoring incoming arcs for each vertex {# G_M is the subgraph induced by M arcs}
while cycles(G_M) **do**
  $K \leftarrow \emptyset$
  for all $c \in$ cycles(G_M) **do**
    $k \leftarrow$ contract(c)
    $K \leftarrow K \cup k$
    for all (i, j) s.t. $j \in$ vertex(C), $i \in$ vertex$(G) \setminus$ vertex(c) **do**
      score$(i, k) \leftarrow$ score$(i, j) -$ (score$(l, j) - \max_j$(score(l, k)), where l is all node in vertex$(G) \setminus$ vertex(c)
    end for
  end for
  for all $k \in K$ **do**
    $l \leftarrow \text{argmax}_{l'}$score$(l', k)$
    $a \leftarrow (l, k) \in S$
    $S \leftarrow S \setminus a$
    $S \leftarrow S \cup (l, k)$
  end for
end while
return G_M

Algorithm 5 Eisner first order

Input: sentence of length n and score(i, j) function
$C[s][t][d][c] \leftarrow 0, \forall s, t, d, c$
for $k = 1, \ldots, n$ **do**
  for $s = 0, \ldots, n - k$ **do**
    $t \leftarrow s + k$
    $C[s][t][\leftarrow][0] = \max_{s \leq r < t} C[s][r][\rightarrow][1] + C[r + 1][t][\leftarrow][1]$+score$(t, s)$
    $C[s][t][\rightarrow][0] = \max_{s \leq r < t} C[s][r][\rightarrow][1] + C[r + 1][t][\leftarrow][1]$+score$(s, t)$
    $C[s][t][\leftarrow][1] = \max_{s \leq r < t} C[s][r][\leftarrow][1] + C[r][t][\leftarrow][0]$
    $C[s][t][\rightarrow][1] = \max_{s < r \leq t} C[s][r][\rightarrow][0] + C[r][t][\rightarrow][1]$
  end for
end for

Table $C[s][t][\leftrightarrow][\{0, 1\}]$ represents the best span from start token s to end token t. The arrows $\leftrightarrow$ indicate which endpoint acts as a partial head. The index $\{0, 1\}$ indicates if the span is open or closed:

open spans can be extended in both ends.

closed spans are finished structures but can be extended in only one of their ends.

A span is a subsequence of tokens where one of the endpoints is its head or *partial* root. This first-order arc-factorization is a main limitation as discussed in chapter 2. In the context of syntactic dependency parsing as dependencies are independently scored, longer constructions such as prepositional phrases are hard to capture. The $O(n^3)$ algorithm is considered *fast* at a practical level.

A.3 Second-order Siblings

McDonald and Pereira (2006) extended the first-order algorithm to capture sibling information. The cost of the algorithm is still $O(n^3)$, i.e., the same asymp-

Algorithm 6 Eisner second order with siblings

Input: sentence of length n and score(h, s, m) function
$C[s][t][d][c] \leftarrow 0, \forall s, t, d, c$
for $k = 1, \ldots, n$ **do**
  for $s = 0, \ldots, n - k$ **do**
    $t \leftarrow s + k$
    # sibling construction
    $C[s][t][\leftrightarrow][2] = \max_{s \leq r < t} C[s][r][\leftarrow][1] + C[r+1][t][\rightarrow][1]$
    # building of a dependency without sibling
    $C[s][t][\leftarrow][0] = C[s][t-1][\rightarrow][1] + C[t][t][\leftarrow][1] + \text{score}(t, -, s)$
    $C[s][t][\rightarrow][0] = C[s][s][\rightarrow][1] + C[s+1][t][\leftarrow][1] + \text{score}(s, -, t)$
    # building of a dependency with sibling
    $C[s][t][\leftarrow][0] = \max_{s \leq r < t} \{C[s][t][\rightarrow][0],$
      $\max_{s \leq r < t} C[s][r][\leftrightarrow][2] + C[r][t][\leftarrow][0] + \text{score}(t, r, s)\}$
    $C[s][t][\rightarrow][0] = \max_{s \leq r < t} \{C[s][r][\rightarrow][0],$
      $\max_{s < r \leq t} C[r][t][\leftarrow][0] + C[r][t][\leftrightarrow][2] + \text{score}(s, r, t)\}$
    # building of complete spans
    $C[s][t][\leftarrow][1] = \max_{s \leq r < t} C[s][r][\leftarrow][1] + C[r][t][\leftarrow][0]$
    $C[s][t][\rightarrow][1] = \max_{s < r \leq t} C[s][r][\rightarrow][0] + C[r][t][\rightarrow][1]$
  end for
end for

totic cost as the first-order algorithm even though at a practical level is noticeably slower, see algorithm 6.

A.4 Second-order Grandchildren

Carreras (2007) further improved the second-order parsing algorithm adding

factors that look at some grandchildren of the head, potentially capturing deeper relations, see algorithm 7.

Algorithm 7 Eisner second order with grandchildren

Input: sentence of length n and score$_{\{hm,ch,cmi,cmo\}}$ function
$C[s][t][d][m] \leftarrow 0, \forall s, t, d, m$
$O[s][t][d][l] \leftarrow 0, \forall s, t, d, l$
for $k = 1, \ldots, n$ **do**
  for $s = 0, \ldots, n - k$ **do**
    $t \leftarrow s + k$
    $\forall l \quad O[s][t][\leftarrow][l] = \max_{r,cmi,ch}$
      $C[s][r][\rightarrow][cmi] + C[r+1][t][\leftarrow][ch]$
      $+$score$(t, s, l)+$score$_{cmi}(t, s, c_{mi}, l)+$
      score$_{ch}(t, s, l, c_h)+$
      $\sum_{p_i} \max_{l_{sem}}$scoresem$(t, s, p_i, l_{sem})/q$

    $\forall l \quad O[s][t][\rightarrow][l] = \max_{r,cmi,ch}$
      $C[s][r][\rightarrow][ch] + C[r+1][t][\leftarrow][cmi]+$
      score$(s, t, l)+$score$_{cmi}(s, t, c_{mi}, l)+$
      score$_{ch}(s, t, l, c_h)+$
      $\sum_{p_i} \max_{l_{sem}}$scoresem$(t, s, p_i, l_{sem})/q$

    $\forall m \quad C[s][t][\leftarrow][m] = \max_{l,cmo}$
      $C[s][m][\leftarrow][cmo] + O[m][t][\leftarrow][l]+$
      score$_{cmo}(s, m, l, c_{mo})$

    $\forall m \quad C[s][t][\rightarrow][m] = \max_{l,cmo}$
      $O[s][m][\rightarrow][l] + C[m][t][\rightarrow][cmo]+$
      score$_{cmo}(m, t, l, c_{mo})$
  end for
end for

The key idea of this algorithm is an efficient indexing of *closed* and *open* structures that are also defined in a chart table C. We describe these structures:

Closed structures are indexed by the start of the span s, the end token t, and in this case also by the syntactic label l.

Open structures are indexed by the start of the span s, the end token t, and in addition a child m that is the closet child to the head of the span whether s or t.

These structures require a space of $O(n^2L + n^3)$. Algorithm 7 shows a combined table of $O(n^4L)$ only for clarity reasons.

A.5 Shift-reduce and Transition-based Parsers

Shift-reduce (Nivre and Nilsson, 2005) are bottom-up parsers that can be defined as parsing a sentence from left to right using a stack of symbols. This definition is not strictly applicable to bottom-up Eisner parsers.

We start by defining an abstract shift-reduce parser. Let $\{\alpha, w\}$ be the configuration of the parser in any given time, where α is a stack of symbols or partially processed items and w is the remaining input sequence.

Two basic processing rules define a bottom up parser:

- shift $\langle \alpha, aw \rangle \rightarrow \langle \alpha a, w \rangle$
- reduce $\langle \beta\alpha, w \rangle \rightarrow \langle \beta A, w \rangle$, for some grammar rule or transformation $A \longrightarrow \alpha$.

Note that the algorithm is indeterministic and searches for any derivation from $\langle \lambda, w \rangle$ to the final configuration $\langle S, \lambda \rangle$. Where λ is the empty string symbol, and S the start symbol of a grammar.

MaltParser Model Shift-reduce parsers were extended and adapted by Nivre (2003) and Covington (2001) to the context of dependency parsing. We describe these extensions in terms of transitions T and configurations C as follows:

- C is the set of configurations c, where a configuration $c = \langle \alpha, w, A \rangle$ is:
  - α is a stack of tokens.
  - w is the remaining part of the word.
  - A is a set of labeled dependency arcs. We note the set as $A = \{i \xrightarrow{l} j\}$. The induced graph by A is G_A with vertices $V = \{0, \ldots, n\}$ and edges in A. It represents the partially constructed tree.

- T is the set of transitions t that are mappings between configurations. The function is not necessary defined for all configurations in C.

We note the stack as $\alpha|i$ representing that the token i is on the top of the stack and α is the remaining part of the stack. String operations noted by $j|w$ represent that j is the first token of the sequence jw. The initial configuration of the parser for an input string x is $C(\alpha, w, A) = \{\lambda, x, \emptyset\}$.

We define the set of transitions between states for the arc-eager Nivre's approach:

- *left arc for label l* $\langle \alpha|i, j|w, A\rangle \rightarrow \left\langle \alpha, j|w, A \cup (j \xrightarrow{k} i, k)\right\rangle$
  if i is not the root node and there is no previous assigned head for token j (one head constraint).
- *right arc for label l* $\langle \alpha|i, j|w, A\rangle \rightarrow \left\langle \alpha|i|j, w, A \cup (i \xrightarrow{k} j, k)\right\rangle$ if there is no previous assigned head for token j.
- *reduce* $\langle \alpha|i, w, A\rangle \rightarrow \langle \alpha, w, A\rangle$
  if i has not an assigned head.
- *shift* $\langle \alpha, i|w, A\rangle \rightarrow \langle \alpha|i, w, A\rangle$

The previous parsing model definition is non-deterministic. Usually, as in Nivre et al. (2007b) we isolate the non-deterministic component of the parser in an *oracle* function, later this function can be approximated by for example a machine learning classifier. Algorithm 8 is a linear time sketch based on the MaltParser procedure with an *oracle* function assumed to be constant-time.

SVM (Joachims, 1999) and MBL (Memory-Based Learning, k-nearest neighbors) (Daelemans and van den Bosch, 2005) are widely chosen machine learning methods for this task, and are implemented in the publicly available *MaltParser*. Even though the algorithm is $O(n)$, it is costly to train at a practical level.

The constraints imposed by the admissible transitions restrict the set of parseable dependency trees to the projective trees.

Algorithm 8 Oracle shift-reduce parser

```
initial configuration c = {λ, x, ∅}
while is_not_terminal(c) do
    if α = λ then
        c ← shift(c)
    else
        t ←oracle(c)
        c ←t(c)
    end if
end while
return graph(c)
```

A.6 Structured Perceptron

Algorithm 9 Structured perceptron

```
Input: examples (x, y) from the training set
w ← 0
for t ← 1 to num_epochs do
    for all (x, y) ∈ training_set do
        ŷ =inference_algorithm(x, w)
        for all factor ∈ y \ ŷ do
            w^(l) ← w^(l) + φ(factor, x, ŷ)
        end for
        for all factor ∈ ŷ \ y do
            w^(l) ← w^(l) − φ(factor, x, ŷ)
        end for
    end for
end for
return w_avg # the average over the training set and epochs
```

The structured perceptron (Collins, 2002) is a simple learning strategy widely applied in NLP. We show an example of this method in algorithm 9. If we assume a dependency parsing setting, the *inference_algorithm* is the parser (e.g., Eisner). This parser generates a dependency tree $\hat{y}$, that can be decomposed in a set of factors $\langle h, m, l \rangle$ for the first-order case. For any misclassified factor, we update the weights of missing (i.e., $f \in y \setminus \hat{y}$) or overpredicted (i.e., $f \in \hat{y} \setminus y$).

A version of this strategy (Shen and Joshi, 2005) shown in algorithm 10 evaluates and ranks a higher number of factors and the update rules are applied

whenever the ranked score of the output inference algorithm does not match the reference ranking.

Algorithm 10 Reranking perceptron

```
Input: examples (x, y) from the training set and GEN function
w ← 0
for t ← 1 to num_epochs do
  for all (x, y) ∈ training_set do
    Ŷ = GEN(x)
    ŷ = argmax_{ŷ′∈Ŷ} score(x, y′, w)
    for all factor ∈ y \ ŷ do
      w^(l) ← w^(l) + φ(factor, x, ŷ)
    end for
    for all factor ∈ ŷ \ y do
      w^(l) ← w^(l) − φ(factor, x, ŷ)
    end for
  end for
end for
```

B

The CoNLL-2006 and 2007 Shared Tasks

The CoNLL-2006 and CoNLL-2007 Shared Tasks (Buchholz et al., 2006; Nivre et al., 2007a) were devoted to syntactic dependency parsing. These tasks boosted research on the field and summarized some of the most relevant approaches.

Table B.1[1] shows the top 5 systems among the 23 presented at the CoNLL-2007 Shared Task closed challenge. The Shared Task evaluated 10 languages but here we only briefly review the top performing systems for the English dataset.

Table B.2[1] summarizes the architecture of these 5 systems. The *prep* column of the table indicates if a system applied a preprocessing step such as the Nivre and Nilsson pseudo-projective algorithm, see Nivre et al. (2007b). The next *model* columns shows that most systems build the labeled syntactic tree in a single step process. An exception is the Nakagawa two-step process. The first step of that system is the unlabeled dependency graph generation and then syntactic label are annotated. The *inference* column shows the different parsing algorithms applied. The *learning* column indicates the machine learning method, among them ISBN (Titov and Henderson, 2007b) that are Bayesian probabilistic graphical models and Gibbs sampling (Geman and Geman, 1984) to estimate probabilities.

[1]Extracted from http://depparse.uvt.nl/depparse-wiki/AllScores.

Team	LAS (%)
Carreras	89.61
Sagae	89.01
Nakagawa	88.41
Titov	88.39
Nilsson	88.11

Table B.1: Best CoNLL-2007 systems for the English language

Team	prep	model	inference	learning	comb
Carreras	no	Eisner-Carreras	Eisner-Carreras	perceptron	—
Sagae	yes	Shift-reduce	LR greedy best-first, left-right	SVM	—
Nakagawa	no	probabilistic model + classification	MST + classification	Gibbs sampling + SVM	—
Titov	yes	shift-reduce	probabilistic, beam of sequences	ISBN	—
Nilsson	yes	shift-reduce	arc-eager	SVM	CLE

Table B.2: Summary of top performing CoNLL-2007 Shared Task systems

Most participants did not report the consumed resourced for training therefore, unfortunately, efficiency cannot be evaluated but we suspect that top performing teams invested large amounts of computational resources.

We observe that shift-reduce and Eisner-based models are the most common approaches. These models offer simplicity, reduced training times and a competitive performance as their main advantages.

Classification tasks, if tractable, can be performed by SVM classifiers. In Eisner-based systems it is more common to apply the Perceptron algorithm, or at a higher cost, MIRA.

The features defined by McDonald et al. (2005b) are widely used with almost no modifications by all teams.

C

DATASETS, TREEBANKS AND MEASURES

Along this dissertation, our main sources of training and testing data are the CoNLL-2008 and 2009 Shared Task datasets (Surdeanu et al., 2008; Hajič et al., 2009). These Shared Tasks made available and popularized in a common format datasets for Catalan and Spanish (Taulé et al., 2008), English (Surdeanu et al., 2008), German (Burchardt et al., 2006), Czech (Hajič et al., 2006), Chinese (Palmer and Xue, 2009) and Japanese (Kawahara et al., 2002).

- **Catalan and Spanish**. The Catalan and Spanish dataset was built by an automatic conversion process to dependencies from a constituent representation. The source of the data is the AnCora corpus consisting mainly in news.

- **English**. The English datasets merge the Penn Treebank 3, the BBN Corpus, the PropBank I and the NomBank. The merging and conversion process is described by Surdeanu et al. (2008).

- **German**. The German dataset is based on the SALSA corpus and only contains verbal predicates. In addition, out-of-domain data is provided from a sample of the Europarl corpus.

- **Czech**. The Czech datasets are from the Prague Dependency Treebank 2.0. Lemmas were automatically added. The analytical and tectogram-

	cat	spa	eng	ger	cze	chi	jap
sentences	13.2	14.3	39.3	36	38.7	22.3	4.4
tokens	390	427	958	649	653	609	113
avg len	29.6	29.8	18.7	18	16.8	27.3	18.0

Table C.1: Summary of CoNLL-2009 datasets

matical layers provide the data for dependencies and semantic roles respectively.

- **Chinese**. The Chinese corpus was generated from the Chinese Treebank 6.0 and the Chinese Proposition Bank 2.0. Also converting constituent structure to a dependency formalism.

- **Japanese**. Japanese datasets consists of sentences from the Kyoto University Text corpus extracted form *Mainichi* Newspapers. For further details see the previous references.

An overview of the datasets in terms of number of sentences, number of tokens and average length of sentences is shown in Table C.1[1]. Before these Shared Tasks some of these datasets were not available following a dependency framework. Generally, automatic procedures were implemented for this conversion process from a constituent to a dependency formalism.

For example, for the English dataset, the main idea of the conversion procedure is to extract the head from each phrase and then set all other tokens as its dependents. Further details can be found in Surdeanu et al. (2008). As a side-effect the number of dependency labels increased. This constitutes a practical burden for many dependency parsing and machine learning algorithms.

Regarding semantic roles there was also a conversion process from the original annotations of PropBank and NomBank on top of the constituents to the new dependency formalism. Semantic predicates were already originally assigned to individual tokens. A procedure aimed to avoid incompatibilities takes the boundaries of semantic arguments and then identifies the head. If more than

[1]Extracted from Hajič et al. (2009).

one head is found the argument is split, e.g., if we are dealing with the argument *A0*, the two splits will be tagged as *A0* and *C-A0*. These cases accounts for only the 0.7% of the arguments.

Some of the semantically annotated datasets were inspired from the PropBank and NomBank approach. Following this, the argument labels are neutrally named as *A0, A1,*... representing the arguments a predicate can take. Adjunct arguments are labeled as *AM-TMP, AM-MNR, AM-LOC, AM-NEG,*... and are not predicate-specific. Even though an effort by the annotators is made by making the corpus consistent across the AX labels for different predicates, there is no guarantee that the meaning is always the same. E.g., *A3* for *go* is the start point but *A3* for *do* is the instrument. Therefore a single data-driven classifier trained for the *AX* argument must in fact predict substantially different roles.

The datasets were split in the usual training, development and test sections. In some languages (English, German and Czech) out-of-domain data was provided only for testing purposes. An extract from the *Brown* corpus was provided as a out-of-domain dataset for the English language. Consistently, in syntactic dependency parsing and SRL a significant drop in performance is observed when testing on out-of-domain data. This *domain adaptation* problem is also common in other machine learning tasks (Daumé III and Marcu, 2006).

C.1 Relevant Measures and Metrics

A stating point to define our evaluation measures and metrics is the official CoNLL-2008 Shared Task scores (Surdeanu et al., 2008). These scores allow us to compare to other state-of-the-art systems and are commonly reported in SRL and dependency parsing data-driven research.

C.1.1 Syntactic scoring

The CoNLL-2008 and 2009 Shared Tasks brought in a convenient way a set of common evaluation measures and scripts to automatically compute these

measures.

Labeled attachment score (LAS) measures the percentage of tokens with a correct head (i.e., a correct dependency arc) and *also* a correct syntactic label.

Unlabeled attachment score (UAS) measures the percentage of tokens with a correct head (i.e., only a correct dependency arc).

Labeled accuracy score (LAC) measures the percentage of tokens with a correct syntactic label without regarding if the dependency points to a correct or incorrect head.

The main measure to compare syntax across systems is the *LAS*.

C.1.2 Semantic scoring

The semantic arguments are considered as semantic dependencies between the predicate and each argument, i.e., the predicate is considered as a root and its semantic arguments as its semantic dependents. In addition, each predicate sense is considered as a dependency from a virtual *root* node to the predicate. This is merely intended to provide a unified point of view for semantic scoring. The semantic structures formed are always single-rooted connected graphs, but not necessary acyclic. This approach allows us to define a unified scoring including predicate sense predictions. Along this dissertation we do not address predicate sense disambiguation thus we will not consider it when reporting semantic scores. We briefly review the *precision* and *recall* measures.

Precision is the fraction of correctly identified/tagged arguments with respect to the number of predicted arguments.

$$\text{precision} = \frac{|\text{correct predictions}|}{|\text{total predictions}|}$$

Recall is the fraction of correctly identified/tagged arguments with respect to the total number of arguments to predict.

$$\text{recall} = \frac{|\text{correct predictions}|}{|\text{total number of predictions to made}|}$$

F$_1$ is the harmonic mean between *Precision* and *Recall*

$$\text{F}_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

We distinguish between labeled and unlabeled measures. Unlabeled measures do not regard semantic labels, they only account for the identification of the semantic link between and argument and a predicate.

Unlabeled Precision the precision of predicted unannotated semantics links.

Unlabeled Recall the recall of predicted unannotated semantics links.

Unlabeled F$_1$ the F_1 of the two previous measures.

Labeled Precision precision of the predicted semantic links and their labels.

Labeled Recall recall of the predicted semantic links and their labels.

Labeled F$_1$ the F_1 of the two previous measures.

Labeled measures consider a correct prediction when both the argument is identified and the role label is correct. In the case of predicates, according to the official scorer the predicted senses must be correct. For example[2], if a correct semantic parse is to identify a verb and 3 predicates, say:

```
verb.01: ARG0, ARG1, ARGM-TMP
```

And the system output is

```
verb.02: ARG0, ARG1, ARGM-LOC
```

[2]Example extracted from the CoNLL-2008 Shared Task website http://www.yr-bcn.es/conll2008/

The labeled precision score will be $2/4$. The incorrect sense for the verb ('01' instead of '02') accounts for one error.

Lastly, another semantic measure accounts for the complete correct semantic annotations.

Perfect Proposition $\mathbf{F}_1$ scores the entire semantic frame. It is computed as the F_1 of the completely correct set of arguments and sense for each predicate. Note that in the setting where predicates are not provided, it cannot be computed as the percentage of semantic propositions with all their arguments and sense correct as we can overpredict or underpredict the predicates that defines each semantic proposition.

A semantic frame comprises a given predicate and all its arguments. Note that in other contexts a semantic frame also refer to the set of admissible arguments for a predicate. The main measure to compare semantics across systems is the *Labeled F_1*.

C.1.3 Global scoring

Finally, several other measures combine the syntactic and semantic scores.

Exact Match is the percentage of sentences that are completely correct, including syntactic dependencies, semantic dependencies and predicates.

Labeled Macro $\mathbf{F}_1$ This measure is computed using the F_1 averaging of the *Labeled macro precision* and *Labeled macro recall* which are

$$\text{LabeledMacroPrecision} = \text{LabelSemanticPrecision} + \text{LAS}$$

$$\text{LabeledMacroRecall} = \text{LabelSemanticRecall} + \text{LAS} \quad .$$

Micro $\mathbf{F}_1$. This measure is computed considering all syntactic and semantic dependencies within the same bag, i.e., each prediction is considered an individual problem, precision and recall are computed and finally the F_1 score.

semantic labeled F_1 / syntactic LAS . This measure is intended to measure the relative performance of the semantic component. The measure tries to capture the performance of the semantic component with respect to the syntactic parse. As the syntactic parse could significantly affect the semantic component this measure is aimed to give a more fair comparison of the semantic components of the systems.

REFERENCES

Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. 1986. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley.

Hans C. Boas. 2002. Bilingual framenet dictionaries for machine translation. In *Language Resources and Evaluation*.

Bernd Bohnet. 2009. Efficient parsing of syntactic and semantic dependency structures. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 67–72, Boulder, Colorado, June. Association for Computational Linguistics.

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a ccg parser. In *In Proceedings of the 20th International Conference on Computational Linguistics (COLING '04*, pages 1240–1246.

Sabine Buchholz, Erwin Marsi, Amit Dubey, and Yuval Krymolowski. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-2006)*.

Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Padó, and Manfred Pinkal. 2006. The SALSA corpus: a German corpus resource for lexical semantics. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-2006)*, Genoa, Italy.

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

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the conll-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL-2004*, pages 89–97. Boston, MA, USA.

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. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Wanxiang Che, Zhenghua Li, Yuxuan Hu, Yongqiang Li, Bing Qin, Ting Liu, and Sheng Li. 2008. A cascaded syntactic and semantic dependency parsing system. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 238–242, Manchester, England, August. Coling 2008 Organizing Committee.

Wanxiang Che, Zhenghua Li, Yongqiang Li, Yuhang Guo, Bing Qin, and Ting Liu. 2009. Multilingual dependency-based syntactic and semantic parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 49–54, Boulder, Colorado, June. Association for Computational Linguistics.

Enhong Chen, Liu Shi, and Dawei Hu, 2008. *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, chapter Probabilistic Model for Syntactic and Semantic Dependency Parsing, pages 263–267. Coling 2008 Organizing Committee.

Y. J. Chu and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica*, pages 1396–1400.

Massimiliano Ciaramita, Giuseppe Attardi, Felice Dell'Orletta, and Mihai Surdeanu. 2008. Desrl: A linear-time semantic role labeling system. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 258–262, Manchester, England, August. Coling 2008 Organizing Committee.

Trevor Cohn and Philip Blunsom. 2005. Semantic role labelling with tree conditional random fields. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis.

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. Association for Computational Linguistics, July.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms*. The MIT Press.

Michael A. Covington. 2001. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*.

Walter Daelemans and Antal van den Bosch. 2005. *Memory-Based Language Processing*. Cambridge University Press.

Qifeng Dai, Enhong Chen, and Liu Shi. 2009. An iterative approach for joint dependency parsing and semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 19–24, Boulder, Colorado, June. Association for Computational Linguistics.

Dipanjan Das, André F. T. Martins, and Noah A. Smith. 2012. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In **SEM 2012: 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 2012)*, pages 209–217, Montréal, Canada, 7-8 June. Association for Computational Linguistics.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*.

Edsger W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

Jack Edmonds and Richard M. Karp. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, April.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*.

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.

Charles J. Fillmore, Josef Ruppenhofer, and Collin F. Baker, 2004. *FrameNet and Representing the Link between Semantic and Syntactic Relations*, pages 19–62. Language and Linguistics Monographs Series B. Institute of Linguistics, Academia Sinica, Taipei.

Ruifang Ge and Raymond Mooney, 2005. *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, chapter A Statistical Semantic Parser that Integrates Syntax and Semantics, pages 9–16. Association for Computational Linguistics.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *EEE Transactions on Pattern Analysis and Machine Intelligence*.

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. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Jan Hajič, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, and Zdeněk Žabokrtský. 2006. Prague Dependency Treebank 2.0.

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), June 4-5*, Boulder, Colorado, USA.

James Henderson, Paola Merlo, Gabrielle Musillo, and Ivan Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *Proceedings of CoNLL-2008 Shared Task*.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Comput. Linguist.*, 39(4):949–998, December.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. *Advances in Kernel Methods - Support Vector Learning, Bernhard Scholkopf, Christopher J. C. Burges, and Alexander J. Smola eds., MIT Press, Cambridge, USA, 1998.*

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic–semantic analysis with propbank and nombank. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 183–187, Manchester, England, August. Coling 2008 Organizing Committee.

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. Association for Computational Linguistics.

Richard M. Karp. 1972. *Reducibility Among Combinatorial Problems*. Complexity of Computer Computations.

Daisuke Kawahara, Sadao Kurohashi, and Kôiti Hasida. 2002. Construction of a Japanese relevance-tagged corpus. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, pages 2008–2013, Las Palmas, Canary Islands.

Fang Kong, Yancui Li, Guodong Zhou, Qiaoming Zhu, and Peide Qian. 2008. Using semantic roles for coreference resolution. *Advanced Language Processing and Web Information Technology, International Conference on*, 0:150–155.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11, Uppsala, Sweden, July. Association for Computational Linguistics.

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. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Marco Kuhlmann. 2014. Linköping: Cubic-time graph parsing with a simple scoring scheme. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 395–399, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

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

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Morgan & Claypool Publishers.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1381–1391, Baltimore, Maryland, June. Association for Computational Linguistics.

Junhui Li, Guodong Zhou, and Hwee Tou Ng. 2010. Joint syntactic and semantic parsing of chinese. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1108–1117, Stroudsburg, PA, USA. Association for Computational Linguistics.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 590–599, Portland, Oregon, USA, June. Association for Computational Linguistics.

Xavier Lluís and Lluís Màrquez. 2008. A joint model for parsing syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 188–192, Manchester, England, August. Coling 2008 Organizing Committee.

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. Association for Computational Linguistics.

Xavier Lluís, Xavier Carreras, and Lluís Màrquez. 2013. Joint Arc-factored Parsing of Syntactic and Semantic Dependencies. *Transactions of the Association for Computational Linguistics (TACL)*, 1(1):219–230, May.

Xavier Lluís, Xavier Carreras, and Lluís Lei Màrquez. 2014. A shortest-path method for arc-factored semantic role labeling. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Qatar, August.

Luís Màrquez, Pere R. Comas, Jesús Giménez, and Neus Català. 2005. Semantic role labeling as sequential tagging. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*.

Lluís Màrquez, Marta Recasens, and Emili Sapena. 2013. Coreference resolution: An empirical study based on semeval-2010 shared task 1. *Lang. Resour. Eval.*, 47(3):661–694, September.

André F. T. Martins and Mariana S. C. Almeida. 2014. Priberam: A turbo semantic parser with second order features. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 471–476, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

André F. T. 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. Association for Computational Linguistics.

André F. T. Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 617–622, Sofia, Bulgaria, August. Association for Computational Linguistics.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Empirical Methods in Natural Language Processing and Natural Language Learning (EMNLP-CoNLL 2007)*.

Ryan McDonald and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(1), March.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *International Conference on Parsing Technologies (IWPT 2007)*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. 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. Association for Computational Linguistics.

Ryan McDonald, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorith. In *Human Language Technologies and Empirical Methods in Natural Language Processing HLT-EMNLP 2005*.

Gabor Melli, Zhongmin Shi, Yang Wang, Yudong Liu, Anoop Sarkar, and Fred Popowich. 2006. Description of squash, the sfu question answering summary handler for the duc-2006 summarization task. In *DUC-2006 Summarization Task*.

Igor A. Mel'čuk. 1981. Meaning-text models: A recent trend in soviet linguistics. In *Annual Review of Anthropology 10*, pages 27–62.

Igor A. Mel'čuk. 1998. *Dependency Syntax: Theory and Practice*. State University of New York Press.

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. Association for Computational Linguistics.

Roser Morante, Vincent Van Asch, and Antal van den Bosch. 2009a. Dependency parsing and semantic role labeling as a single task. In *Proceedings of the International Conference RANLP-2009*, pages 275–280, Borovets, Bulgaria, September. Association for Computational Linguistics.

Roser Morante, Vincent Van Asch, and Antal van den Bosch. 2009b. Joint memory-based learning of syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 25–30, Boulder, Colorado, June. Association for Computational Linguistics.

Erwan Moreau and Isabelle Tellier. 2009. The crotal srl system : a generic tool based on tree-structured crf. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 91–96, Boulder, Colorado, June. Association for Computational Linguistics.

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.

Gabriele Musillo and Paola Merlo. 2006. Accurate parsing of the proposition bank. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 101–104, New York City, USA, June. Association for Computational Linguistics.

Jason Naradowsky, Sebastian Riedel, and David Smith. 2012. Improving nlp through marginalization of hidden syntactic structure. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 810–820, Jeju Island, Korea, July. Association for Computational Linguistics.

Srini Narayanan and Sanda Harabagiu. 2004. Question answering based on semantic structures. In *International Conference on Computational Linguistics*.

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. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007a. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the 11th Conference on Computational Natural Language Learning (CoNLL-2007)*.

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

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*.

Joakim Nivre. 2006. Constraints on non-projective dependency parsing. In *EACL*, pages 73–80.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Martha Palmer and Nianwen Xue. 2009. Adding semantic roles to the Chinese Treebank. *Natural Language Engineering*, 15(1):143–172.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of sematnic roles. *Computational Linguistics*, 31(1):71–105, March.

Martha Palmer, Paul Kingsburry, and Daniel Gildea. 2006. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*.

Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of Coling 2004*.

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.

Corentin Ribeyre, Eric Villemonte de la Clergerie, and Djamé Seddah. 2014. Alpage: Transition-based semantic graph parsing with syntactic features. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 97–103, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

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. Association for Computational Linguistics.

Alexander M. Rush and Michael Collins. 2012. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *Journal of Artificial Intelligence Research*, 45:305–262.

Alexander M. Rush and Slav Petrov. 2012. Vine pruning for efficient multi-pass dependency parsing. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 498–507, Montréal, Canada, June. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Yvonne Samuelsson, Oscar Täckström, Sumithra Velupillai, Johan Eklund, Mark Fishel, and Markus Saers, 2008. *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, chapter Mixing and Blending Syntactic and Semantic Dependencies, pages 248–252. Coling 2008 Organizing Committee.

Alexander Schrijver. 1998. *Theory of Linear and Integer Programming*. John Wiley & sons.

Libin Shen and Aravind K. Joshi. 2005. Ranking and reranking with perceptron. *Machine Learning*.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 12–21.

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.

Weiwei Sun, Hongzhan Li, and Zhifang Sui, 2008. *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, chapter The Integration of Dependency Relation Classification and Semantic Role Labeling Using Bilayer Maximum Entropy Markov Models, pages 243–247. Coling 2008 Organizing Committee.

Mihai Surdeanu, Sanda M. Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Meeting of the Association for Computational Linguistics*, pages 8–15.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*.

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. Coling 2008 Organizing Committee.

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

Robert E. Tarjan. 1977. Finding optimum branchings. *Networks*.

Mariona Taulé, Maria Antònia Martí, and Marta Recasens. 2008. AnCora: Multilevel Annotated Corpora for Catalan and Spanish. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC-2008)*, Marrakesh, Morroco.

Ivan Titov and James Henderson. 2007a. Incremental bayesian networks for structure prediction. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 887–894, New York, NY, USA. ACM.

Ivan Titov and James Henderson. 2007b. A latent variable model for generative dependency parsing. In *Proceedings of the International Conference on Parsing Technologies (IWPT-07)*.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online graph planarisation for synchronous parsing of semantic and syntactic dependencies. In *Proceedings of ICJAI*, pages 1562–1567.

N. Tomizawa. 1971. On some techniques useful for solution of transportation network problems. *Networks*, 1(2):173–194.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 589–596, Stroudsburg, PA, USA. Association for Computational Linguistics.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

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

Yotaro Watanabe, Masayuki Asahara, and Yuji Matsumoto. 2009. Multilingual syntactic-semantic dependency parsing with three-stage approximate max-margin linear models. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 114–119, Boulder, Colorado, June. Association for Computational Linguistics.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967, Prague, Czech Republic, June. Association for Computational Linguistics.

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. Association for Computational Linguistics.

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.

Szu-ting Yi and Martha Palmer. 2005. The integration of syntactic parsing and semantic role labeling. In *Proceedings of CoNLL-2005*.

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

Hao Zhang and Ryan McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 320–331, Jeju Island, Korea, July. Association for Computational Linguistics.

Hai Zhao and Chunyu Kit. 2008. Parsing syntactic and semantic dependencies with two single-stage maximum entropy models. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 203–207, Manchester, England, August. Coling 2008 Organizing Committee.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009a. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*

(CoNLL 2009): Shared Task, pages 61–66, Boulder, Colorado, June. Association for Computational Linguistics.

Hai Zhao, Wenliang Chen, Chunyu Kity, and Guodong Zhou. 2009b. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 55–60, Boulder, Colorado, June. Association for Computational Linguistics.