Dependency-based Semantic Analysis of Natural-language Text

Richard Johansson

Doctoral dissertation, 2008

Department of Computer Science
Lund University

Graduate School of Language Technology

ISBN 978-91-628-7628-9
ISSN 1404-1219
Dissertation 30, 2008
LU-CS-DISS:2008-3

Department of Computer Science
Lund University
Box 118
SE-221 00 Lund
Sweden

Email: `richardj@df.lth.se`
WWW: `http://www.df.lth.se/~richardj`

Typeset using LaTeX 2_ε

Printed at Tryckeriet i E-huset, Lunds Tekniska Högskola.

Abstract

Semantic roles, logical relations such as AGENT or INSTRUMENT that hold between events and their participants and circumstances, need to be determined automatically by several types of applications in natural language processing. This process is referred to as *semantic role labeling*. This dissertation describes how to construct statistical models for semantic role labeling of English text, and how role semantics is related to surface syntax.

It is generally agreed that the problem of semantic role labeling is closely tied to syntactic analysis. Most previous implementations of semantic role labelers have used constituents as the syntactic input, while *dependency* representations, in which the syntactic structure is viewed as a graph of labeled word-to-word relations, has received very little attention in comparison. Contrary to previous claims, this work demonstrates empirically that dependency representations can serve as the input for semantic role labelers and achieve similar results. This is important theoretically since it makes the syntactic–semantic interface conceptually simpler and more intuitive, but also has practical significance since there are languages for which constituent annotation is infeasible.

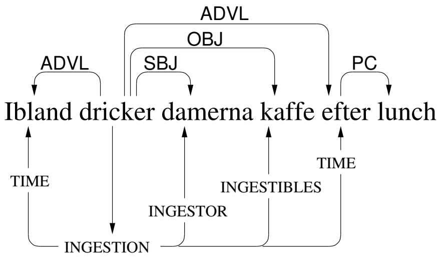
The dissertation devotes considerable effort to investigating the relation between syntactic representation and semantic role labeling performance. Apart from the main result that dependency-based semantic role labeling rivals its constituent-based counterpart, the empirical experiments support two findings: First, that the dependency-syntactic representation has to be well-designed in order to achieve a good performance in semantic role labeling. Secondly, that the choice of syntactic representation affects the substages of the semantic role labeling task differently; above all, the role classification task, which relies strongly on lexical features, is shown to benefit from dependency representations.

The systems presented in this work have been evaluated in two international open evaluations, in both of which they achieved the top result.

Sammanfattning

Denna avhandling beskriver hur man kan konstruera statistiskt baserade datorprogram som analyserar text.

Analysen sker på två nivåer: den *syntaktiska*, som beskriver de grammatiska sambanden mellan orden i en meningen, och den *semantiska*, som beskriver de betydelsemässiga sambanden. Vi kan åskådliggöra dessa samband genom diagram – grafer – som i figuren nedan.



För att ta ett exempel kan vi betrakta ordet *dricker*. I den semantiska grafen (under meningen) kan vi se att detta ord uttrycker en situation som vi kan kalla INGESTION[1]. Detta ord har betydelsemässiga samband med andra ord i meningen: *damerna* fungerar som INGESTOR, alltså den som dricker, och *kaffe* som INGESTIBLES, det som blir uppdrucket. Även dessa samband kan vi läsa i den semantiska grafen. På motsvarande sätt kan vi läsa av syntaktiska samband i grafen som visas över meningen. Vi kan se att *damerna* fungerar som grammatiskt subjekt (SBJ) för *dricker* och *kaffe* som objekt (OBJ).

Det är uppenbart att de två graferna har någon typ av samband. I meningen ovan motsvarades till exempel det grammatiska subjektet av den som dricker, INGESTOR. Dessa samband är vad avhandlingen

[1]Symbolerna INGESTION, INGESTOR, etc. är tagna från databasen FrameNet.

undersöker. Till att börja med beskriver vi några principer för hur sambanden på den syntaktiska nivån ska beskrivas, och vilken betydelse detta har för den semantiska analysen. Därefter undersöker vi hur vi kan konstruera statistiska modeller för att bygga den semantiska analysen utifrån den syntaktiska. Slutligen jämför vi olika typer av syntaktiska grafer med avseende på hur lätt det är att bygga semantiska grafer. De datorprogram som beskrivs i avhandligen har deltagit i två internationella utvärderingar. I båda utvärderingarna fick våra system det högsta resultatet av de deltagande.

Automatisk syntaktisk och semantisk analys kan vara till nytta i olika datorprogram som hanterar text. Ett ofta anfört exempel är *informationsextraktion*. *Frågebesvarande* system är en annan tillämpning. Det kan också användas i *sammanfattning*, alltså där datorns uppgift är att söka upp de mest relevanta styckena i en större text. *Textkategorisering* är en annan tillämpning där syntaktisk och semantisk analys kan vara till nytta. Textkategorisering innebär att datorprogrammet avgör vilken typ av text man har att göra med, t.ex. ekonomi, sport eller vetenskap. Det finns också ett antal tänkbara tillämpningar som hittills inte undersökts. Till dessa hör *informationssökning*.

Acknowledgements

My first and foremost thanks goes to Pierre Nugues, my supervisor, for introducing me to the topic of natural language processing. Pierre has a rare talent for instilling the qualitites that are needed by every researcher: enthusiasm and independent thinking.

I am deeply grateful to Joakim Nivre for making use of my work in the 2007 and 2008 CoNLL Shared Tasks, and for including me in the organizing committee of the 2008 Shared Task. I also wish to thank the other members in that committee for stimulating discussions, from which I learned a lot: Mihai Surdeanu, Lluís Màrquez, and Adam Meyers.

I would like to thank the students who carried out their thesis work with us: Anders Berglund, Magnus Danielsson, Jacob Persson, and Dan Thorin. All four theses resulted in substantial research results.

The Department of Computer Science at LTH has provided a friendly and relaxed research environment.

The Swedish Graduate School of Language Technology (GSLT) funded my research during 2006–2008. I was very lucky to be accepted to the graduate school, and their funding started at a critical point. It is very uncertain how this research would have turned out without it. The GSLT is also a very friendly and diverse group of people.

During 2003 and 2004, my position was funded by grant number 2002-02380 from the Language Technology program of Vinnova.

A significant part of the research described here concerns FrameNet, kindly provided by ICSI Berkeley via an academic license.

My final acknowledgement goes to my family: my father Bo, mother Barbro, brother Robert, and to the Saar family: my partner Piret and her parents Henn and Svetlana.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The representation of *semantic roles*, the logical relations that hold between an event and its participants, is needed in many applications in natural language processing. *Semantic role labeling*, the process of automatically extracting role-semantic structures, has recently been studied intensely. It has obvious applications for template-filling tasks such as information extraction and has also been applied in question answering, entailment recognition, summarization and text categorization. While the task of full natural language understanding is poorly understood in general, semantic role structures constitute a tractable fragment of the full spectrum of human semantic processing.

The recent advances in automatic semantic role labeling have been made possible by large-scale annotation projects that have resulted in role-semantically annotated resources such as FrameNet and PropBank (Baker, Fillmore, and Lowe, 1998; Palmer, Gildea, and Kingsbury, 2005; Xue and Palmer, 2007). These corpora have made it feasible to apply statistical techniques to the problem (Gildea and Jurafsky, 2002).

From the start, it has been assumed that automatic role-semantic annotation must be performed on top of a *syntactic representation*, a data structure that describes how an arrangement of surface words form a complete grammatical sentence. This has sometimes been contested (Collobert and Weston, 2007) but still seems to be the received wisdom of the field (Gildea and Palmer, 2002; Punyakanok, Roth, and Yih, 2008).

The question then arises on what information the syntactic structure should represent in order to be practical for automatic semantic role labeling. We aim to find a *parsimonious* representation – one that is *expressive* enough to allow us to perform semantic analysis on top of it, but is also *economical* as possible, meaning that it contains no redundant structure.

The central claim of this dissertation is that automatic role-semantic analysis can be performed using a *dependency representation* of syntax. In a dependency representation, syntactic relationships between words are represented as a graph of labeled edges between words. This representation of syntax emphasizes *function* – the label on an edge between two words represents how the words cooperate to form a complete structure. This is to be contrasted with the more widely used constituent structures, which represent the hierarchical organization of phrases but not their grammatical function.

We claim that for the particular task of automatic role-semantic analysis, dependency representations are both expressive and economical, and thus deserve more attention than they have been given until now. We carry out a number of experiments in semantic role labeling of English to support our claims. However, not just any dependency-syntactic representation will be good enough to serve as the syntactic input of a semantic role labeler. We will thus devote considerable effort to carefully define what to represent in the dependency graphs: which nodes to connect and the directions and labels of the dependency arcs.

To estimate parameters in statistical dependency parsers, a collection of syntactically annotated sentences – a *treebank* – is needed. However, since no dependency treebank for English exists, we are forced to create dependency structures automatically from the Penn Treebank, a constituent treebank. This limits what we can achieve by automatic methods alone, since we cannot extract information that is not represented in the Penn Treebank.

To empirically demonstrate the parsimony of the dependency representation for the task of semantic role labeling, we carry out a series of evaluations where we compare it to other types of syntactic representations, all based on constituents in some form. Since the results are as good as with constituent representations, this constitutes an empirical demonstration of the *expressivity* of the representation – since the results are equivalent – and the *economy* – since a dependency tree lacks phrases and thus has fewer nodes and edges than a constituent tree.

Example

Figure 1.1 shows how the sentence *Chrysler plans new investment in Latin America* is represented using dependency-syntactic (above the text) and role-semantic links (below). The syntactic dependency structure represents, for instance, that the main verb of the sentence, *plans*, has a subject *Chrysler* and an object *investment*. The corresponding semantic structure contains a semantic predicate corresponding to the verb *plans* – we represent this predicate using a word sense label `plan.01`, meaning that this word is an instance of the first sense of *plan* described in the lexicon. Two semantic arguments directly corresponding to its syntactic counterparts are connected to the predicate: *Chrysler* is marked as argument 0, and if we refer to the lexical entry of *plan*, we see that this argument corresponds to the *planner*, the active participant in the act of planning. Similarly, *investment* is marked as argument 1, corresponding to the *thing planned*. The word *investment* is itself a predicate having two arguments: an *investor* (*Chrysler*, argument 0) and *purpose* (*Latin America*, argument 2).

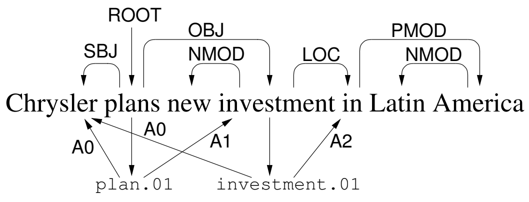


Figure 1.1: *Syntactic and role-semantic representations of a sentence.*

1.1 Overview of this Dissertation

The rest of this dissertation is organized as follows:

Chapter 2: Role Semantics and Its Applications. This chapter formally defines the concepts of semantic roles and frames, gives an overview of role-semantic linguistic resources, and reviews published work on role semantics in practical NLP applications.

Chapter 3: Dependency-syntactic Representations. Here, we turn to the question of describing the organizational relations that hold between words in a sentence. We argue that dependency graphs are expressive and economical as the input representation when predicting role-semantic structures.

Chapter 4: Automatic Construction of an English Dependency Treebank. This chapter describes how dependency structures can be automatically extracted from a constituent treebank. To justify the design decisions, we apply the principles laid down in Chapter 3 to analyze a number of nontrivial constructions.

Chapter 5: Dependency-based Role-Semantic Analysis. The treebank created in the previous chapter forms the basis of the work described in this chapter, which describes the implementation of automatic role-semantic analyzers for English in the FrameNet, PropBank, and NomBank frameworks. Features for statistical classifiers are described, and a number of evaluation metrics are introduced.

Chapter 6: Comparing Syntactic Representations for Automatic Role-semantic Analysis. The experiments in this chapter compare the effect of syntactic representation on semantic role labeling performance. We compare dependency-based and constituent-based semantic role labelers, and investigate the effect of the design of the dependency representation.

Chapter 7: Extensions of the Classifier-based Model. In this chapter, we describe three extensions to the basic model: linguistic constraints, reranking of complete predicate–argument structures, and integration of syntactic and semantic analysis.

Chapter 8: Conclusion. This chapter concludes the dissertation by summarizing the main points and describing possible future directions.

1.2 Published Work

The core parts of the material describing dependency-syntactic representations and their conversion from constituents was first published in this paper:

Johansson, Richard and Pierre Nugues. 2007a. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, Tartu, Estonia.

The syntactic framework was then revised considerably for the CoNLL-2008 Shared Task. This is described in the CoNLL-2008 shared task introduction paper:

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

The description of the classifier-based semantic role labeling architecture was first published in the SemEval task on Frame-semantic Structure Extraction:

Johansson, Richard and Pierre Nugues. 2007b. Semantic structure extraction using nonprojective dependency trees. In *Proceedings of SemEval-2007*, Prague, Czech Republic.

Preliminary experiments investigating the differences between syntactic representations were published in this article:

Johansson, Richard and Pierre Nugues. 2007c. Syntactic representations considered for frame-semantic analysis. In *Proceedings of the Sixth International Workshop on Treebanks and Linguistic Theories*, Bergen, Norway.

The extended experiments, which form the basis of the material in Chapter 6, were described in this article:

Johansson, Richard and Pierre Nugues. 2008c. The effect of syntactic representation on semantic role labeling. In *Proceedings of COLING*, Manchester, United Kingdom.

The extensions to the classifier-based semantic role labeler have been described in two papers, the first of which used a dependency-based evaluation metric and the second one a segment metric:

Johansson, Richard and Pierre Nugues. 2008b. Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *Proceedings of the Shared Task Session of CoNLL-2008*, Manchester, United Kingdom.

Johansson, Richard and Pierre Nugues. 2008a. Dependency-based semantic role labeling of PropBank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chapter 2

Role Semantics and Its Applications

We model the process of communication as an exchange and harmonization of structures of meaning: semantic structures. The natural languages that humans use are a tool for encoding these structures using the limited expressive capabilities of our speech apparatus. The relations between natural language sentences and their corresponding semantic structures are the topic of this dissertation.

To be able to make meaningful observations about the semantic structures, we need to make the assumption that they can be *decomposed* into smaller building blocks which we can study independently. This is the well-known *principle of compositionality*, which has been argued for and against since antiquity.[1] It is well known and very obvious that this principle is unrealistic in the general case. However, the work that we describe here will concern only the representation of information in language that is to be taken completely literally, completely disregarding its rhetorical context and all possible interpretations of humor, sarcasm, or allegory.

The core assumption in the principle of compositionality is that complex structures are formed by combining smaller structures. This decomposition is usually hierarchical, so that some semantic elements – which we refer to as *functors* – require other elements – the *arguments* – to form complete structures. Every functor has a number of *slots*

[1]It was known to early Indian philosophers such as Yāska. Plato proposed – and refuted – the principle in *Theaetetus*. Later Indian grammarians debated it intensely.

for arguments. We refer to these complex terms as *functor–argument* structures. Such representations are widespread, and it has been conjectured that they have a neural basis (Hurford, 2003).

Given a world, the functor–argument structures may be interpreted as statements about this world. The functors are then referred to as *predicates* in a predicate logic. Predicate logic is the foundation of virtually all formalisms used in knowledge representation and theories of natural-language semantics. However, since this dissertation concerns only the superficial *construction* of the logical formulae given a text, not how they relate to the world or how new facts are inferred from them, we will use the word *predicates* to refer to functors throughout, although this is an abuse of the term. This is standard practice in the NLP field.

To illustrate the representational formalisms that we will describe in this chapter, we will refer to the situation denoted by the following sentence:

Example 2.1. Alexander eats an olive.

If we assume that the sentence is intended to be read completely literally, without metaphors or irony, a formulation of Example 2.1 in an event-based predicate logic (Davidson, 1967) might be

$$\exists x, e : \text{olive}(x) \wedge \text{eat}(e, \text{Alexander}, x)$$

in addition to some temporal description of the event e.

In this work, our effort will be spent on finding event predicates and their arguments. In the logical representations, we will omit entity predicates, quantifiers, and representation of temporal, aspectual, and modal structure. This simplification of the logical representation is often referred to as a *shallow semantic representation*. We thus rewrite the full representation of the example to the following simplified formula:

$$\text{eat}(e, \text{Alexander}, \text{olive})$$

To be able to reason about the relations between predicates and arguments, we introduce auxiliary predicates (which we will refer to as *slot connectors*) that connect the event entity e to its participants. Then the formula above becomes

$$\text{eat}(e) \wedge \textsc{FirstArgOfEat}(e, \text{Alexander}) \wedge \textsc{SecondArgOfEat}(e, \text{olive})$$

2.1 Role Semantics

Role semantics is the assumption that slot connectors associated with different predicates can be meaningfully grouped into equivalence classes – *semantic roles*[2]. If we are speaking about language, these classes are then assumed to be reflected in surface realizations.

For example, we might say that the relation between *Alexander* and *eats* is in some way "the same" as that between *Barbara* and *puts on* in *Barbara puts on a hat*, i.e. that of a sentient being who intentionally instigates the event, and who is the focus of the narrative. The linguistic rationale behind this intuition is that the arguments connected by these slot connectors tend to appear as surface subjects in active clauses. Of course, this "sameness" can only be rough since language discretizes a continuous world.

Definition 2.1. A *role-semantic equivalence* is an equivalence relation over slot connectors, and a *semantic role* is an equivalence class under a role-semantic equivalence relation.

Introducing descriptive names for the equivalence classes, this allows us to rewrite the example as follows:

$$\text{eat}(e) \wedge \textsc{Agent}(e, \text{Alexander}) \wedge \textsc{Theme}(e, \text{olive})$$

It is also customary to group semantic roles into *core* roles and *peripheral* roles. Core arguments for a predicate are semantically central participants, while peripheral roles "set the scene." The set of core arguments allowed by a predicate is called its *semantic valency*, using a metaphor from chemistry popularized by Tesnière (1959)[3].

2.1.1 Role Semantics in Linguistic Theory

In linguistic theories, semantic roles have been used as a device to explain the process of *linking*: the realization of semantic arguments as surface forms. As mentioned above, for instance, when explaining why one semantic argument of a predicate appears on the surface as a subject, it comes very naturally to posit a general semantic category similar to AGENT. The purpose of this abstraction is to allow us to *generalize* over predicates when reasoning about predicate–argument relations.

There are several linguistic transformations, some of which are or have formerly been explicitly marked in morphology, which

[2]The term *thematic role* or *θ-role* (*theta role*) is also common. Depending on context and tradition, the terms *semantic*, *thematic*, and θ role may have slightly different meanings and may not be completely interchangeable.

[3]The metaphor seems to have been around in Russian and German linguistics before Tesnière, see Schubert (1987).

can be most conveniently defined using a role-based framework: active/passive (*to tell/to be told*), inchoative/causative (*to rise/to raise, to fall/to fell*), and various diathesis alternations (*smeared paint on the wall/smeared the wall with paint*). It has been claimed that the semantic role categories are universal and possibly innate, and this could explain why native speakers intuitively know what grammatical construction to use for previously unseen words such as *to fax* or *to fedex*, depending on the semantics of the word. The research investigating these questions has primarily focused on the relations between verbs and their arguments (a prototypical example is Levin, 1993), although there also have been some less verb-centered frameworks. Semantic roles are now found in many theoretical descriptions, and they have even found their way into mainstream generative linguistics (Chomsky, 1981). For a general overview of current role-based research in typology, theoretical linguistics, and neurolinguistics, see Bornkessel et al. (2006).

The earliest known example of role semantics in a linguistic setting is the *kāraka* system of Pāṇini, found in *Āṣṭādhyāyī*, an early grammar of Sanskrit (∼500 BC). This system was used in the semantic representation given as input to an algorithm (rewriting system) to carry out the transition from semantics to surface morphology. This representation used the following *kāraka*, factors which have a function in the accomplishment of action (*kriyā*).

Apādāna. "That which is firm when departure takes place." (source)

Saṃpradāna. "He whom one aims at with the object." (dative/benefactor/target)

Karaṇa. "That which effects most." (instrument)

Adhikaraṇa. "Location."

Karman. "What the agent seeks most to attain." (objective/theme)

Kartṛ. "He/that which is independent in action." (agent)

A similar purpose – explaining the interface between surface and deep structure – made Fillmore (1968), in the context of generative–transformational grammatical theory, propose the following list of "deep cases", although he also included the caveat "additional cases will surely be needed." Note the similarity to the above list.

Agentive. "The case of the typically animate perceived instigator of the action identified by the verb."

Instrumental. "The case of the inanimate force or object causally involved in the action or state identified by the verb."

Dative. "The case of the animate being affected by the state or action identified by the verb."

Factitive. "The case of the object or being resulting from the action or state identified by the verb, or understood as a part of the meaning of the verb."

Locative. "The case which identifies the location or spatial orientation of the state or action identified by the verb."

Objective. "The semantically most neutral case, the case of anything representable by a noun whose role in the action or state identified by the verb is identified by the semantic interpretation of the verb itself [. . .]"

Both lists were derived from surface systems of grammatical cases, hence Fillmore's terminology. However, they arrived at these systems from very different starting points: Pāṇini used the *kāraka* system as a convenient device to simplify the exposition of a grammar for a particular language, without any claim of universality or cognitive grounding, while Fillmore was trying to to establish universal properties of all languages: "a set of universal, presumably innate, concepts which identify certain types of judgments human beings are capable of making about the events that are going on around them" (Fillmore, 1968, p. 24).

2.2 Frame Semantics

Pāṇini and Fillmore posited *universal* sets of semantic roles. The universal semantic roles, and their scientific grounds, have been challenged repeatedly. First of all, there has been no universal set that has been generally agreed upon – using the terminology introduced above, it is notoriously difficult to find *meaningful* role-semantic equivalence classes that cover all slot connectors in all predicates. It has not been possible to give strict definitions of the roles, and as we saw above, the lists of semantic roles of Pāṇini and Fillmore both include an ill-defined "catch-all" category (*karman*/objective). The problems appear both when defining the roles on the semantic level and when describing the link to the surface-linguistic level. Symmetrical predicates such as *resemble* and pairs like *buy/sell* are examples that are difficult to fit into the framework of universal roles. For a summary of criticisms of the concept of semantic role, see Ratté (1994).

To circumvent the definitional problems of universal semantic role sets, Fillmore (1976) proposed the concept of *semantic frame* – a set of predicates that share the same role set.

Definition 2.2. A *semantic frame* is a pair $F = \langle L, \sim_F \rangle$ such that L is a set of predicates, and $\sim_F$ is a role-semantic equivalence over the slot connectors for the predicates in L.

While the formal requirement is only that the set of predicates shares a set of semantic roles, linguists who group predicates into frames usually make a stronger assumption: that the predicates relate to a shared "situation" or domain:

> A word's meaning can be understood only with reference to a structured background of experience, beliefs, or practices, constituting a kind of conceptual prerequisite for understanding the meaning. Speakers can be said to know the meaning of the word only by first understanding the background frames that motivate the concept that the word encodes. Within such an approach, words or word senses are not related to each other directly, word to word, but only by way of their links to common background frames and indications of the manner in which their meanings highlight particular elements of such frames.
>
> Fillmore and Atkins (1992), pp. 76–77.

For instance, in FrameNet, a frame-semantic lexical database (see 2.3.1), the word *eat* belongs to the frame INGESTION, which it shares with words (verbs and nouns) such as *devour*, *drink*, *gulp*, ... Conventionally, descriptive frame-specific names such as INGESTOR are then given to the semantic roles, rather than abstract names such as AGENT. We thus write the example as follows:

$$\text{INGESTION:eat}(e) \wedge \text{INGESTOR}(e, \text{Alexander}) \wedge \text{INGESTIBLES}(e, \text{olive})$$

In this formulation, frame semantics has less predictive power than role semantics with universal roles, and to be able to describe general linguistic processes such as diathesis alternation, frame-semantic theories may introduce frame-to-frame relations such as inheritance from more abstract frames, which may then contain abstract roles such as AGENT.

This is the case with FrameNet, for instance. Of course, care must be taken so that these relations do not just lead back to the problems that made universal role sets impractical.

It has been conjectured that a system of frames and frame-to-frame relations in a given language would be structurally similar to its counterparts in other languages, and it is this assumption that underpins the hypothesis that frame semantics could be useful in machine translation (Boas, 2002; Boas, 2005). This assumption has been used with some success to automatically construct role-semantic resources in new languages (see 2.4.3). However, investigations of frame-semantic parallelism (Padó, 2007a; Padó, 2007b) demonstrate clearly that the frame structures in a pair of languages are not generally isomorphic, not even for very closely related languages such as English and German. We can also be fairly sure that the structure parallelism decreases as typological and cultural distance grows, although this has not been investigated to our knowledge.

2.3 Role-Semantic Lexicons and Corpora

To be able to construct computer systems that automatically carry out a role-semantic analysis, significant efforts are needed to construct resources that make this possible.

- Role-semantic *lexicons* define the semantic valencies of individual lexical items.
- Annotated *corpora* allow us to create statistical models in systems that generate or analyze semantic role structures, as well as evaluating their performance.

This section gives an overview of existing role-semantic resources.

2.3.1 FrameNet, SALSA

FrameNet (Baker, Fillmore, and Lowe, 1998) is a lexical database grounded in the frame-semantic paradigm. It consists of the following parts:

- A lexicon that lists a frame for every word sense,
- A frame ontology, which defines the semantic roles for each frame and frame-to-frame relations such as inheritance, part-of, and causative-of,

- A collection of lexical examples manually sampled from the British National Corpus.

With release 1.3 of FrameNet, a small corpus of running text was also added, in order to create a corpus from which reliable statistical inferences can be drawn. However, the projects that have trained statistical models on FrameNet have utilized the lexical examples as the primary source of training data, since this collection is much larger. On a practical level, FrameNet has a problem of low coverage and incompatibility with other resources. The low coverage has led to a number of ad-hoc methods to extend the lexicon (Burchardt, Erk, and Frank, 2005; Johansson and Nugues, 2007d).

There are also projects to create FrameNets for Spanish (Subirats, 2008) and Japanese (Ohara et al., 2003). In addition, there are a number of preliminary proposals and pilot projects for a wide range of languages.

A more corpus-oriented project is SALSA (Burchardt et al., 2006). In contrast to FrameNet, this project annotated a real corpus of German, which is promising for statistical systems. The project tried to maintain compatibility with the English FrameNet as far as possible, which could make it useful in multilingual applications. Another interesting difference compared to FrameNet is that SALSA annotated directly on top of the TIGER treebank. An automatic semantic role labeler for German trained on the SALSA corpus has been developed (Erk and Padó, 2006).

2.3.2 VerbNet

VerbNet (Kipper, Dang, and Palmer, 2000) is a role-semantic lexicon for English based on the theoretical framework of Levin (1993). In VerbNet, verbs are grouped into hierarchical classes depending on which subcategorization patterns they allow. It makes stronger assumptions than FrameNet: that not only the allowed semantic arguments, but also the syntactic transformations that are allowed by a verb, are determined by the semantics of the verb.

VerbNet uses a set of universal semantic roles. Table 2.1 shows the semantic role labels used in VerbNet. The descriptions are taken from the project website.

Actor	Used for some communication classes when both arguments can be considered symmetrical.
Agent	Generally a human or an animate subject.
Asset	Used for the *Sum of Money* alternation.
Attribute	Attribute of Patient/Theme refers to a quality of something that is being changed.
Beneficiary	The entity that benefits from some action.
Cause	Used mostly by classes involving psychological verbs and verbs involving the body.
Destination	End point of motion, or direction towards which the motion is directed.
Source	Start point of the motion.
Location	Underspecified destination, source, or place.
Experiencer	A participant that is aware or experiencing something.
Extent	Specifies the range or degree of change.
Instrument	Used for objects (or forces) that come in contact with an object and cause some change in them.
Material	Start point of transformation.
Product	End result of transformation.
Patient	Participant that is undergoing a process or that has been affected in some way.
Predicate	Used for classes with a predicative complement.
Recipient	Target of the transfer.
Stimulus	Used by verbs of perception for events or objects that elicit some response from an experiencer.
Theme	Participant in a location or undergoing a change of location.
Time	Class-specific role, used in Begin-55.1 class to express time.
Topic	Topic of communication verbs to handle theme/topic of the conversation or transfer of message.

Table 2.1: *The semantic role labels used in VerbNet.*

2.3.3 PropBank and NomBank

PropBank (Palmer, Gildea, and Kingsbury, 2005) adds a layer of semantic role annotation on top of the Penn Treebank of constituent-syntactic annotation (Marcus, Santorini, and Marcinkiewicz, 1993). This framework is theoretically agnostic; the core semantic arguments are just assigned numbers (ARG0, ARG1, ...). The allowed core arguments for every verb are listed in a lexicon. In addition, adjuncts are annotated using generic labels such as ARGM-DIR (direction), ARGM-PNC (purpose/cause). The following is an analysis of the example sentence according to PropBank conventions.

$$\text{eat}(e) \land \textsc{Arg0}(e, \text{Alexander}) \land \textsc{Arg1}(e, \text{olive})$$

The semantic role labels are consistent across different diathesis alternations of the same verb. However, unlike in FrameNet or VerbNet, it is not assumed that the labels are meaningful across different verbs. A general rough convention is that ARG0 corresponds to the argument having most properties of a "proto-agent" and ARG1 to a "proto-patient" (Dowty, 1991). The other core arguments for a particular verb are numbered "by decreasing degree of prominence." For instance, ARG2 for the verb *make* corresponds to the VerbNet role MATERIAL, but for the verb *multiply*, it corresponds to EXTENT. Despite this, statistical systems that carry out a PropBank-style semantic analysis typically treat role label assignment as a well-defined classification problem, and it generally seems that semantic role labeling is easier with PropBank than with other frameworks, although it is difficult to discern to what extent this is caused by the framework itself as opposed to factors such as data quality and domain variability.

Unlike FrameNet, PropBank is defined for verbs only, although it is often possible to express a predicate as a noun. For instance, *Alexander's death* refers to the same state of affairs as *Alexander died*. NomBank (Meyers et al., 2004) is a project that addresses this issue by attempting to generalize PropBank to nominal predicates. Its framework is almost identical to PropBank's, and it tries to stay close to PropBank's role definitions, so that for instance the nominalization *death* has the same semantic role range as the verb *die*. So far, there has been little research on automatic semantic NomBank semantic role labeling – it seems to be considerably more difficult than for PropBank (Surdeanu et al., 2008).

PropBank-based semantic role labelers have been implemented for other languages than English, for instance Chinese (Xue, 2008), Arabic (Diab, Moschitti, and Pighin, 2008), Spanish, and Catalan (Surdeanu, Morante, and Màrquez, 2008).

2.3.4 Prague Dependency Treebank

The Prague Dependency Treebank (Hajič, 1998) is probably the most well-known treebank based on syntactic dependency. It consists of three layers: the morphological, the analytical (surface syntax), and the tectogrammatical ("underlying" syntax). The final of these is a semantic dependency structure that (among other things) annotates predicate–argument structure using semantic role labels such as ACT (actor) and PAT (patient). Like SALSA and PropBank, the semantic layer of the Prague Dependency Treebank has the advantage of being linked to a syntactic layer, which makes it easier to construct automatic semantic role labelers, assuming that a syntactic parser is available. A comparison of PropBank and the tectogrammatical layer has been made by Hajičová and Kučerová (2002).

2.4 Role Semantics in Practical Applications

Role semantics has been proposed as a practical intermediate structure that mediates between raw syntax and domain-specific representations. As we saw previously, it has been employed in linguistics as an auxiliary device, a transitional representation used in translating "deep" semantic structures to surface structures. In natural language processing, on the other hand, it may serve as a generic semantic representation, mediating between surface structures and application-specific representations.

The reason for the appeal of role semantics in this context is that these representations are simple enough for building computer systems that extract them automatically with a fair degree of accuracy. Since they are not full predicate-logic formulae, automatic systems can avoid resolving notoriously hard and poorly understood linguistic problems such as quantifier scope ambiguity, reference resolution, and temporal, modal, and discourse structure.

Despite their relative simplicity, these representations provide more abstraction than what is possible when using syntax only. The reason for this is that they are to a large extent invariant under paraphrasing, which was indeed the reason to introduce them in the first place. Role-semantic representations preserve structure under operations such as passivization (*they told a lie / a lie was told*) and diathesis alternations such as dative shift (*they told me a lie / they told a lie to me*). If FrameNet is used, it may also be possible to generalize from verbs such as *speak* to nouns such as *statement*.

The following subsections give a brief overview of how role semantics has been applied (or proposed to be applied) in applications.

2.4.1 Template-filling Applications

Possibly, the most obvious practical applications of automatic systems for role-semantic analysis are information extraction systems that fill templates. In these cases, it might be possible to map semantic roles directly to template slots. Figure 2.1 shows such a mapping from two FrameNet frames to a simple template, taken from the paper by Moschitti, Morărescu, and Harabagiu (2003). This example shows the advantage that the additional level of abstraction offers – details of linguistic variations are abstracted away. In other cases, a more complex mapping than in this simple example may be needed, but it can be expected that information extraction systems can be in general constructed more rapidly if semantic role information is available: Surdeanu et al. (2003) constructed such mappings manually in a couple of hours per domain.

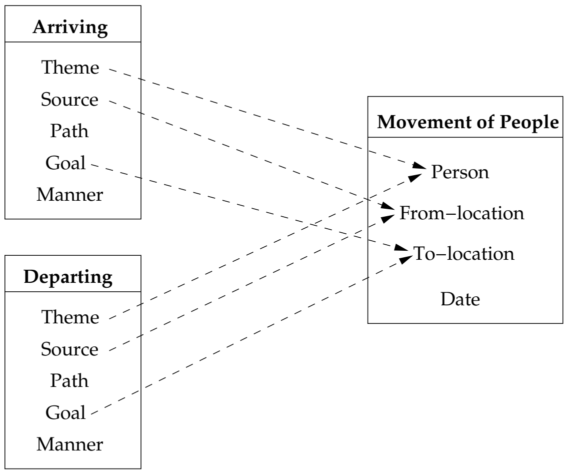


Figure 2.1: *Mapping from FrameNet frames to a template in an information extraction application.*

2.4.2 Graph-based Applications

In some applications, it it fruitful to build graphs to represent texts. These graphs can then be used to measure similarity or for locating semantically prominent parts. Role-semantic representations have a very natural place in such representations. A typical example is question answering, in which a relevant document and a relevant passage in the document are to be extracted by an automatic system. Semantic role labelers have been shown to improve performance in question answering (Narayanan and Harabagiu, 2004; Shen and Lapata, 2007).

Figure 2.2 shows how frame semantics can be used as text representations in question answering. Here, we search for a passage that can answer the question *When did Alexander marry?*, and the answer passage *...Alexander's 327 BC wedding ...* can be extracted by exploiting the semantic structure similarity.

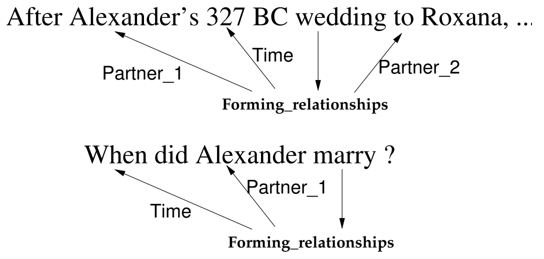


Figure 2.2: *Examples of semantic graphs in a question answering problem.*

A similar problem is entailment recognition: given a text and a statement, determine whether or not the statement is a logical consequence of the text. There are many ways to approach this problem, but it seems that methods using role-based semantic graphs work well (Haghighi, Ng, and Manning, 2005; Hickl et al., 2006). Semantic role labeling has also been applied in summarization, in which simplifying transformation operations are applied to semantic graphs (Melli et al., 2005).

2.4.3 Multilingual Applications

One of the most widely cited conjectured applications for role and frame semantics is automatic translation (Boas, 2002). Classical methods in machine translation were based on an analysis component that extracted a language-independent semantic representation, an *interlingua*, from which the text in the target language was produced by a generation component. When the field was swept clear by the introduction of statistical methods, interlingua-based translation fell out of favor, and word-based translation based on the noisy channel model dominated the field completely. Recently, however, statistical machine translation has moved to more complex representations than just words, and it is conceivable that frame and role semantics can be helpful in designing a future new interlingua (Boas, 2005).

However, this presupposes that the frame-semantic structures in the two languages are similar, or that suitable mappings can be constructed (or learned). And above all, practical application of frame semantics in this context is currently limited by the absence of frame ontologies, lexicons, and annotated corpora for other languages than English.

For non-frame role semantics, the Chinese PropBank has recently received much attention (Xue and Palmer, 2007; Xue, 2008). It remains to be seen whether this resource can facilitate translation; since PropBank core argument labels do not generally have a consistent interpretation across predicates, this seems unlikely (see also a related discussion by Padó (2007a), pages 40–42). Preliminary attempts to integrate semantic role labelers in syntax-based machine translation have not improved results (Liu and Gildea, 2008).

The assumption that a text and its translation have isomorphic frame-semantic structures has also made it possible to automatically annotate role-semantic structure in corpora in a new language (Padó and Lapata, 2005; Padó, 2007a; Johansson and Nugues, 2006).

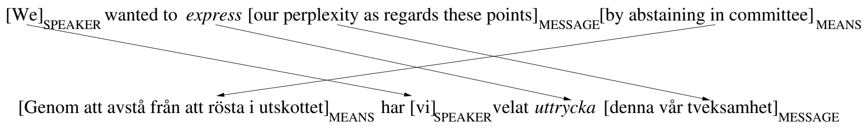

[We]SPEAKER wanted to *express* [our perplexity as regards these points]MESSAGE [by abstaining in committee]MEANS
[Genom att avstå från att rösta i utskottet]MEANS har [vi]SPEAKER velat *uttrycka* [denna vår tveksamhet]MESSAGE

Figure 2.3: *Example of isomorphic frame-semantic structure in an English–Swedish sentence pair.*

For instance, figure 2.3 shows an example of an English–Swedish sentence pair from the Europarl corpus, and its corresponding frame-semantic annotation, which in this case allows a perfect transfer. For the transfer method to be meaningful, the following assumptions must be made:

- The complete frame ontology in the English FrameNet is meaningful in Swedish as well, and each frame has the same set of semantic roles and the same relations to other frames.
- When a predicate belongs to a certain frame in English, it has a counterpart in Swedish that belongs to the same frame.
- Some of the semantic roles on the English side have counterparts with the same semantic roles on the Swedish side.

The structural similarity assumptions are invalid in many cases, depending on typological and cultural distance between languages, but also on genre – in fiction, for instance, translation is often less literal for artistic reasons. However, the assumptions seem to work fairly well in practice: Johansson and Nugues (2006) describe an experiment where automatically annotated frame-semantic data were used to train an automatic semantic role labeler for Swedish that achieved an F_1-score of 0.55 on a test corpus[4].

2.4.4 Vector Space Applications

Text categorization is the task of automatically assigning a document to one or more predefined categories. The most successful algorithms to solve this problem have been based on statistical classification methods using a feature representation of the complete document based only on the individual words in the document – the *bag-of-words* representation.

It has recently been shown (Persson, 2008; Persson, Johansson, and Nugues, 2008) that categorization accuracy can be improved by adding predicate–argument features to a bag-of-words representation. The predicate–argument structures were extracted using the system described in Chapter 7. The extensions of the feature set led to error reductions of between 2 and 10 percent for categories having more than 2,000 training documents.

This application is different from those described above, since the predicate–argument structures are used *implicitly*, as a part of a feature representation, rather than explicitly.

It remains to be seen whether this feature representation can be applied in other tasks that are commonly solved using bag-of-words representations, such as information retrieval or sentiment classification.

[4]See 5.4 for a definition of this metric

2.4.5 Applications Using Domain-Specific Role Sets

Role-semantic representation frameworks have also been created or adapted for specific applications. This is advantageous for several reasons: First, theoretical problems of definition of roles are less severe when only a small fragment of language is covered – this is similar to the view of frame semantics, where semantic roles are defined with respect to *situations*. Second, domain-specific semantic role labelers can often avoid a very significant obstacle in automatic semantic role labeling: word sense ambiguity – the sense of the predicate word determines which semantic roles are played by its arguments. A typical example of an application for which a domain-specific semantic role labeler was created is Carsim (Johansson et al., 2005), a system for automatic illustration of traffic accident news texts in Swedish.

Chapter 3

Dependency-syntactic Representations

The information communicated by humans is expressed by words arranged in a sequence. The meaning of the complete utterance is determined in part by the meaning of the words themselves, but also by the pattern in which the words are arranged. When we arrange words in a pattern, their organization is a device that we use to signal a semantic relationship between the concepts denoted by the words. Saussure (1916) famously characterized language as a system of *signs*, where a sign consists of a *signifier*, a symbol that is used to communicate a *signified*, a unit of information; in our case, we can say that the arrangement of words is a signifier referring to a semantic relationship – the signified.

Syntax is the study of how a set of surface words may be arranged in a sequence to form a complete sentence. A *syntactic representation* of a sentence is a data structure that represents how words in a sequence form a pattern. In the framework of syntactic *dependency*, the syntactic representation consists of a graph of *binary relations* between words. The concept of syntactic dependency was famously introduced into modern linguistics by Tesnière (1959):

> The sentence is an *organized whole*; its constituent parts are the *words*. Every word that functions as part of a sentence is no longer isolated as in the dictionary: the mind perceives *connections* between the word and its neighbors; the totality of these connections forms the scaffolding of

the sentence. The structural connections establish relations of *dependency* among the words. Each such connection in principle links a *superior* term and an *inferior* term. The superior term receives the name *governor*; the inferior term receives the name *dependent*.[1]

To exemplify the concept of syntactic dependency graph, Figure 3.1 shows a possible dependency-syntactic representation of the sentence *Yesterday she gave the horse an apple.*

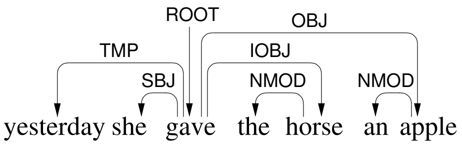


Figure 3.1: *Example of a dependency representation.*

The three main properties of dependency graphs when used as syntactic representations are thus:

- The graph consists of edges between words, and the presence of an edge between two words denotes a grammatical cooperation between those words.
- The edges are *directed*, and the direction of an edge between two words denotes which of them determines the grammatical behavior of the complete structure.
- The edges are *labeled*, and the label associated with an edge between two words denotes the nature of their grammatical relationship, the *grammatical function*.

Just as we defined semantic roles (see 2.1) as equivalence classes of relations between predicates and arguments, we define grammatical functions by noting that the nature of grammatical cooperation between governor and dependent is "similar" in many cases. This observation allows us to introduce a finite number of equivalence classes over the infinite set of syntactic relations between governor and dependent, and thus use a finite number of grammatical function labels. For instance, a finite verb is often preceded by a phrase in the nominative case

[1]Translation from French by Kuhlmann (2007).

with which it morphologically agrees in grammatical number, and we introduce the grammatical function label of *subject* (SBJ in the Figure 3.1) to denote this type of cooperation. The grammatical function is one of the primary (but not the only) means of expressing the semantic role relations that hold between a predicate and its arguments. Note that the inventory of grammatical function labels is specific to a language, unlike semantic role labels, which are defined in terms of logic.

Dependency representations are often contrasted with *constituent* representations, which are more widespread in linguistic theories (at least in those describing English). These structures instead represent the grammatical organization of a sentence by means of hierarchically organized *constituents* or *phrases*, and are classically associated with context-free grammars. Figure 3.2 shows a constituent representation of the example sentence.

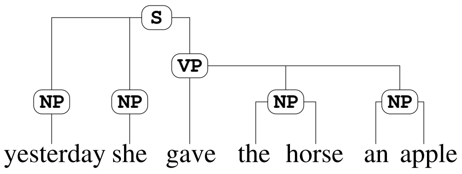


Figure 3.2: *Example of a constituent representation.*

Dependencies with labeled edges and constituents with labeled phrases can be seen as representations of two different views of the organization of a sentence, where either the nature of cooperation or the hierarchical organization is made explicit. To some extent, constituents can be derived from dependencies and vice versa. The question is then which of the two views – if any – to treat as the primitive representation from which the other is derived. In this work, we regard dependencies as primitive, and derive constituents only when necessary (in constituent-based evaluation). However, let it be emphasized that this is for proof of concept, not an ideological choice.

The chapter starts by giving an overview of research on the relation between syntax and automatic semantic role labeling. We then focus on dependency syntax: First, we give formal definitions of the concept of a dependency graph. With the formal machinery in place, we turn to the question of defining the meaning of the dependencies that constitute the graphs.

3.1 The Role of Syntax in Automatic Semantic Role Labeling

Since the beginning of research in automatic semantic analysis of natural-language text, it has been assumed that semantic analysis is closely related to syntactic analysis. This is of course very intuitive – as mentioned above, the surface organization of words is the encoding of the meaning of the complete structure.

The assumption of syntactic representation has also been pervasive in research on automatic semantic role labeling. For instance, the seminal work by Gildea and Jurafsky (2002), which introduced semantic role labeling to a general audience, carried out semantic role analysis on top of the output of a constituent parser.

In search of the simplest syntactic representation necessary for semantic role labeling, shallow[2] syntax produced by chunkers is an obvious candidate (Carreras and Màrquez, 2004). The general consensus seems to be that shallow syntax is insufficient (Gildea and Palmer, 2002; Punyakanok, Roth, and Yih, 2008). Still, Màrquez et al. (2005) showed that competitive performance can be achieved using a shallow syntactic representation based on chunk and clause bracketing. Interestingly, semantic role labelers based on shallow syntax seem to be complementary to those based on full constituents, which make them useful for building hybrid systems.

To follow this line to its logical conclusion, a recent paper by Collobert and Weston (2007) describes a system based on a neural network, which performs semantic role segmentation using no syntactic intermediate representation whatsoever, although they suggest that results could be improved by using a chunk-based input representation. However, this work is difficult to compare to previously published results, since the systems are evaluated using a word error rate rather than a standard segment-based metric (see 5.4 for a further discussion on evaluation). The article is also lacking in analysis; no explanation is given as to *why* their system is able to identify semantic role segments of arbitrary length without having access to a recursive syntactic representation. It seems that the bag-of-words feature representation that is induced by their word vectors can implicitly capture information about structure nesting that is made explicit in syntactic representations, but

[2]In this work, *shallow* syntax means a simplified constituent structure, such as that output by a chunker. In other traditions, such as in feature-based grammar frameworks, *shallow* syntax is roughly the same as feature-less surface syntax.

this is left unexplained. We find it unlikely that their method could work in a language with free word order.

Nevertheless, their success raises profound questions about the intellectual grounding of the traditional approaches to role-semantic analysis. If their results would be improved further, and proper explanations were given, the scientific method would force us to question whether syntax-driven semantic analysis, and indeed the explicit representation of syntactic structure, has any scientific validity at all.

By habit, most systems for automatic role-semantic analysis have used constituents as in the Penn Treebank (Marcus, Santorini, and Marcinkiewicz, 1993), produced by Collins' (1997) or Charniak's (2000) statistical parsers. Dependency syntax has received very little attention for the SRL task, despite a surge of interest in dependency parsing during the last few years (Buchholz and Marsi, 2006). The earliest work on dependency-based semantic role analysis was done in the context of the Prague Treebank for Czech, where automatic systems were created to assist humans in annotating the tectogrammatical (deep-syntactic or shallow-semantic) layer (Žabokrtský, 2000; Žabokrtský, Sgall, and Džeroski, 2002). For English, the literature on dependency-based SRL is scant; the first work we know is the preliminary experiment by Hacioglu (2004), which does semantic role analysis using a gold-standard dependency treebank. Another example is the experiment by Pradhan et al. (2005b), which is the first work we are aware of that used an automatic parser. Unfortunately, the results were negative: The F1 measure on the test set dropped from 79.9 to 47.2 (from 83.7 to 61.7 when using a head-based evaluation).

However, there are a number of linguistic motivations why dependency syntax could be beneficial in an SRL context, even for a constituent-friendly language like English. First, complex linguistic phenomena such as *wh*-word extraction and topicalization can be transparently represented by allowing nonprojective dependency links (see definition in 3.2). These links also justify why dependency syntax is often considered superior for free-word-order languages; it is even very questionable whether the traditional constituent-based SRL strategies are viable for such languages. Second, *grammatical function* such as subject and object is an integral concept in dependency syntax. This concept is intuitive when reasoning about the link between syntax and semantics, and it has been used earlier in semantic interpreters such as Absity (Hirst, 1983).

Although this is not generally acknowledged, traditional semantic role analyzers based on constituent syntax have already incorporated features that point toward a dependency representation. Most importantly, almost every system uses a set of rules to extract a *head* in each constituent[3]. This is considered crucial, since it is necessary for extracting lexical features. The second hallmark of dependency representations – grammatical function – is also used in constituent-based systems in the guise of features such as position and "governing category." Explicit grammatical functions have rarely been used – except from the systems for tectogrammatical analysis of Czech, the only published work we are aware of is a tentative experiment (Toutanova, Haghighi, and Manning, 2005) in which grammatical functions extracted from a gold-standard treebank were used.

As an illustration of our previous points, consider the sentence *yesterday she gave the horse an apple* and its constituent-syntactic representation, shown in Figure 3.2. We are interested in determining the semantic relations between the predicate *gave* and its noun phrase argument *the horse*. Based on the syntactic representation, a statistical semantic role labeler extracts features representing the relation between predicate and argument: For constituent-based systems, this is typically the PATH NP↑VP↓VB, the GOVERNING CATEGORY VP, and the POSITION of the argument with respect to the predicate, AFTER. However, based on these features only, we would have to assign the same semantic role label to the following noun phrase *an apple*, for which these features have the same values. So how would a state-of-the-art constituent-based semantic role labeler be able to correctly predict the semantic roles RECIPIENT of the phrase *the horse*, even though the grammatical features cannot discriminate? Probably, it can do this because it utilizes one or more of the following strategies:

- Its machine learning component may have learned to cluster words referring to animate beings (such as *horse*), and these words increase the probability of the semantic role RECIPIENT. To be able to extract this feature, the head (*horse*) needs to be determined using head-finding rules, and as argued above, this constitutes a step towards a dependency representation.
- Its machine learning component may have learnt that an NP followed by another NP should be tagged with RECIPIENT rather than THEME.
- It might use a global reranker that has learnt that RECIPIENT+THEME is preferable to RECIPIENT+RECIPIENT, or a global

[3]Modern generative syntactic theory is based on the X-bar framework, which also requires every phrase to have a head. This is not reflected in the Penn Treebank, though.

constraint system that rules out two identical core argument labels.

However, the solution could be much simpler: the NP *the horse* has a *grammatical function* of indirect object, and for the word *give*, the indirect object is the surface realization of the RECIPIENT. This does not conflict with the following NP *the apple*, which is a direct object, corresponding to the semantic role of THEME. As described previously, the grammatical function is one of the primary means of encoding semantic role information. The three features used by the constituent-based systems are implicit reflections of this property, which is made explicit in the dependency-syntactic structure in Figure 3.1.

3.2 Formal Definitions

This section introduces the formal machinery that we will make use of. We first define the concepts of token and sentence.

Definition 3.1. A *token* t is a tuple $\langle i, f_1, \ldots, f_k \rangle$, where $i \in \mathbb{N}$ is called the index and $f_1, \ldots, f_k$ are atomic features.

Definition 3.2. A *sentence* $\boldsymbol{x}$ is a set $\{t_0, \ldots, t_n\}$ of tokens whose indices are $0, \ldots, n$, respectively.

To simplify exposition, we assume that the first token t_0 in each sentence is a dummy symbol not corresponding to a word.

We are now ready to give the definition of a dependency graph.

Definition 3.3. Given a label set L and a sentence $\boldsymbol{x}$, a *labeled dependency graph* for $\boldsymbol{x}$ is a pair $\langle \boldsymbol{x}, E \rangle$, where $E \subseteq \boldsymbol{x} \times \boldsymbol{x} \times L$. We refer to a tuple $\langle g, d, l \rangle$ as a *labeled dependency edge*, and we write it as $g \xrightarrow{l} d$ or just $g \rightarrow d$. We say that g is a *governor*[4] of d with edge label l, and that d is the *dependent* of g. The transitive–reflexive closure of the governor relation is called *dominance*.[5]

Definition 3.4. For a labeled dependency graph G for the sentence $\boldsymbol{x}$ and label set L, we say that G is *well-formed* if it satisfies the following two constraints:

[4]Following Mel′čuk (2003), we use the term *governor* rather than *head*. We reserve the term *head* for the head word of a constituent or subtree to avoid confusion.

[5]The transitive–reflexive closure R^* of a relation R is defined as follows: $\forall x : xR^*x$, $\forall x, y, z : xR^*y \wedge yRz \Rightarrow xR^*z$. The definition of dominance used here is slightly different from its usual definition in graph theory: x dominates y if every path from the root to y contains x. These definitions are of course equivalent if the graph is a tree.

ROOTEDNESS. The first token t_0 in $\boldsymbol{x}$ is a *root* in G, i.e. there is no token g in $\boldsymbol{x}$ such that $g \rightarrow t_0$.

CONNECTEDNESS. The root t_0 dominates every token in $\boldsymbol{x}$.

In addition, all dependency graphs that we will use in this work satisfy the following constraint.

UNIQUE GOVERNOR. For every token t in $\boldsymbol{x}$, there is at most one governor.

In other words, the dependency graph is a tree. There are syntactic theories that do not impose this constraint, for instance the *discontinuous grammar* (Buch-Kromann, 2006) used to annotate the Danish Dependency Treebank (Trautner Kromann, 2003), and this results in a more expressive syntactic framework that allows convenient analyses of some tricky phenomena that we will discuss in Chapter 4, where the hierarchical organization of a structure is unclear. However, parsing with such graphs is difficult – see the discussion in 7.2 for details.

We finally introduce the *projectivity* property of dependency graphs (Lecerf, 1960).

Definition 3.5. A dependency edge $g \rightarrow d$ in a dependency graph G for a sentence $\boldsymbol{x}$ is called *projective* if all tokens in $\boldsymbol{x}$ between g and d are dominated by g. The graph G is called *projective* if all its edges are projective.

Visually, this means that the dependency graph can be drawn above the sentence without reordering the words, so that no edges cross.[6]

This property is important, since it means that embedded structures are continuous and can be processed recursively, and this enables parsing algorithms to use stacks or dynamic programming. The connection with context-free grammars is also apparent. It seems that the majority of sentences in all human languages have projective dependency-syntactic analyses. Especially in some of the world's most widely spoken languages, such as Chinese, English, and Japanese, the syntactic structures are almost exclusively projective. In some other languages, nonprojective structures are more common, but still very close to projective structures (Kuhlmann and Nivre, 2006).

[6]We make no distinction in this work between projectivity and *planarity*, since we assume that every sentence starts with a dummy root token.

Constituent Trees

Although the work described in this dissertation is mainly based on dependency syntax, we also need to define the notion of constituent for completeness. We also introduce some auxiliary notation that will be employed when we describe transformations on constituent trees in Chapter 4.

Definition 3.6. Given a label set L and a sentence $\boldsymbol{x}$, a *constituent* is either a single token from $\boldsymbol{x}$, which is then referred to as a *terminal node*, or a *nonterminal node*: a tuple $c = \langle l, C \rangle$, where $l \in L$ and C is a set of constituents. If $d \in C$, we say that c is a *parent* of d, or that d is a *child* of c. The transitive–reflexive closure of the parent relation is called *dominance*. For a nonterminal constituent c, we will sometimes use the shorthand $|c|$ for its number of children, i.e. $|C|$.

Definition 3.7. Given a label set L and a sentence $\boldsymbol{x}$, a *constituent tree* for $\boldsymbol{x}$ is a pair $\langle \boldsymbol{x}, T \rangle$, where T is a set of constituents for which the following properties hold:

ROOTEDNESS. There is exactly one constituent $r \in T$, referred to as the *top node* or *root*, which is the child of no constituent in T.

CONNECTEDNESS. Every constituent in T dominates at least one token in x, and the top node dominates every token.

UNIQUE PARENT. Every constituent in T except the top node is the child of exactly one other constituent in T.

This definition allows discontinuous constituents as in the TIGER treebank (Brants et al., 2002). However, most constituent treebanks, such as the Penn Treebank, differ in this respect, and use other means to encode discontinuous structures.

Definition 3.8. The *start-token index* $\mathrm{sti}(c)$ of a constituent c is defined as follows:

$$\begin{array}{ll} \mathrm{sti}(c) = i & \text{if } c \text{ is a terminal node with index } i \\ \mathrm{sti}(c) = \max_{d \in C} \mathrm{sti}(d) & \text{if } c \text{ is a nonterminal } \langle l, C \rangle. \end{array}$$

We also define the *start-token order* $\preceq_{\mathrm{st}}$ so that $c \preceq_{\mathrm{st}} d$ if $\mathrm{sti}(c) \leq \mathrm{sti}(d)$. For a nonterminal node c, we will use c_i to refer to the i-th child of c in start-token order.

3.3 Dependency Graphs as Syntactic Representations

With the formal framework in place, we now need to define denotations of its concepts. In this section, we will outline what information should be represented in the graphs. We will define criteria for establishing that a direct dependency holds between two words, and for determining its direction and label.

The word–word relations that we will encode in dependency graphs are *surface-syntactic*, and the criteria that we will use to construct representations are primarily based on observable linguistic phenomena, such as word order and morphology, rather than on underlying semantic structures.

Our surface-syntactic methodology makes it natural to impose the following restrictions on the syntactic structures:

- *Monostratal*: We use only a single layer of syntactic information, rather than multiple layers as for instance in Meaning–Text Theory (Mel'čuk, 1988) or Extensible Dependency Grammar (Debusmann, Duchier, and Kruijff, 2004).

- *Single governor*: As mentioned in 3.2, we allow only one governor per token, thereby imposing a tree structure on the dependency graph. This contrasts with frameworks such as Word Grammar (Hudson, 1984) and Discontinuous Grammar (Buch-Kromann, 2006).

- *Surface tokens*: Except the dummy root token, every node in a syntactic tree corresponds to a single surface word.

Our reasons for these restrictions are *parsimony* – we want to find the simplest possible representation that allows accurate semantic analysis; *efficiency* – multiple layers, multiple governors, multi-word nodes, and insertion of empty nodes all make automatic syntactic analysis more complex; *reliability* – features for statistical systems can be expected to be more reliable if based on observable surface phenomena; and *scientific rigor* – empty nodes and additional "deep-syntactic" layers are theoretically questionable and hard to define stringently. As pointed out by Schubert (1987), if we allow more complex representations, "one is too easily tempted to make descriptions not about the words found, but about abstract structures. This entails the danger of reasoning about 'underlying' or 'kernel' sentences, 'implicit' predications or the like."

We thus take the word as the minimal unit in syntactic structure. It could well be argued that the minimal unit should not be the word but the morpheme. This is certainly true in agglutinative languages such as Turkish or Finnish, but also in some cases in English – consider, for instance, the structure of *pro- and anti-government demonstrations*. Even when we consider the word as the minimal unit, we must be careful to define what constitutes a word, although this consideration plays a minimal practical role in this work since we used pretokenized data[7]. Multi-word tokens were used by Tesnière (1959), and are also common in analyses of Japanese syntax, where the *bunsetsu* (a phrase followed by a case-marking postposition) can be regarded as the minimal unit in syntax.

With regards to the structure of the dependency graphs, we will *not* impose a projectivity restriction, unlike some other descriptions. Although English dependency graphs – like in other languages – are most often projective, there are situations where this does not hold, often due to topicalization and "heavy shifts."

3.3.1 A System of Criteria for Establishing Syntactic Dependencies

As stated previously, the presence of a link between two words means that they cooperate in forming a complete grammatical structure. The direction of the link denotes which of the words is regarded as the one that determines the grammatical behavior of the complete structure. We now turn to the task of defining criteria for establishing these properties. This section describes a typical system of such criteria that was originally described by Mel'čuk (2003). It should be noted that the criteria given here should be viewed as a set of general guidelines, not an algorithm. It is an open and interesting question to what extent they could serve as a basis for algorithms for unsupervised dependency annotation.

Connectedness Criteria

The first criteria, the *connectedness criteria*, which formalize the notion of "cooperating" that we have referred to, describe necessary conditions

[7]In the Penn Treebank conventions, contractions such as *cannot* and *I'm* are split, as well as possessive constructions such as *Alexander's*. In the CoNLL-2008 data, most hyphenated words, like *part-of-speech*, were also split.

for a dependency relation to hold between two words. They say nothing about the direction or label of the relation.

A1 *Linear arrangement of words.* The words w_1 and w_2 considered in a communicatively neutral sentence can have a direct syntactic dependency link between them only if the linear position in the sentence of one of them must be specified with respect to the other.

A2 *Potential prosodic unity.* There can be a direct syntactic link between the words w_1 and w_2 only if

- either w_1 and w_2 can form an utterance, i.e. a special prosodic unit – a phrase,
- or w_1, w_2, and some set of words W form a phrase of which w_1 is the head, and w_2 and W also form a phrase, of which w_2 is the head.

There is some vagueness in these criteria. In A1, for instance, it must be determined what is or what is not a "communicatively neutral sentence." Criterion A2 is even more problematic, since it refers to the undefined concept of "prosodic unit." An expression of the idea behind the two criteria that would be more empirically testable is whether the "unit" (w_1, w_2, and possibly some other words to form a complete structure) is able to appear in more than one linguistic context.

Direction Criteria

The second set of criteria is used to establish the *direction* of the dependency link. These criteria are ordered hierarchically: For instance, if B1 clearly applies, then we do not take B2 or B3 into account.

B1 *Attachment behavior.* In the phrase $w_1 - w_2$, w_1 is the syntactic governor of w_2 if the attachment behavior – the *passive syntactic valency* – of the whole phrase is determined by w_1 to a greater extent than by w_2.

B2 *Morphological contact point.* In the phrase $w_1 - w_2$, where B1 does not apply, it is w_1 that is the syntactic governor of w_2 if w_1 controls the inflection of words external to the phrase or w_1's inflection is controlled by such words.

B3 *Semantic links.* In the phrase $w_1 - w_2$, where B1 and B2 do not apply, w_1 is the syntactic governor of w_2 if there is a semantic dependency $w_1 \to w_2$.

Here, B1 and B2 are the formalization of what was previously written about the governor being the main cause of the grammatical behavior of the complete structure, and B3 used to break remaining ties.

Criteria for Grammatical Function Label Inventory

The final set of criteria states necessary conditions for the inventory of grammatical function labels to be well-formed. It should be noted that the definition of grammatical functions is intertwined with the definition of parts of speech (Schubert, 1987). However, we will take the parts of speech for granted in this work.

C1 *Absence of semantic contrast.* A syntactic relation r must not describe two different phrases $w_i \xrightarrow{r} w_j$ and $w_m \xrightarrow{r} w_n$ where

- w_i and w_m are forms of the same lexeme, and w_j and w_n are forms of the same lexeme,
- the two phrases contrast semantically,
- the phrases differ formally only by some syntactic means of expression (i.e. by word order, by syntactic prosody, or by syntactic grammemes).

C2 *Substitutability with prototype.* For every syntactic relation r, there must exist a *prototype category* X other than substitute pronouns such that for any syntactic configuration $h \xrightarrow{r} \Delta_Y$, replacing Δ_Y by Δ_X, where Δ_X is headed by a word of the prototype category X, does not affect its syntactic wellformedness.

C3 *Repeatability.* Any syntactic relation must be either unlimitedly repeatable or non-repeatable.

Criterion C3 is related to the distinction between complements and adjuncts – every complement function is supposed to appear only once under a governor, while adjuncts may appear more than once. While it is hard in practice to consistently distinguish complements and adjuncts, this criterion forces us to define complement functions carefully if introduced.

It can be observed that trivially large sets of function labels (i.e. when every label encodes the complete context) always pass these tests.

The set of functions should be small enough to allow us to make general observations, but the criteria force us to make the set large enough to make explicit the distinctions in organizational patterns used in language to express semantic distinctions. We can thus view the search for the function inventory as a constrained optimization problem where we look for a minimal set satisfying the criteria.

Additional Connectedness Criteria

We will encounter a number of situations where Mel'čuk's connectedness criteria (A1 and A2) are not enough to determine the structure, and we may end up with strongly connected clusters of three or more tokens. In these ambiguous situations, we may have use of additional rules of thumb. We first introduce two definitions:

Definition 3.9. The words w_1 and w_2 are said to be *lexically dependent* if the lexical form of w_2 is selected by w_1.

Definition 3.10. The words w_1 and w_2 are said to be *morphologically dependent* if the morphological inflection of w_2 depends on the lexical form or some feature of w_1.

A typical example of lexical dependence is the selection of a preposition by a verb, such as *depend <u>on</u>* or *believe <u>in</u>*. Morphological dependence is common, such as number agreement between a verb and its subject, and gender agreement between a pronoun and its referent. It was the main tool for determining syntactic dependency in Tesnière's work (1959), although this is not explicitly stated (Schubert, 1987).

Unlike syntactic dependency (according to our framework), lexical and morphological relations are often bidirectional. A well-known example is the morphological relation between a verb and its subject: The number feature of the verb is controlled by the subject noun phrase, and the case of the noun phrase is set to nominative to reflect its grammatical function with respect to the verb. When establishing syntactic dependencies, we will make use of morphological dependence that is manifested in features that reflect syntactic properties. Examples of such morphological features are case in nouns and mood in verbs, both of which can be imposed by a grammatical function with respect to a governor.

We can now formulate the additional connectedness criteria.

α**1** *Morphological dependence.* If the connectedness criteria A1 and A2 allow dependency links between all three words w_1, w_2, and

w_3, and there is a morphological dependency between w_1 and w_2 in the form of government of features that reflect syntactic properties, then the $w_1 - w_2$ link should take priority.

α2 *Lexical dependence*. If the connectedness criteria A1, A2, and $\alpha 1$ allow dependency links between all three words w_1, w_2, and w_3, and there is a lexical dependency between w_1 and w_2, then the $w_1 - w_2$ link should take priority.

Chapter 4

Automatic Construction of an English Dependency Treebank

In Chapter 3, we argued that dependency-syntactic representations have a potential to be useful for automatic semantic analysis. To support these claims, we have to train a syntactic parser on a dependency treebank. However, except for preliminary efforts (Rambow et al., 2002), there exists no dependency-annotated treebank of English. Also, we can rule out manual corpus annotation since it is very costly and time-consuming. We thus have to resort to automatic conversion from an existing constituent-based resource, the Penn Treebank (Marcus, Santorini, and Marcinkiewicz, 1993).

This chapter describes the algorithms that carry out the constituent-to-conversion algorithms. In addition, we apply the principles laid down in Chapter 3 to disentangle some nontrivial linguistic considerations. We give dependency-syntactic analyses of a number of constructions in English.

Relying on an external resource of course limits our options when annotating dependencies, since the only structures that can be derived are those that can be deterministically transformed from this source. Fortunately, the Penn Treebank is richly structured. However, there are some limitations: For instance, we cannot introduce a distinction between complements and adjuncts except in verb phrases, since this distinction is not generally made in the Treebank.

The algorithms described in this chapter were used to create the dependency treebanks used in the experiments in this dissertation. Note, however, that for historical reasons there are two treebanks: first, the treebank used in Chapter 6 and the FrameNet-related part of Chapter 5; secondly, the treebank that was publicly distributed for the CoNLL-2008 Shared Task (Surdeanu et al., 2008), used in the Chapter 7 and the PropBank-related part of Chapter 5. When we need to distinguisth the two treebanks, we will refer to them as the LTH and CoNLL-2008 treebanks, respectively. They differ only slightly, primarily in how underspecified noun phrase structure is resolved: The LTH treebank used the noun phrase bracketing by Vadas and Curran (2007), while the CoNLL-2008 treebank imported noun phrase structure from GLARF (Meyers et al., 2001), an annotation framework developed in the NomBank project (Meyers et al., 2004).

4.1 The Penn Treebank

The Penn Treebank (Marcus, Santorini, and Marcinkiewicz, 1993) is the most widely used syntactic resource for English. The core part of the representation is a constituent structure, as defined in 3.2. Discontinuous constituents are not allowed, or even expressible, since the annotation format uses nested bracketing to encode the constituent structure. The constituent structure has been extended in three ways:

- *Empty categories*, i.e. terminal nodes that do not correspond to surface words.
- *Secondary edges* that encode a number of nonlocal relations. Except for edges used in gapping, every edge is connected to an empty category.
- *Secondary labels* (or "dash tags" in Penn Treebank jargon) that represent functional or structural properties that are not part of the constituent structure. Most secondary labels, but not all, are grammatical function labels.

The extensions are not present in the output of the popular constituent parsers. However, there are systems that insert empty categories and secondary edges (Johnson, 2002; Schmid, 2006) and secondary labels (Blaheta, 2004; Merlo and Musillo, 2005).

Figure 4.1 shows a tree from the Penn Treebank. It contains three empty categories: an empty subordinating conjunction 0, and two trace nodes (*T*). The traces encode underlying "movements" that are supposed to have taken place and reflect the theoretical roots of the Penn Treebank in the transformational paradigm. There are secondary edges from the traces to the respective "moved" constituents, and four secondary labels representing grammatical functions: two SBJ, a CLR and a PRP.

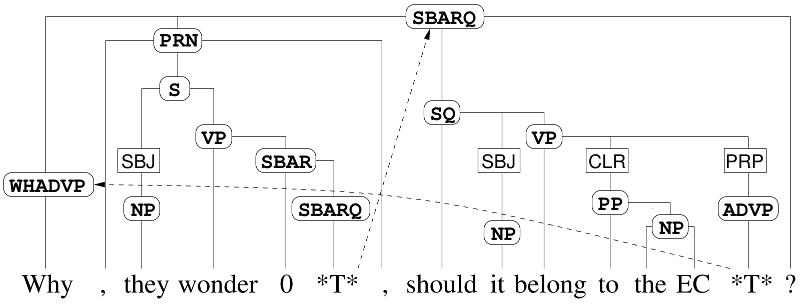


Figure 4.1: *A constituent tree from the Penn Treebank.*

Apart from its use in dependency syntax, conversion from the Penn Treebank has been used for a number of other syntactic formalisms. Hockenmaier and Steedman (2007) devised a conversion algorithm similar to ours to automatically create a treebank of derivations in the combinatory category grammar framework. Similarly, Miyao and Tsuji (2008) and Cahill et al. (2008) applied rule-based methods to convert Treebank trees into HPSG and LFG feature structure trees, respectively.

In addition to the constituents in the treebank itself, we used the noun phrase bracketing by Vadas and Curran (2007). This had to be done because a large number of noun phrases with a complex internal structure are annotated using a completely flat structure in the Penn Treebank. An extreme example is the noun phrase *other small apparel makers, button suppliers, trucking firms and fabric houses*. The main reasons for this are probably practical; it saves annotation time, and the internal structure may not be entirely clear to the manual annotators unless they are domain experts. However, the flat structure is very unappealing when the phrase is converted to a dependency structure, since this would make all words in the noun phrase direct dependents of the head word.

4.2 Dependency Analysis of Some Constructions in English

This section provides linguistic motivations for the constituent-to-conversion algorithms to be described in 4.3. We apply the syntactic framework described in 3.3 to a number of constructions in English grammar. We will restrict the attention to a selected set of problematic constructions; in most other cases, it is very obvious how to create the dependency structures from the constituents. For instance, we won't spend any effort on explaining why a noun should be selected as the head of a noun phrase[1].

While the underlying philosophy of dependency syntax, as opposed to other frameworks, is to emphasize *function* rather than *structure*, it is by no means obvious that the syntactic relationships between words in a sentence can be satisfactorily represented using only *binary* relations. In fact, problematic situations often arise when there is a configuration of words that is naturally described using an n-ary relation rather than a tree of binary relations. In the most simple of these cases, two semantically prominent "content" words are connected via a functional word that has a role in expressing the syntactic or semantic relation between them. Examples of this situation include verb–preposition–noun and conjunct–conjunction–conjunct.

Below, we will describe a range of such linguistic phenomena, and discuss how to impose a decomposition into binary relations. Our method is based on examining and generalizing from one or a few prototypical examples.

4.2.1 Prepositions

To illustrate how to analyze how prepositions should be handled, consider the fragment *depends on water*. Here, the connectedness criteria (A1 and A2) allow dependency links between all three words: both *on* and *water* must be placed after the verb *depends*, and *on* must be placed before *water*.

[1]Even this supposedly simple example could be contested: the Danish Dependency Treebank, for instance, regards the determiner as the head in a noun phrase. This may also have some support in unsupervised induction of dependency structures (Klein and Manning, 2004). The discussion of course comes down to your conception of what types of relations the dependency structures should encode.

As mentioned above, this is a situation where have a ternary relation: a relation between *depends* and *water*, and a syntactic relation marker *on*.

Intuitively, the finite verb should be the root of the structure, and it is also clearly confirmed by dominance criterion B1 that the finite verb *depends* dominates both *on* and *water*, regardless of how other links are configured.

This leaves us with three alternatives, shown in Figure 4.2. Which analysis to choose is controversial, and all three variants have been proposed in literature. We prefer to eliminate the first alternative (both

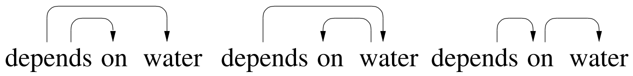


Figure 4.2: *Possible dependency representations of* depends on water*.*

on and *water* as direct dependents) because we feel that *on* and *water* are strongly linked and could appear as a single unit. This is of course also paralleled by the concept of *prepositional phrase* in constituent frameworks.

Of the two remaining analyses, we prefer the final one because criterion B1 favors an analysis with the preposition as the governor – it is the preposition *on* that is the primary explanation of where the phrase *on water* can be attached. Alternatively, we could have invoked our additional criterion $\alpha2$, since the preposition *on* is selected by *depend*. Another argument for this analysis is that English allows a phenomenon referred to as *preposition stranding*, such as in *the water we depend on*, where the prepositional complement is extracted. In these situations, the structure is simpler if the verb and preposition are linked, because only one long-distance link is needed.

This analysis is contested by frameworks that place more emphasis on isomorphism between syntactic and semantic structure: The semantically prominent words are *depend* and *water*, and *on* is only a marker of the relation between them (Johansson and Nugues, 2007a).

4.2.2 Subordinating Conjunctions, Infinitive Markers

The relations between the verb in a main clause, a subordinating conjunction, and a subordinated clause can be analyzed similarly to how we treated prepositional phrases above. However, we have to accept that structures that are only superficially different, such as *think that it's enough* and *think it's enough*, have different syntactic analyses since we use no empty categories in our framework. Similarly, the infinitive marker *to* is analyzed as the head of an infinitival clause.

4.2.3 Relative Pronouns and Question Words

A trickier case is how to handle relative pronouns, such as *which* in the fragment *water which I drink*. The question is how to determine the function of the relative pronoun *which*: like a subordinating conjunction, it is the introducer of a subordinated clause, but it also fills the object valency slot of the verb *drink*. It can thus be argued to play a dual syntactic role, and this is actually the analysis in some frameworks. The double function has also led some, such as Tesnière (1959), to propose to split relative words into two tokens *wh-ich*. The first part *wh-* (or *qu-* in Tesnière's case) then serves as a subordinator, and the second part *-ich* as the valency filler of the verb. In our case, we have to assign a single function, since we assume monostratal, single-governor dependency structures, and as argued previously, we won't modify the Penn tokenization.

We will analyze a more complex fragment, taken from the Treebank, to answer this question.

Example 4.1. goals for which he had to settle

We will first make two assumptions that we hope the reader accepts unquestioningly: first, that *goals* dominates all other words in the fragments; secondly, that *he had to settle* is conventionally analyzed.

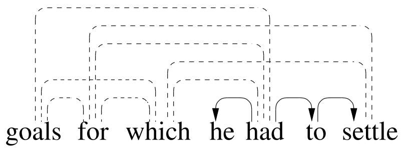


Figure 4.3: *Possible dependencies in a relative clause.*

Apart from this, Criteria A1 and A2 allow many possible analyses. Figure 4.3 shows the possible links using dashed lines. To disentangle this structure, we then proceed to identify the lexical relations, In this case, we have two of these: Most obviously, *for* is selected by *settle*. Also, *which* is selected by the preposition *for* – to see this, try to substitute the word *that*. This is also consistent with the notion of a `WHPP` phrase, as used in the Treebank.

If we accept these lexical relations, then Criterion $\alpha 2$ disambiguates the structure. The word *settle* already has a governor *to*, so it must be the governor of *for*. In consequence, *which* is subordinated to *for*. Finally, the only place where an incoming edge can be connected into the relative clause is on the word *had*. Figure 4.4 shows the disambiguated structure.

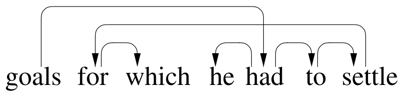


Figure 4.4: *Disambiguated structure of a relative clause.*

We thus generalize from this example to the following general principle: The subordinating nature of the relative pronoun is *not* expressed, but instead its function as a valency filler.

The same reasoning holds for *wh*-questions, which are similar in structure to relative clauses. One could argue that the head of the structure is the question word, since it is the primary determinant of the function of the complete structure. However, since the question word also fills a valency slot of the verb, we mark it as a dependent.

4.2.4 Coordination

The problems with representing non-binary relations appear once again with the phenomenon of coordination, which has been discussed to great lengths in literature. To illustrate, consider Example 4.2. Here, two conjuncts (*olive* and *orange*) are joined by a conjunction (*and*).

Example 4.2. Alexander ate an olive and an orange.

It is not trivial to represent the semantics of the sentence, but one possible analysis is the following:

$$\begin{aligned}&eat(e_1) \land \textsc{Agent}(e_1, \text{Alexander}) \land \textsc{Patient}(e_1, \text{olive}) \land \\ &eat(e_2) \land \textsc{Agent}(e_2, \text{Alexander}) \land \textsc{Patient}(e_2, \text{orange})\end{aligned}$$

The coordination of two noun phrases can thus be viewed as a shorthand for the logical coordination of two full event structures. Inspired by the semantics of the sentence, some accounts of coordination treat it as a process of deletion: *Alexander ate an olive and ~~Alexander ate~~ an orange*.

However, an explicit representation of the deletion is of course totally unacceptable in a surface-syntactic framework, since it would lead to a proliferation of empty categories. Also, the semantics may not always be as clear as in this example.

Applying the connectedness criteria, we see again that we have a strongly connected cluster: The criteria allow dependency links between all four words *ate, olive, and*, and *orange*. Let us postulate that the finite verb *ate* is the head of the complete structure – the opposite position, in which *and* is the root, could only be supported if we take the view that coordination involves deletion. Also, we find it plausible that the coordinated structure *an olive and an orange* should be regarded as a single unit and that there should only be a single dependency link leaving this unit. This analysis makes it easier to adhere to Criterion C3 for coordinated complements. For instance, if we would regard both *olive* and *orange* as direct objects of *ate*, Criterion C3 would force us either to label the two dependencies differently, which would defeat the purpose of this analysis, or to regard objects as adjuncts.

To find the head of the coordinated fragment *an olive and an orange*, Criterion B1 forces one of the conjuncts to be the head – the complete structure has the same syntactic behavior as the conjuncts would have separately[2].

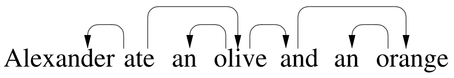


Figure 4.5: *Representation of coordination.*

Regarding the place of the conjunction in the dependency structure, we can apply a very similar argument as with prepositions to introduce a *coordination phrase*[3] consisting of the conjunction and the second conjunct, and as in the prepositional phrase, the marker (i.e. the conjunction) becomes the head. Although most accounts of coordination in the linguistic literature seem to assume symmetry, asymmetrical

[2]There are of course many situations where the situation is not so simple. For instance, the Penn Treebank uses a special phrase label `UCP` for coordinated structures of disparate elements.

[3]This is not to be confused with `CONJP` in the Penn Treebank, which just denotes a multiword conjunction such as *as well as*.

analyses using coordination phrases exist as well (Johannessen, 1998)[4]. The analysis of the sentence is shown in Figure 4.5.

The following two examples illustrate the coordination phrase. Example 4.3 shows how it appears on its own in a fragment, and Example 4.4 how it can be dislocated in a heavy noun phrase.

Example 4.3. Will Alexander come? Yes, *and Barbara* too.

Example 4.4. Coffee was brought *and a platter of pale, sharp cheeses*.

Although the examples discussed above all involved coordinated nouns, we apply the same analysis of coordination of words of other types. It could be possible to use multiple levels of coordinations, as is done in some treebanks. For instance, the Talbanken treebank of Swedish (Einarsson, 1976) uses three levels: sentence, clause, and phrase coordination.

Our asymmetrical analysis of coordination implies that the syntactic dependency representation of a coordinated structure is not isomorphic to its semantic counterpart. More seriously, in a framework without empty categories or multiple governors, this analysis results in interpretational ambiguity, exemplified in Figures 4.6 and 4.7: The dependency graphs do not tell us whether *old* applies to *men and women* or just to *men*, and similarly we cannot tell whether *it* is an object of *heard and saw* or only of *heard*. This has made a number of dependency-based treebanks such as the Prague Treebank (Hajič, 1998) adopt a symmetric analysis where the two conjuncts are treated as dependents of the conjunction.

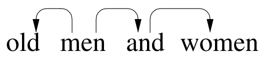


Figure 4.6: *Dependency representation of* Old men and women.

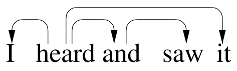


Figure 4.7: *Dependency representation of* I heard and saw it.

[4]Johannessen (1998) assumes an X-bar analysis in which the conjunction is the head, the first conjunct the specifier, and the second conjunct the complement. With her analysis, we would thus still end up with a symmetrical dependency structure.

It may be argued that this ambiguity is not of a *syntactic* nature but rather semantic/pragmatic. For instance, the semantics of the sentence in Figure 4.6 cannot be disambiguated even by humans without knowing the context, but we still see clearly that the fragment is grammatical, and we could say that this implies that the syntactic representation should represent only that. However, the sentence in Figure 4.7 is not ambiguous to humans, because *it* serves as an obligatory object for both *heard* and *saw*. The suggestion that underspecified representation should be preferred is thus compatible with modification (such as *old* in Figure 4.6) but not with complementation (as *it* in Figure 4.7), which is *syntactically* required to form a complete structure.

Mel'čuk (1988; 2003) and Hudson (1984) both tried to solve ambiguity problems arising from coordination by extending the dependency framework with simple phrase-like structures. Other solutions may be to allow empty categories or multiple heads. A solution that does not require a modification of the dependency framework is to encode attachment information in the edges, and this is indeed the solution that comes to mind when considering Criterion C1. Special edge labels may be introduced to distinguish attachment to the first conjunct from attachment to the complete structure. However, this solution perverts the meaning of the edge labels; a new representational dimension is introduced which is orthogonal to the concept of grammatical function.

We have opted to use the asymmetric analysis despite the ambiguity problems, not only because this is what is imposed by the surface-syntactic criteria, but also because it has been shown empirically that this improves parsing performance for a number of languages (Nilsson, Nivre, and Hall, 2006).

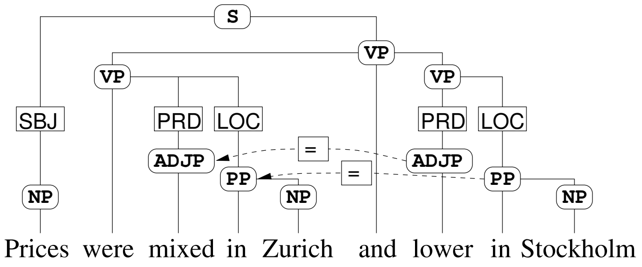


Figure 4.8: *Penn Treebank representation of a sentence with gapping.*

Another issue with coordinated structures, which is problematic whether you use a symmetric or asymmetric analysis, is that coordination often results in repeated parts being left out (ellipsed). This is referred to as *gapping*. Figure 4.8 shows how the sentence *Prices were mixed in Zurich and lower in Stockholm* is represented in the Penn Treebank. Here, two verb phrases are coordinated, and the word *were* is ellipsed from the second one. Structure parallelism between the parts in the verb phrases is represented using secondary edges (=).

This phenomenon, which is fortunately infrequent, is difficult to represent satisfactorily in a framework without empty categories. In our representation, shown in Figure 4.9, we form a coordination phrase that includes the conjunction and the "hanging" parts of the second verb phrase. In addition, we add a special structural label GAP to indicate that they should be interpreted as dependents of an ellipsed governor.

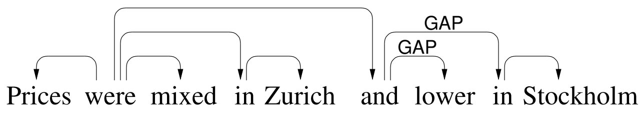


Figure 4.9: *Dependency representation of a sentence with gapping.*

4.2.5 Small Clauses

One peculiar and interesting family of constructions for which syntactic analysis is nontrivial is those involving a so-called *small clause*[5], that is a semantic predicate–argument structure that has a surface realization without a tense. An example of this phenomenon is Example 4.5, where the sentence in (a) may be reformulated using a small clause in (b).

Example 4.5.
(a) They believed that he lied.
(b) They believed him to lie.

The semantics of both (a) and (b) can be analyzed as follows in a PropBank representation:

$$\text{believe.01}(e_1) \wedge \textsc{Arg0}(e_1, \text{they}) \wedge \textsc{Arg1}(e_1, e_2) \wedge \text{lie.02}(e_2) \wedge \textsc{Arg0}(e_2, \text{he})$$

[5]The terminology used for these constructions and their various subtypes varies considerably between syntactic theories. In this work, a *small clause* is any tenseless subordinated clause, defined in practice as an unlabeled S node directly under a VP node. This is also the terminology used in the Treebank (Bies et al., 1995, chapter 15).

There are two opposing ideas of how to represent the syntactic structure of (b): In the first, the small-clause analysis, syntactic structure is supposed to be isomorphic to the semantic structure and thus similar to that of (a), meaning that *him* is analyzed as the syntactic subject of *lie*. As a consequence, it must be explained why the subject does not appear in nominative case. In Chomskian schools, this is referred to as *exceptional case marking* (ECM). This view is also held by Latin-inspired grammarians such as Tesnière (1959), who argue that *him to lie* is a nominalization of *he lied*, and that the case of *him* is a consequence of the grammatical function of the whole clause. The second analysis argues that although *him* is the logical subject of *lie*, its case indicates that it has been *raised* to the position of grammatical object rather than subject. This position has most notably been advocated by Postal (1974) but also in surface-oriented dependency syntax (Mel'čuk, 2003).

It is difficult to analyze this construction using the connectedness criteria, since the analysis depends on our intuition: with a small-clause (Latin-inspired) outlook, we could say that *him to lie* is a prosodic unit (A2) and that the case of *him* indicates a morphological relation with the word *to* (α1). Conversely, in the raising-to-object view, *him to lie* is *not* a prosodic unit and the morphological relation instead holds between *him* and *believe*.

Although we see the point of the small-clause analysis in some other contexts than this,we settled for the raising-to-object analysis without much hesitation. The reason for this is the instablility of the purported syntactic unit *him to lie* – it easily breaks apart due to passivization or topicalization, for instance. The case of *him* is also clearly governed by its syntactic relation to *believe*, since it would be nominative if *believe* were passive. The logical link between *him* and *believe* could well be represented in a multistratal representation, but it should not be regarded as a surface-syntactic relation.

By doing this analysis, we diverge from the convention in the Penn Treebank (Bies et al., 1995, chapter 15), which brackets the small clause as a syntactic unit, and thus stays closer to semantics rather than to surface syntax. This style of bracketing is used very liberally in the Treebank. PropBank, on the other hand, takes a middle position by splitting many of the small clauses in the Treebank, but not all (Babko-Malaya et al., 2006).

In addition to raising of grammatical subjects to object position, as we saw above, we may also see raising to *subject* position. This happens for verbs such as *seem*, for instance in *he seemed to lie*. The analysis of these constructions is less controversial – it would be hard to argue in a surface-syntactic framework that *he* is the grammatical subject of *lie* rather than of *seemed*.

A superficially similar construction is *control*, in which the verb also takes its subject or object from a subordinate clause, and which does have a semantic relation to that subject or object. Subject control verbs include *try*, and object control verbs include *force*, *persuade*. It may be difficult to distinguish raising and control, and the Treebank only marks control (of objects) for a small selected set of verbs, and treats the rest as verbs taking small clause complements.

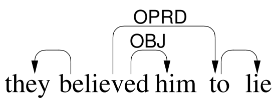


Figure 4.10: *Dependency representation of a sentence with gapping.*

Figure 4.10 shows the dependency-syntactic analysis of the sentence *they believed him to lie*. As argued above, we treat *him* as a syntactic object of *believed*. Regarding function labeling, we labeled the predicative part of all small clauses using the label `OPRD` (object predicative). We are aware that this terminology is correct for only a subset of the constructions for which the Treebank employs a small-clause analysis. Also, the label does probably not adhere to Criteria C1 and C2, but we did not have time to carefully categorize the various types of constructions or implement a procedure to distinguish them.

4.3 Automatic Conversion of Constituents to Dependencies

As argued previously, constituents and dependencies can be seen as two different *views* of the grammatical organization of the sentence. This implies that it should in principle be possible to convert one representation into the other.

The main effort when moving between representations is to make explicit what is implicit in the original representation. In a constituent representation, the recursive groupings are explicit but grammatical function and governor–dependent hierarchy is underspecified. Since dependency syntax represents grammatical structure by means of labeled binary governor–dependent relations rather than phrases, the task of the conversion procedure in our case is to identify and label governor–dependent pairs. The idea underpinning constituent-to-dependency conversion algorithms (Magerman, 1994; Collins, 1999; Yamada and Matsumoto, 2003) is that governor–dependent pairs are created from constituents by first selecting a *head token* in every constituent and then adding dependency links in which the head word is the governor. The dependency labels are subsequently inferred from the phrase–subphrase or phrase–word relations. Figure 4.11 shows an example of the general idea: In a noun phrase (left), our algorithm selects a head token (*sea*, illustrated by the dotted arrow) and adds dependency links (right) to the other words in the constituent. The dependency labels on all three links are NMOD (modifier of nominal) since the corresponding words are all subordinated elements in a noun phrase.

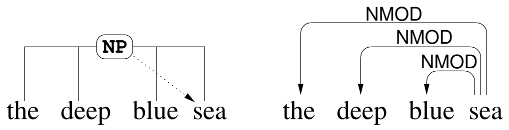


Figure 4.11: *Conversion of a noun phrase to a dependency tree.*

Algorithm 4.1 Overview of the constituent-to-dependency conversion algorithm.

```
function CONSTITUENTS-TO-DEPENDENCIES($T$)
input Constituent tree $T$
  PREPROCESS($T$)
  ASSIGN-HEADS($T$)
  ASSIGN-FUNCTIONS($T$)
  return CREATE-DEPENDENCY-TREE($T$)
```

Algorithm 4.1 shows the main steps in the constituent-to-dependency conversion algorithm for a constituent tree T. The first step applies a number of structural transformations to the constituent tree to simplify conversion or to reflect theoretical considerations. Next, a head word is assigned to every constituent. After this, grammatical functions can be inferred, finally allowing a dependency tree to be created.

The first three steps convert a Penn-style constituent tree to a head-marked and function-annotated tree. An interesting contrast to the

Penn Treebank is the TIGER treebank of German (Brants et al., 2002). In this treebank, heads and functions are explicitly annotated, which makes the treebank easy to convert to dependency structures.

4.3.1 Preprocessing of Constituents

Before the two main steps of the conversion process – head selection and function labeling – can be carried out, the constituent structure needs to be modified to facilitate conversion. In addition, since the Penn Treebank reflects a theoretical tradition that differs from our surface-syntactic philosophy, the organization of constituents sometimes has to be restructured. These transformations are shown in Algorithm 4.2.

Algorithm 4.2 Preprocessing constituents.

```
procedure PREPROCESS($T$)
input Constituent tree $T$
  REMOVE-UNUSED-LABELS($T$)
  REATTACH-REFERENTS($T$)
  INSERT-AUXILIARY-HEAD-GROUPS($T$)
  INSERT-COORDINATION-PHRASES($T$)
  SPLIT-SMALL-CLAUSES($T$)
```

The first step is to remove secondary labels that are not used. We removed four Penn Treebank edge labels that reflect a structural property rather than a grammatical function: HLN (headline), TTL (title), NOM (non-NP acting as a nominal), and TPC (topicalization). The final one, topicalization, represents a property of a phrase that is arguably more semantically relevant than the three others, e.g. when analyzing the rhetorical structure. However, we still think that is a property that is orthogonal to grammatical function – after all, an object is just an object whether fronted or not.

The CLR ("closely related") label is also worth a discussion. It represents a number of complemental relations, such as the relation between verb and preposition in *depend on* or between verb and noun in *take part*. While useful, and especially for semantics, it has been shown that this tag is not consistently annotated. In this work, we kept the CLR label despite its inconsistencies[6].

[6]In the CoNLL-2008 treebank, on the other hand, we removed the CLR label for compatibility reasons since it is not used in the constituent annotation of the Brown Corpus. For a behind-the-scenes story about CLR, see Blaheta (2004), pages 5–6.

The next transformation, shown in Algorithm 4.3, was to reattach referents of secondary edges. The three types of long-distance dependencies handled in this step were

- discontinuity links, `*ICH*`,
- traces of "transformations" such as *wh*-movement and topicalization, `*T*`, based on the discussion in 4.2.3,
- right node raising, `*RNR*`. These links always come in pairs, but we used only the first of them due to the single-governor constraint.

We did not consider secondary edges representing "logical dependency" such as the logical object in passivization. Note that the algorithm has to take circularity into account, such as in Figure 4.1.

Figure 4.12 shows an example of how constituents are modified by the reattachment transformation.

Algorithm 4.3 Reattachment of referents of secondary edges.

```
procedure REATTACH-REFERENTS($T$)
input Constituent tree $T$
  for each empty category $t$ in $T$
    if $t$ is linked to a constituent $C$
    and the label of $t$ is in { *ICH*, *T*, *RNR* }
      if $C$ has a child $c$ that dominates $t$
        disconnect $c$ from $C$ and attach it to the parent of $C$
      disconnect the secondary edge
      disconnect $C$ and attach it to the parent of $t$
```

The reattachment of constituents sometimes results in nonprojective dependency links: in the CoNLL-2008 corpus, 0.4 percent of the dependencies were nonprojective. 7.6 percent of the sentences had at least one nonprojective link.

Next, Algorithm 4.4 brackets the head part of `VP`s and `PP`s, which is flatly annotated in the Treebank. Recall that C_i refers to the i-th child of C according to the start-token order. We refer to the inserted constituents as "head groups" (`VG` and `PG`). While also linguistically justified, the main purpose of the transformation is to make the processing of coordination more uniform. The effect on the algorithm on a prepositional phrase is shown in Figure 4.13.

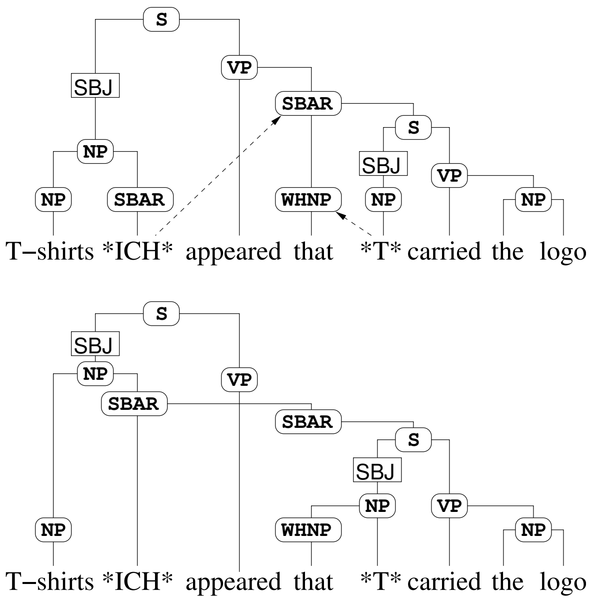

S
SBJ
VP
SBAR
NP
S
SBJ
VP
NP
SBAR
WHNP
NP
NP
T−shirts *ICH* appeared that *T* carried the logo
S
SBJ
NP
VP
SBAR
SBAR
S
SBJ
NP
VP
NP
WHNP
NP
T−shirts *ICH* appeared that *T* carried the logo

Figure 4.12: *Reattachment of referents of secondary edges.*

After auxiliary head groups have been inserted, coordinate structures can be processed. This transformation results in a right-branching structure using coordination phrases (&P). The treatment of coordination is based on the discussion in 4.2.4. Figure 4.14 exemplifies the effect of this transformation. Algorithm 4.5 shows the pseudocode. $|C|$ is a shorthand for the number of children of C, as defined in 3.2. A coordinator is either a single token with the CC tag, or a multiword conjunction, CONJP. A separator is a single token whose tag is either , or :. The procedure uses a heuristic IS-COORDINATED that determines whether or not coordination is present. The most interesting

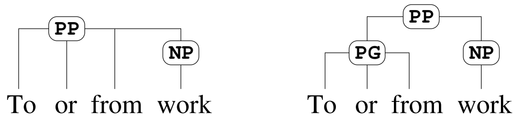


Figure 4.13: *Auxiliary head groups in a prepositional phrase.*

Algorithm 4.4 Insertion of auxiliary head groups.

```
procedure INSERT-AUXILIARY-HEAD-GROUPS($T$)
input Constituent tree $T$
  for every nonterminal constituent $C$ in $T$
    if $C$ is a VP
      $i \leftarrow$ start-token index of first finite verb child of $C$
      $j \leftarrow$ start-token index of last finite verb child of $C$
      BRACKET($C, i, j$,VG)
    else if $C$ is a PP
      $i \leftarrow$ start-token index of first preposition child of $C$
      $j \leftarrow$ start-token index of last preposition child of $C$
      BRACKET($C, i, j$,PG)

function BRACKET($C, i, j, l$)
input Constituent $C$, left and right index $i, j$, label $l$
  create a constituent $G$ whose label is $l$
  move the children $C_i, \ldots C_j$ from $C$ to $G$
  add $G$ as a child of $C$
  return $G$
```

case is for noun phrases, where a comma-separated structure without coordinators may also be an apposition. We used a simple heuristic based on the number of separators.

The final transformation in the preprocessing step, Algorithm 4.6, is based on the discussion in 4.2.5. It raises the logical subjects of small clauses to object position in the surrounding verb phrases. Function labels are also added: if the small clause has no label, it receives the OPRD label. Otherwise, PRD labels are replaced by OPRD. Figures 4.15 and 4.16 show examples of how small clauses are treated. Note that the transformation only applies to small clauses in object position.

4.3.2 Head Token Assignment

As described above, the fundamental principle of the constituent-to-dependency conversion process is based on the idea that every constituent can be assigned a *head token*, a single word that is the main controller of the grammatical behavior of the whole structure. This principle underpins most modern theories of syntax, including main-

Algorithm 4.5 Insertion of coordination phrases.

```
procedure INSERT-COORDINATION-PHRASES($T$)
input Constituent tree $T$
  for every nonterminal constituent $C$ in $T$
    if IS-COORDINATED($C$)
      $l \leftarrow$ the constituent label of $C$
      for $i \in [|C|, \ldots, 2]$
        if $C_{i-1}$ is a coordinator or separator
        and $C_i$ is not a coordinator
          $c \leftarrow$BRACKET($C, i, |C|, l$)
          set the secondary label of $c$ to CONJ
        if $C_i$ is a coordinator or $C_{i-1}$ is a separator
          $c \leftarrow$BRACKET($C, i, |C|$,&P)
          set the secondary label of $c$ to COORD

function IS-COORDINATED($C$)
input Constituent $C$
  if $C$ has the constituent label UCP, return True
  if $C$ has a coordinator child that is not leftmost, return True
  if $C$ has a separator child $c$, and $c$ is not leftmost or rightmost
    if $C$ is an NP and the number of separators is 1, return False
    else return True
  else return False
```

stream generative linguistics, where it is one of the main assumptions of the X-bar system.

The method to assign heads to constituents is based on the idea of *percolation* and is directly taken from previous work. The procedure assigns a *head child* in every constituent. These can then be used to recursively compute head words of every constituent, as shown in Algorithm 4.7. The main effort is thus to select head children (FIND-HEAD-CHILD), and since the Penn Treebank does not annotate this information, we had to resort to a set of rules.

Noun phrases need special consideration. Their selection procedure is shown in Algorithm 4.8. We apply this algorithm to constituents with labels NP, NX, WHNP, and NML (noun phrases added by Vadas and Curran, 2007). Note that grammatical function labels are taken into account in this procedure. The purpose of this is that the function label indicates that the child has a syntactic function with respect to

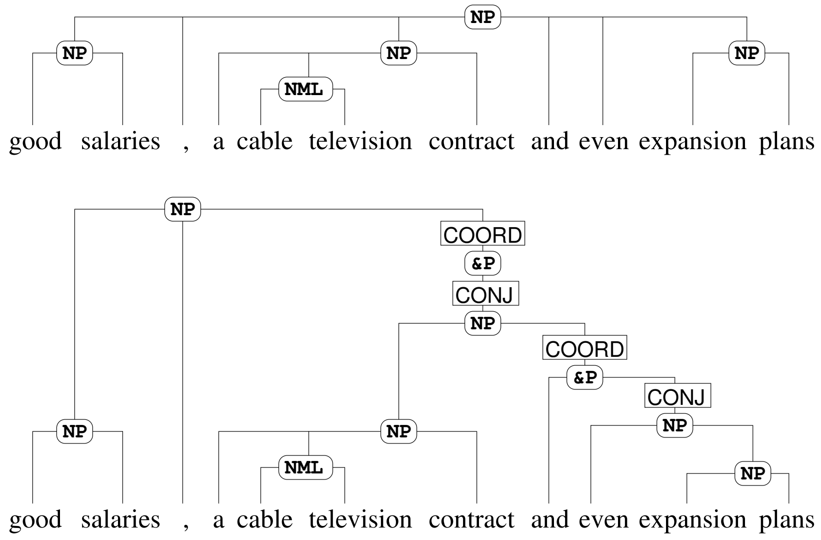

NP
NP
NP
NML
NP
good salaries , a cable television contract and even expansion plans
NP
COORD
&P
CONJ
NP
COORD
&P
CONJ
NP
NP
NP
NML
NP
good salaries , a cable television contract and even expansion plans

Figure 4.14: *Insertion of coordination phrases (`&P`).*

Algorithm 4.6 Splitting small clauses.

```
procedure SPLIT-SMALL-CLAUSES($T$)
input Constituent tree $T$
  for each verb phrase $C$ in $T$
    if $C$ has a child $S$ and the phrase label of $S$ is S
    and $S$ is not preceded by a “ or , tag
    and $S$ has a subject child $s$ that is not in genitive
      move $s$ from $S$ to $C$
      set the secondary label of $s$ to OBJ
      if $S$ has a child with a label $l$ containing PRD
        replace PRD with OPRD in $l$
        set the secondary label of $S$ to $l$
      else
        set the secondary label of $S$ to OPRD
```

the whole constituent, which is a clear indication that it should not be regarded as the head.

To find the head children for other types of constituents, a system

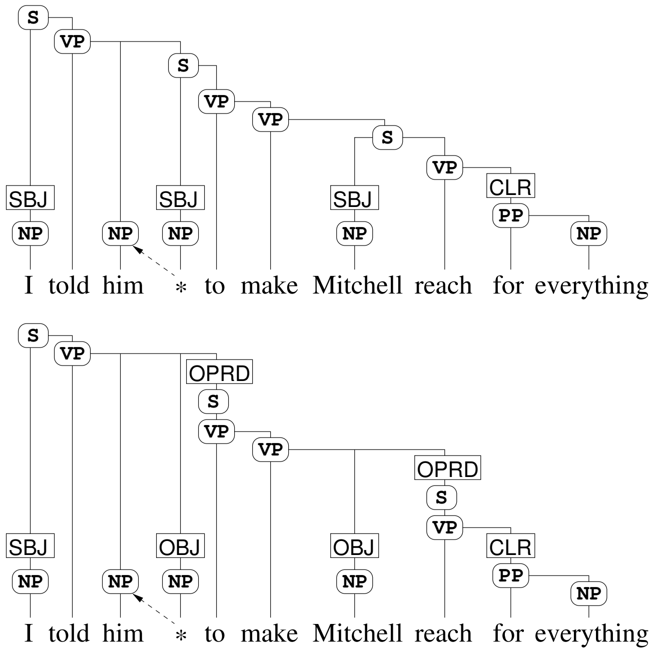


Figure 4.15: *Splitting of small clauses.*

Algorithm 4.7 Head percolation.

```
procedure ASSIGN-HEADS($T$)
input Constituent tree $T$
  PERCOLATE($T$.root)

procedure PERCOLATE($N$)
input Constituent $N$
  if $N$ is a terminal node
    $N$.head $\leftarrow$ $N$
  else
    for each child $C$ of $N$
      PERCOLATE($C$)
    $H \leftarrow$ FIND-HEAD-CHILD($N$)
    $N$.head $\leftarrow$ $H$.head
```

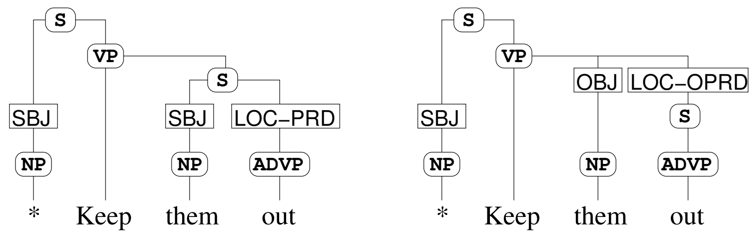


Figure 4.16: *Relabeling in small clauses.*

of rules is used (Table 4.1), based on the system by Yamada and Matsumoto (2003) but changed to reflect the modified constituent structure described in 4.3.1. The first column in the table indicates the constituent label, the second is the search direction, and the third is a priority list of phrase types to look for. For instance, to find the head of an `S`, we look from right to left for a `VP`. If no `VP` is found, look for anything with a `PRD` function tag, and so on. In addition, all rules implicitly prefer non-punctuation tokens over punctuation, and surface words over empty categories.

Algorithm 4.8 Head child selection for noun phrases.

```
function FIND-HEAD-CHILD-NP($N$)
input Noun phrase $N$
    Search ← for NN, NNP, NNPS, NNS, NX, or JJR without function.
    Else search → for NP, NML, or WHNP without function.
    Else search → for $ or #.
    Else search ← for CD.
    Else search ← for JJ, JJS, RB, QP.
    Else search → for a determiner that is not a, an, or the.
    Else select the last non-punctuation child.
```

4.3.3 Function Labeling

The Penn Treebank (versions II and III) annotates some constituents with grammatical function labels. However, this annotation is not complete, and to be able to convert the constituents to dependencies, our algorithms have to insert more function labels.

&P	→	CC|CONJP
ADJP, JJP	←	NNS QP NN $ ADVP JJ VBN VBG ADJP|JJP JJR NP JJS DT FW RBR RBS SBAR RB
ADVP	→	RB RBR RBS FW ADVP TO CD JJR JJ IN NP JJS NN
CONJP	→	CC RB IN
FRAG	→	NN⋆|NP W⋆ SBAR PP|IN ADJP|JJ|JJP ADVP RB
INTJ	←	⋆
LST	→	LS :
PG	→	IN TO VBG VBN FW
PP, WHPP	→	PG
PRN	→	S⋆ N⋆ W⋆ PP|IN ADJP|JJ⋆ ADVP|RB⋆
PRT	→	RP
QP	←	$ IN NNS NN JJ RB DT CD QP JJR JJS
RRC	→	VP NP ADVP ADJP|JJP PP
S	←	VP ⋆-PRD S SBAR ADJP|JJP UCP NP
SBAR	←	PG S SQ SINV SBAR FRAG
SBARQ	←	SQ S SINV SBARQ FRAG
SINV	←	VG VP ⋆-PRD S SINV ADJP NP
SQ	←	VG ⋆-PRD VP SQ
UCP	→	⋆
VG	→	VB⋆
VP	→	VG VP ⋆-PRD ADJP NN NNS NP
WHADJP	←	CC WRB JJ ADJP|JJP
WHADVP	→	CC WRB
X	→	⋆

Table 4.1: *Head child selection rules.*

Algorithm 4.9 shows how function labels are added to constituents lacking one. Appendix A lists the complete set of grammatical function tags used in the dependency treebanks.

The algorithm makes use of an auxiliary procedure POTENTIAL-OBJECT, shown separately in Algorithm 4.10, that determines whether or not a constituent (if it appears under a verb phrase) can be interpreted as an object. In the Penn Treebank, the absence of a function label on a constituent in a verb phrase indicates that it should be taken as a complement. The most common type of object is a noun phrase directly subordinated under a verb phrase, but we also include subordinated clauses, either in the form of direct speech (S, SINV, SQ, SBARQ) or not (SBAR). Note that we assume that small clauses already have been assigned a function tag at this stage, so they are not assumed to be objects. The heuristics used for SBARs in POTENTIAL-OBJECT are formally redundant but were added for robustness to filter out constituents where Penn annotators had forgotten to add a grammatical function tag.

The function labeling algorithm distinguishes direct and indirect objects. Procedurally, we define the indirect object as the *first of two objects*. Adding the IOBJ labels is not problematic if there is more than one object, in which case the IOBJ label is assigned to the first of them. However, if we make a distinction between direct and indirect object, it is not clear that there won't occur cases where there is only a single object, but that object should have an IOBJ function tag (such as in *Tell me!*). To have an idea of the number of such cases, we inspected a large set of instances of the verbs *give*, *tell*, and *provide*. Fortunately, the Treebank annotates most of those cases with an empty node to denote a missing object, although there are a few annotation errors that make the rule fail. Note that the IOBJ label was not used in the CoNLL-2008 treebank.

Regarding a few of the function tags from Penn, we introduced minor modifications. The adverbial tag, ADV, was extended to all unmarked RB, ADVP and PP nodes in verb phrases. According to Penn annotation conventions, ADV is implicit in these cases. The label representing the logical subject in passive clauses, LGS, was moved to the edge between the verb phrase and *by*, rather than the edge between *by* and the noun phrase.

In addition, it is necessary to mention the adverbial function tags used in the Penn Treebank, shown in Table 4.2. It could clearly be argued that the distinction between these functions is not syntactic but semantic, and that therefore they should not be reflected in a surface-syntactic framework. Indeed, most of them have a direct counterpart in PropBank. For instance, a phrase carrying the grammatical function label TMP is highly likely to be marked as a semantic adjunct ARGM-TMP in PropBank. This semantic information is sometimes orthogonal to the dimension of grammatical functions. This leads to non-atomic labels such as TMP-CLR, TMP-PRD.

The TMP label may appear in the following four different syntactic contexts.

- As an adjunct of a verb (TMP).
- As a complement of a verb (TMP-CLR).
- As an adjunct or complement of a nominal (TMP).

Algorithm 4.9 Function labeling.

```
procedure ASSIGN-FUNCTIONS($T$)
input Constituent tree $T$
  for each constituent $C$ in $T$
    if $C$ is the root constituent
      set the function label of $C$ to ROOT
    else if $C$ has no function tag from Penn or previous stages
      $L \leftarrow$ INFER-FUNCTION($C$)
      set the function label of $C$ to $L$

function INFER-FUNCTION($C$)
input Constituent $C$
  let $c$ be the head of $C$, $P$ the parent of $C$, $p$ the head of $P$,
    and $R$ the right sibling of $C$
  if $c = p$ return $\emptyset$
  if POTENTIAL-OBJECT($C$) and POTENTIAL-OBJECT($R$) return IOBJ
  if POTENTIAL-OBJECT($C$) return OBJ
  if $C$ is PRN return PRN
  if $c$ is punctuation return P
  if $C$ is PP, ADVP, or SBAR and $P$ is VP return ADV
  if $C$ is PRT and $P$ is VP return PRT
  if $p$ is TO and $C$ is VP return IM
  if $C$ is VP and $P$ is VP, SQ, or SINV return VC
  if $P$ is SBAR and $p$ is IN return SUB
  if $P$ is VP, S, SBAR, SBARQ, SINV, or SQ
    and $C$ is RB or ADVP return ADV
  if $P$ is NP, NX, NAC, NML, or WHNP return NMOD
  if $P$ is ADJP, ADVP, JJP, WHADJP, or WHADVP return AMOD
  if $P$ is PP or WHPP return PMOD
  else return DEP
```

- As a predicative complement of a copula (TMP−PRD).

However, we kept the adverbial tags since we think this information can be useful, and should not be discarded unless it can be argued that it is harmful, not only on purist grounds. With some effort, it might also be said that these distinctions *are* meaningful in surface syntax. Many languages have word order preferences that require that temporal adjuncts generally should go before locative adjuncts, for instance.

Algorithm 4.10 Finding potential objects.

```
function POTENTIAL-OBJECT($C$)
input Constituent $C$
  if $C$ has a function label other than OBJ return False
  if $C$ is an NP, S, SQ, SINV, or SBARQ return True
  if $C$ is a UCP
      for each child $C_i$ of $C$
          if POTENTIAL-OBJECT($C_i$) return True
  if $C$ is an SBAR
      if $C$.head is as, because, for, since or with return False
      else return True
  else return False
```

Label	Definition
ADV	General adverbial
DIR	Direction
EXT	Extent
LOC	Location
MNR	Manner
PRP	Purpose / reason
TMP	Temporal

Table 4.2: *Adverbial function tags in the Penn Treebank.*

4.3.4 Dependency Tree Creation

After the preceding steps, which produce a richly labeled constituent tree, the final step of building the dependency tree is a formality. Algorithm 4.11 shows the pseudocode.

Algorithm 4.11 Creation of a dependency tree from a labeled and head-marked constituent tree.

```
function CREATE-DEPENDENCY-TREE($(T)$)
input Constituent tree $T$
  $D \leftarrow \{\}$
  for each token $t$ in $T$
    let $C$ be the highest constituent that $t$ is the head of
    let $P$ be the parent of $C$
    let $l$ be the secondary label of $C$
    $D \leftarrow D \cup P.\text{head} \xrightarrow{l} t$
  return $D$
```

Chapter 5

Dependency-based Role-Semantic Analysis

This chapter shows how the role-semantic representations described in Chapter 2 can be *automatically* constructed by algorithms. The architecture that we describe in this chapter has been implemented in a system that took part in an international evaluation, the SemEval-2007 task on frame-semantic structure extraction (Baker, Ellsworth, and Erk, 2007) which was based on FrameNet. We will refer to this implementation as the FSSE system. In addition, we will also evaluate a similar system on the test corpus from the CoNLL-2008 Shared Task (Surdeanu et al., 2008), based on PropBank and NomBank. This will be referred to as the CoNLL-2008 baseline system – the reason for using the word *baseline* is that we will describe extensions to it in Chapter 7.

We give a brief overview of the field of automatic semantic role labeling. While previously published systems were based on constituent-syntactic input, we demonstrate that a sequential, classifier-based semantic role labeling architecture can be similarly implemented with syntactic input in the form of dependencies constructed by the algorithms described in Chapter 4.

In contrast to our definitions in Chapter 2 of semantic role structures, which were purely based on logic, most annotation systems for semantic role structures are based on labeled segmentation. This is also reflected in the evaluation metrics usually applied for automatic semantic role labelers. We briefly discuss the metrics that have been proposed in literature.

5.1 Automatic Semantic Role Labeling

Automatic determination of semantic roles has a long tradition in natural language processing, and has been used by early systems for semantic interpretation (Hirst, 1983, *inter alia*). However, its breakthrough in modern statistical language processing came with the work by Gildea and Jurafsky (2000). The subsequent article (Gildea and Jurafsky, 2002) has become the standard reference in the field. Around the same time, early work was pursued on automatic analysis of the tectogrammatical layer of the Prague Treebank (Žabokrtský, 2000; Žabokrtský, Sgall, and Džeroski, 2002).

The work by Gildea and Jurafsky (2000) proposed a division of the problem of semantic role analysis into two main subproblems, argument identification and argument labeling, that is still widely used. This work also identified a number of features used by statistical classifiers, which are also more or less standard practice in modern systems: for instance, the parse tree path to capture the grammatical relation between predicate and argument, voice, and lexical features of predicate and argument. It also established the tradition of relying on syntactic parse trees as input, both in determining the possible arguments of a predicate and for extracting classifier features. In addition, a number of extensions are explored: syntactic–semantic integration, generalization of lexical features based on WordNet and automatic clustering, and generalization to unseen domains.

Gildea and Jurafsky's work became the starting point of a wide range of extensions. The first major improvement was the replacement of the smoothed probabilistic models by more sophisticated statistical methods. Early examples include maximum-entropy classifiers used by Fleischman, Kwon, and Hovy (2003), decision trees by Surdeanu et al. (2003), and support vector machines by Hacioglu and Ward (2003).

While considerable performance gains could be achieved by just changing the classification method, there was also much to be done in the design of features for the classifiers. Surdeanu et al. (2003) added a number of features such as the "content word," a semantically prominent word that helps role disambiguation when the head is a grammatical word. Xue and Palmer (2004) and Pradhan et al. (2005a) studied the effect of several possible features.

The enterprise of designing features for semantic role labelers soon reached a ceiling, beyond which improvements were very small. To improve performance further, attention switched to devising new architectures. One of the most common strategies is combination of the output from multiple systems. An example is the method by Pradhan et al. (2005b), in which a chunk-based semantic role labeler uses the output of a number of systems as classifier features. Punyakanok, Roth, and Yih (2005) combined systems by means of optimization under a carefully designed set of hand-coded linguistic constraints. It seems that diversity of the candidate set is an important parameter influencing the success of combination methods (Màrquez et al., 2005). Apart from the combination of multiple systems, another architectural innovation that is nowadays commonly used is reranking of complete predicate–argument structures (Toutanova, Haghighi, and Manning, 2005).

5.2 The Tasks

As mentioned in the introduction, the systems based on the architecture described in this chapter have been evaluated on test data from two international evaluations of semantic role analyzers, the SemEval-2007 task on frame-semantic structure extraction (FSSE) and the CoNLL-2008 Shared Task on joint syntactic and semantic analysis. This section describes task definitions of the FSSE task and the CoNLL-2008 Shared Task. We postpone the discussion of their evaluation metrics to 5.4.

Apart from these two, a number evaluations of semantic role labeling performance have been carried out since 2004. In the CoNLL-2004 Shared Task (Carreras and Màrquez, 2004), the participants trained and evaluated on a small portion of the PropBank corpus annotated with shallow syntax; predictably, the results were modest. The Senseval-3 task on automatic semantic role labeling, which used FrameNet, achieved higher results (Litkowski, 2004). In the CoNLL-2005 Shared Task (Carreras and Màrquez, 2005), in contrast to the 2004 task, the full PropBank corpus was used, and the participants had access to parser output from full constituent parsers. The data set used in this task has become a standard benchmark for the field.

5.2.1 SemEval-2007 Task on Frame-semantic Structure Extraction

The SemEval-2007 task on frame-semantic structure extraction (FSSE), evaluated the performance of semantic role analysis in the FrameNet paradigm (previously described in 2.3.1). In addition, the participating systems were required to identify predicates and assign them to

FrameNet frames, which made this task more complex than previous tasks.

The output of the systems consisted of labeled segments represented using the FrameNet annotation format. Example 5.1 shows an example of the annotated sentence *Dublin excels in packaging its past for the visitor*. The frame is EXPERTISE and there are two segments annotated with role labels, PROTAGONIST and SKILL.

Example 5.1.
[Dublin]$_{\text{PROTAGONIST}}$ EXPERTISE:**excels** [in packaging its past for the visitor]$_{\text{SKILL}}$.

The main resource used by the participants was version 1.3 of FrameNet. This package contained the following parts:

- A lexical database defining frames and mapping predicate words to frames,
- A collection of annotated example sentences for each predicate word, taken from the British National Corpus, 139,439 sentences,
- A corpus of running text that was annotated for the task. These texts were taken from two specific domains: public information about weapons of mass destruction from the Nuclear Threat Initiative (NTI), and travel guide books from Berlitz.

The running text corpus was split into a training part, consisting of 1,728 sentences, and a test part of 120 sentences. The test part was used to score the participating systems. This corpus had 5.9 annotated predicates per sentence.

The majority of the training data thus consisted of "representative" examples sampled from the BNC, since the example collection was much larger than the corpus of running text. This of course leads to a skewed distribution of predicates and arguments. Still, training on the running text corpus only would not be feasible since it is too small. The test corpus was very different from the training data and contained many unseen predicates (and even a number of unseen frames), and proved very difficult for participants.

Apart from FrameNet, there were no restrictions on the tools or resources that the participants could use. Our system used a part-of-speech tagger (Toutanova et al., 2003) and the MaltParser dependency parser (Nivre et al., 2007). In addition, we used a method (Johansson and Nugues, 2007d) to expand the FrameNet dictionary by using WordNet (Fellbaum, 1998), which we will not detail here.

5.2.2 CoNLL-2008 Shared Task on Joint Syntactic and Semantic Analysis

In the second evaluation that we describe here, the CoNLL-2008 Shared Task on joint syntactic and semantic analysis, the participants carried out both dependency parsing and semantic analysis. PropBank and NomBank (see 2.3.3) were used as the representational formalism to annotate predicates and arguments. Like the FSSE task, the task involved identification and disambiguation of predicates in addition to argument identification and labeling.

A novelty compared to previous evaluations of semantic role annotation was that a fully dependency-based representation was used, not only for syntax but also for semantics: A semantic role was not annotated as a labeled segment, but as a labeled link between the predicate word and the argument head word. Figure 5.1 shows an example of the structures to extract in this task. There is one PropBank predicate *plan* and one NomBank predicate *investment*.

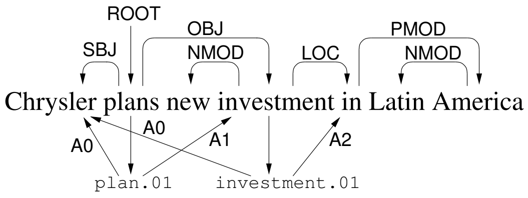


Figure 5.1: *Syntactic and role-semantic representations of a sentence in the CoNLL-2008 Shared Task.*

Similarly to the FSSE task, the participants had access to a the PropBank and NomBank lexicons, and sections 02 – 21 of the WSJ part of the Penn Treebank. Section 24 was used as a tuning set. No other external resources were allowed.[1] The WSJ corpus is all running text, so the skewedness problems of the FrameNet corpora are not an issue here.

The test was carried out using section 23 of WSJ and a small part of section K of the syntactically annotated Brown corpus. Since the Brown test set is very different in style and content from the training

[1]This constitutes the *closed setting*. In an *open setting*, participants were allowed to use any resource but the Treebank and PropBank/NomBank. However, the open setting attracted less interest from participants.

set, the performance on this set is interesting since it gives an idea of the domain sensitivity of the system.

5.3 Classifier-based Semantic Role Labeling using Dependency-Syntactic Input

This section describes an architecture for automatic semantic role labeling based on a sequence of statistical classifiers. This is a direct adaptation for dependency syntax of classifier-based semantic role labelers for constituent syntax, a prototypical example of which is the system by Pradhan et al. (2005a).

Our sequential classifier-based systems for semantic role analysis carry out these four subtasks, as shown in Figure 5.2:

- Identifying the words that should be analyzed as predicates,
- assigning word sense identifiers to the predicates,
- identifying the arguments of every predicate,
- assigning a semantic role label to every argument.

In most of the literature, the first two of these subtasks are not considered. However, since both the FSSE and the CoNLL tasks included predicate identification and disambiguation in the task definition, we include them here as well.

While the FSSE and CoNLL-2008 baseline systems are both designed according to the conceptual framework of a sequence of classifiers, as in Figure 5.2, there are some differences. The most important is that the FSSE system is based on FrameNet, while the CoNLL-2008 baseline system uses PropBank and NomBank. In Figure 5.2, PropBank/NomBank-style annotation is used: The sense identifiers and argument labels are numbers. In FrameNet, we use frame names as sense identifiers, so we would for instance replace `plan.01` with the frame name PURPOSE. Similarly, we would replace the numbered PropBank argument labels A0 and A1 with AGENT and GOAL, respectively.

There are also some engineering differences. In the system that participated in the FSSE task, the classifiers were support vector machines (Boser, Guyon, and Vapnik, 1992) with quadratic kernels that we trained using the LIBSVM package (Chang and Lin, 2001). After training, we converted the set of support vectors to explicit weights for

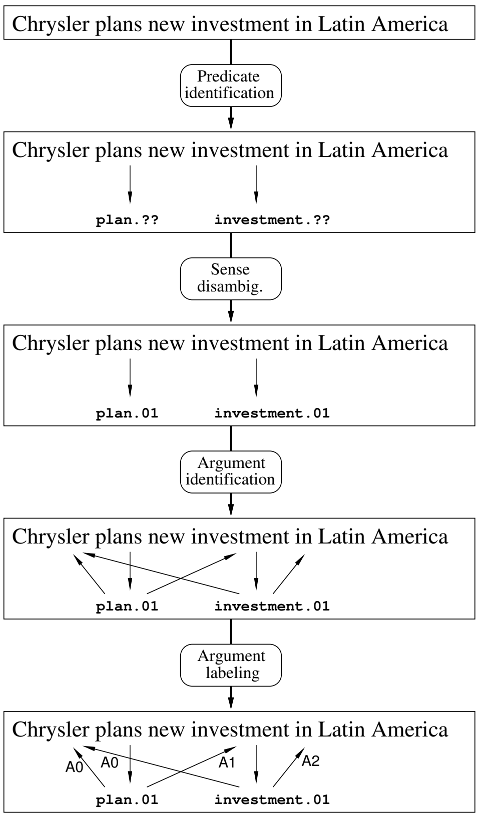

Chrysler plans new investment in Latin America
Predicate identification
Chrysler plans new investment in Latin America
plan.??
investment.??
Sense disambig.
Chrysler plans new investment in Latin America
plan.01
investment.01
Argument identification
Chrysler plans new investment in Latin America
plan.01
investment.01
Argument labeling
Chrysler plans new investment in Latin America
A0
A0
A1
A2
plan.01
investment.01

Figure 5.2: *Example processed by a sequence of classifiers.*

features and feature bigrams to speed up evaluation. For the CoNLL-2008 baseline system, we replaced the kernel-based classifiers with linear classifiers, implemented using the efficient LIBLINEAR package (Fan et al., 2008), which speeds up the training process by several orders of magnitude. We also replaced the support vector machines with L2-regularized logistic regression classifiers. There was no noticeable difference in classification accuracy between logistic and support vector classifiers, but we needed probabilistic output for the extended experiments in Chapter 7, for which logistic classifiers are better suited. An additional requirement when replacing kernel-based classifiers by linear ones is that feature bigrams must be added to the feature set, due to the limited expressivity of linear separators (Minsky and Papert, 1969). This increases feature selection time, but since training is faster with linear classifiers, the total time spent on training and feature selection is still reduced.

Finally, while the FSSE and CoNLL-2008 baseline system both used dependency trees as described in Chapter 4, the parsers were different: The SemEval-2007 system used MaltParser (Nivre et al., 2007), while the CoNLL-2008 baseline system used our own parser, which we will describe in detail in 7.3.

5.3.1 Predicate Identification

The predicate identification task consists of finding the words in a sentence that refer to semantic predicates. We describe two methods to carry out this task: rule-based filtering and a statistical classifier inspired by word sense disambiguation techniques.

Rule-based Predicate Identification

In the FSSE system, we applied a rule-based method to identify predicates. The reason for using rules rather than a statistical classifier was that the training corpus of running text was relatively small, which made it impractical to train a classifier.

We implemented the predicate identification component by using the FrameNet lexicon and a rule-based filter. A set of potential predicates was extracted in a sentence by finding all words (including multiwords) listed in FrameNet. The filtering rules were then applied to remove some of these predicate candidates. Most of the rules concerned prepositions, which have recently been added to FrameNet, and for which we had very little annotation.

We used the following set of filtering rules:

- *have* was retained only if it had an object,
- *be* was retained only if it was preceded by *there*,
- *will* was removed in its modal sense,
- The words *course* and *particular* were removed if part of the expressions *of course* or *in particular*, respectively,
- the prepositions *above*, *against*, *at*, *below*, *beside*, *by*, *in*, *on*, *over*, and *under* were removed unless their grammatical function was locative,

- *after* and *before* were removed unless their function was marked as temporal,
- *into*, *to*, and *through* were removed unless the function was direction,
- *as*, *for*, *so*, and *with* were always removed,
- since the only sense of *of* was PARTITIVE, we removed it unless it was preceded by *only*, *member*, *one*, *most*, *many*, *some*, *few*, *part*, *majority*, *minority*, *proportion*, *half*, *third*, *quarter*, *all*, or *none*, or if it was followed by *all*, *group*, *them*, or *us*.

Classifier-based Predicate Identification

In the CoNLL-2008 baseline system, we once again made use of lexicons to spot possible predicate words, in this case the PropBank and NomBank lexical databases. In our case, however, the filter was based on statistical classifiers instead of hand-written rules, since we had enough running text to train classifiers.

In principle, we treated the problem of predicate identification as a word sense disambiguation problem. While almost every word listed in PropBank or NomBank will always be annotated as a predicate when it appears in text, there are some words, such as the verb *have*, for which we needed to discriminate between a predicate sense and a non-predicate (auxiliary) sense.

Statistical word sense disambiguation of a word extracts features based on the context of the word. In this case, the context is based on the dependency tree. We extracted the following features:

PREDWORD, PREDLEMMA. The surface form and lemma of the predicate.

PREDGOVWORD and PREDGOVPOS. Form and part-of-speech tag of the governor node of the predicate.

DEPLABELS, DEPWORDS, DEPWORDLABELS, DEPPOSS, DEPPOSLABELS. These features describe the set[2] of dependents of the predicate using combinations of dependency labels, words, and parts of speech.

[2]Set-valued features are implemented as a set of Boolean features denoting the presence or absence of a member.

DEPSUBCAT. Subcategorization frame: the concatenation of the dependency labels of the predicate dependents, excluding parentheticals, punctuation, and coordinations.

PREDREL. Dependency relation between the predicate and its governor.

To give an example of the features used in predicate identification, Figure 5.3 shows an example of a sentence. The potential predicate is *had*. The most frequent sense of this word is auxiliary and should not be annotated as a predicate. The extracted features are listed in Table 5.1.

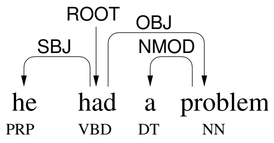


Figure 5.3: *Example of features in predicate identification and sense disambiguation.*

Feature	Value
PREDWORD	had
PREDLEMMA	have
PREDGOVWORD	ROOT
PREDGOVPOS	ROOT
DEPLABELS	{ SBJ, OBJ }
DEPWORDS	{ he, problem }
DEPPOSS	{ PRP, NN }
DEPWORDLABELS	{ he+SBJ, problem+OBJ }
DEPPOSLABELS	{ PRP+SBJ, NN+OBJ }
DEPSUBCAT	SBJ+OBJ
PREDREL	ROOT

Table 5.1: *Example of features in predicate identification and sense disambiguation.*

We used two classifiers: one for verb predicates and one for noun predicates. The size of the training set was 100,000 instances for the verb predicate identifier and 180,000 for the noun predicate identifier.

5.3.2 Predicate Sense Disambiguation

After a predicate has been identified, it needs to be assigned a sense identifier. In FrameNet, this is equivalent to assigning a frame name, and in PropBank/NomBank, we assign a sense label from the sense inventory of the lemma. The formulation of frame assignment as a word sense disambiguation problem comes from Erk (2005). In the running text corpus used for training in the FSSE task, around 40 percent of the predicates had more than one frame listed in the FrameNet database.

Similarly to the classifier-based predicate identifier, we used a statistical method to assign a sense identifier. For the FrameNet case, we trained one classifier for every frame (796 classifiers), which allows for some generalization to new lemmas. For PropBank/NomBank, we trained one classifier per lemma (7,737 classifiers), since the sense labels (which are just numbers) are not meaningful across lemmas. We used the same feature set as for the predicate identification classifier described above.

5.3.3 Argument Identification

The argument identification step finds the arguments for a given predicate. Following Gildea and Jurafsky (2002), the underlying assumption that makes this task possible is that the potential semantic arguments of a predicate are defined by a syntactic parse tree. While a semantic role relation conceptually holds between logical entities, this method sidesteps the messy issue of reference relations between surface words and logical concepts. We thus somewhat sloppily say that a node in a syntactic tree has a semantic role with respect to a surface predicate word. Gildea and Jurafsky (2002) implemented the argument identifier as a binary classifier applied to the nodes in a constituent tree, and this is still the most common method although there are many variants.

We used the same principle for identifying arguments: The possible arguments of a predicate are the nodes in a syntactic tree, although in our case this tree is a dependency tree, and a binary classifier decides whether or not a given parse tree node is an argument of a given predicate word. Moreover, to reduce the size of the training set and to balance the number of positive and negative examples, we applied a *pruning* algorithm, shown in Algorithm 5.1, that formalizes the intuition that arguments tend to be located directly under a predicate node or one of its ancestors in the tree. This is the adaption to dependency syntax of the well-known pruning algorithm by Xue and Palmer (2004).

Algorithm 5.1 Finding potential arguments.

```
function FIND-POTENTIAL-ARGUMENTS($p$)
input Predicate dependency node $p$
  $A \leftarrow [d : p \rightarrow d]$
  $n \leftarrow p$
  repeat
    $n \leftarrow \text{governor}(n)$
    $A \leftarrow A \cup [d : n \rightarrow d]$
  until $n$ is the root node
  return $A$
```

In the training data for the FSSE system, semantic arguments were annotated using labeled segments. To find a dependency node corresponding to a labeled segment, we selected the first node in the list returned by Algorithm 5.1 whose governor was outside the segment.

For the FSSE task, the training set consisted of 1,500,000 instances. Since we used quadratic support vector machines, it was infeasible to train on the full training set. We thus selected a random subset of 200,000 instances as the training set. The training time was roughly 24 hours. For the CoNLL-2008 task, the training set contained 750,000 instances for the verb predicates and 1,000,000 for noun predicates. However, since we used linear classifiers, training took only a few seconds.

5.3.4 Argument Labeling

The argument labeling step consists of assigning a semantic role label to an argument of a predicate. Similarly to the previous steps, we solved this by applying a statistical classifier, in this case a multiclass classifier. In both the Framenet and PropBank/NomBank case, the set of allowed semantic roles is defined by the predicate. To ensure that the output predicted by the classifier was consistent with the predicate, we used only the restricted set of binary subclassifiers[3] allowed by the predicate.

There is a conceptual problem with having a single classifier for all predicates: it overgeneralizes. A semantic role label is only guaranteed to have the same meaning with respect to a single frame (for FrameNet)

[3]In the FSSE system, we used the one-versus-one binarization method that comes with LIBSVM. In the CoNLL-2008 baseline system, we implemented a one-versus-all binarization method.

or lexical unit (for PropBank/NomBank). For instance, the meaning of the CONTENT label in the FrameNet frame EXPERIENCER_SUBJ is not the same as in the frame SOCIABILITY. This problem, while relatively minor in FrameNet, is more marked in PropBank/NomBank, where only the ARG0 and ARG1 labels have a more or less consistent interpretation as “agent-like” and “patient-like” roles, respectively. However, the method of having a single classifier is still commonly used since it gives some generalization between frames or predicates, although this generalization may not be sound. While there are some FrameNet-based systems that train an argument labeler for every frame, it is not feasible to train a PropBank/NomBank argument labeler for every lemma – the training data would be too sparse.

The training set for the argument labeler contained 250,000 instances for the FSSE system. This classifier could be trained faster than the argument identifier since it consists of a large number of binary subclassifier, which allowed parallellization of the training process. For the CoNLL-2008 baseline system, we had 225,000 instances for verb predicates and 150,000 for noun predicates.

5.3.5 Features Used in Argument Identification and Labeling

We used roughly the same feature sets for argument identification and classification. This section describes the features. In the FSSE system, we used all these features, while in the CoNLL-2008 baseline system, we used a feature selection process described in 5.3.6.

PREDLEMMASENSE. The lemma and sense number of the predicate, e.g. *give.01*.

VOICE. For verbs, this feature is Active or Passive. For nouns, it is not defined.

POSITION. Position of the argument with respect to the predicate: Before, After, or On.

ARGWORD and ARGPOS. Lexical form and part-of-speech tag of the argument node.

LEFTWORD, LEFTPOS, RIGHTWORD, RIGHTPOS. Form/part-of-speech tag of the leftmost/rightmost dependent of the argument.

LEFTSIBLINGWORD, LEFTSIBLINGPOS, RIGHTSIBLINGWORD, RIGHTSIBLINGPOS. Form/part-of-speech tag of the left/right sibling of the argument.

PREDPOS. Part-of-speech tag of the predicate.

RELPATH. A representation of the complex grammatical relation between the predicate and the argument. It consists of the sequence of dependency relation labels and link directions in the path between predicate and argument, e.g. `IM↑OPRD↑OBJ↓`.

POSPATH. An alternative view of the grammatical relation, which consists of the POS tags passed when moving from predicate to argument, e.g. `VB↑TO↑VBP↓PRP`.

VERBCHAINHASSUBJ. Binary feature that is set to true if the predicate verb chain has a subject. The purpose of this feature is to resolve verb coordination ambiguity as in Figure 5.4.

CONTROLLERHASOBJ. Binary feature that is true if the link between the predicate verb chain and its governor is OPRD, and the governor has an object. This feature is meant to resolve control ambiguity as in Figure 5.5.

FUNCTION. The grammatical function of the argument node. For direct dependents of the predicate, this is identical to the RELPATH.

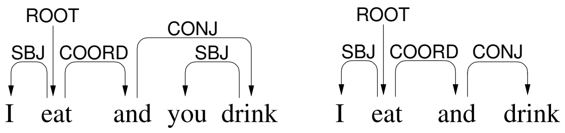


Figure 5.4: *Coordination ambiguity: The subject* I *is in an ambiguous position with respect to* drink.

5.3.6 Feature Selection in the CoNLL-2008 Baseline

The feature sets used by the argument identification and labeling classifiers in the CoNLL-2008 baseline system are shown in Table 5.2. In the table, N or V denotes a feature used by a classifier for noun or verb predicates, respectively.

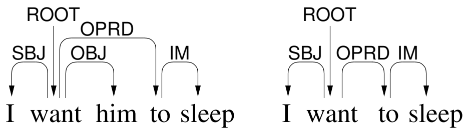


Figure 5.5: *Subject/object control ambiguity:* I *is in an ambiguous position with respect to* sleep.

To select the feature sets, we carried out a greedy forward feature search procedure, shown in Algorithm 5.2. Due to the implementation of classifiers, this was not possible for the FSSE system.

From a set F of features, the algorithm incrementally adds one feature to a working set until no progress is made. To measure progress, an evaluation metric μ is used, in our case the average F-measure in a 5-fold cross-validation on the training set. Also, a tolerance parameter ϵ is needed, and we used a value of 10^{-5}. Since we used linear classifiers, we also needed to select feature pairs, and this is done in the second part of the algorithm. Tables 5.3 and 5.4 list the first 10 features found by the feature selection process for every classifier (feature pairs are omitted). For every feature, the tables also show the improvement in F-measure.

We see that both subtasks seem to be more lexicalized for noun predicates; grammatical features are not expressive enough to disambiguate. A partial reason for this may be that the inventory of dependency relations that we use in noun phrases is limited – as we saw in 4.3.3, we mainly use NMOD except for some locative or temporal modifiers, where we used LOC or TMP, respectively. No complement/adjunct distinction is made since this information is not available in the Treebank. However, the problem does not only stem from our encoding of grammatical structure, but also from the limited grammatical expressivity in nominal structures. For instance, in Example 5.2 we see that what is expressed using a genitive in a nominalization may correspond to either a subject or object in the corresponding structure with a verb predicate.

Example 5.2.
(a) Alexander arrived / Alexander's arrival
(b) They acquitted Alexander / Alexander's acquittal

Feature	ArgId	ArgLbl
PREDGOVWORD/POS	N,V	
DEPLABELSET	N,V	N,V
PREDLEMMASENSE	N,V	N,V
VOICE	V	V
POSITION	N,V	N,V
ARGWORD/POS	N,V	N,V
LEFTWORD/POS	N	N,V
RIGHTWORD/POS	N,V	N,V
LEFTSIBLINGWORD/POS		N,V
RIGHTSIBLINGWORD/POS	N	N
PREDPOS	N,V	V
RELPATH	N,V	N,V
POSPATH	N	
VERBCHAINHASSUBJ	V	V
CONTROLLERHASOBJ	V	N
PREDREL	N,V	N,V
FUNCTION		N,V

Table 5.2: *Classifier features in argument identification (ArgId) and labeling (ArgLbl).*

Feature	Contribution
RELPATH	0.934
ARGWORD	0.0191
VERBCHAINHASSUBJ	0.00204
CONTROLLERHASOBJ	0.00227
PREDLEMMASENSE	8.67e-4
RIGHTPOS	2.92e-4
PREDREL	3.43e-4
ARGPOS	2.67e-4
CHILDDEPSET	2.71e-4
PREDPOS	7.62e-5

Feature	Contribution
RELPATH	0.597
ARGWORD	0.186
PREDLEMMASENSE	0.00594
ARGPOS	0.00403
RIGHTPOS	0.00240
CHILDDEPSET	0.00120
LEFTPOS	5.15e-4
POSPATH	4.39e-4
PREDREL	2.47e-4
POSITION	2.11e-4

Table 5.3: *Contributions of individual features for argument identification for verb (left) and noun (right) predicates.*

5.4 On Evaluating Semantic Role Annotation Quality

Starting in Chapter 2, we defined semantic roles as logical relations holding between an event and its participants. In role-annotated text,

Algorithm 5.2 Greedy forward feature selection.

```
function GREEDY-FORWARD-SELECTION($F, \mu, \epsilon$)
input Feature set $F$, evaluation metric $\mu$, tolerance $\epsilon$
  $S \leftarrow \emptyset$
  $P \leftarrow \emptyset$
  $\mu_{max} \leftarrow -\infty$
  repeat
    $\mu_{curr} \leftarrow \mu_{max}$
    for each $f \in F \setminus S$
      $\mu' \leftarrow \mu(S \cup \{f\}, P)$
      if $\mu' > \mu_{max} + \epsilon$
        $\mu_{max} \leftarrow \mu'$
        $f_{max} \leftarrow f$
    if $\mu_{max} > \mu_{curr}$
      $S \leftarrow S \cup \{f_{max}\}$
  until $\mu_{max} = \mu_{curr}$
  repeat
    $\mu_{curr} \leftarrow \mu_{max}$
    for each $p \in (F \times F) \setminus P$
      $\mu' \leftarrow \mu(S, P \cup \{p\})$
      if $\mu' > \mu_{max} + \epsilon$
        $\mu_{max} \leftarrow \mu'$
        $p_{max} \leftarrow p$
    if $\mu_{max} > \mu_{curr}$
      $P \leftarrow P \cup \{p_{max}\}$
  until $\mu_{max} = \mu_{curr}$
  return $\langle S, P \rangle$
```

these relations are typically annotated by *bracketing* the referring words (predicate and argument words).

The most commonly used metric is the *strict segment metric*, which assumes a bracketed annotation. This metric was used in the 2004 and 2005 CoNLL Shared Tasks and in the FSSE task (among other metrics). In this metric, a labeled segment is counted as correct if the gold standard contains a segment with the same boundaries and label. This is then used to compute the following two measures:

$$\text{precision} = \frac{\text{\#correct}}{\text{\#attempted}} \quad \text{and} \quad \text{recall} = \frac{\text{\#correct}}{\text{\#in gold standard}}$$

Since it is impractical to rank systems when having two separate

Feature	Contribution
RelPath	0.690
ArgWord	0.105
PredLemmaSense	0.0678
ChildDepSet	0.0286
RightWord	0.00428
Voice	0.00376
ArgPOS	0.00300
Position	0.00211
LeftPOS	0.00101
RightPOS	7.95e-4

Feature	Contribution
ArgWord	0.267
PredLemmaSense	0.0908
RelPath	0.0237
RightWord	0.0122
PredPOS	0.00694
LeftSiblingPOS	0.00106
LeftPOS	7.85e-4
LeftWord	9.19e-4
ChildDepSet	9.91e-4
RightPOS	2.41e-4

Table 5.4: *Contributions of individual features for argument labeling for verb (left) predicates and noun (right) predicates.*

measures, the precision and recall measures are typically combined into a single figure using a harmonic mean, the F1-measure (or just F-measure):

$$F_1 = \frac{2PR}{P+R}$$

However, it is questionable whether a segment-based metric is the proper way to evaluate systems whose purpose is to find *semantic relations* between logical entities. We believe that the same criticisms that have been leveled at the bracket-based PARSEVAL metric for constituent structures (Lin, 1998) are equally valid for the segment-based evaluation of SRL systems.

In the end, the evaluation metric must be related to the real-world task that a semantic role labeler is intended to carry out. If it is to be used as a template filler, then a segment-based metric is perfectly natural. If its task is to extract semantic information in some sort of logical formalism, or provide features for a bag-of-words representation, then a segment metric is probably misleading.

5.4.1 Evaluation in FSSE

In the FSSE task, the participating systems were scored according to a number of different metrics, evaluating the performance for both predicates and arguments. The evaluation metric puts more weight on identification and labeling of predicates than of arguments.

For the semantic role annotation, the participants were required to submit conventional labeled segment annotation, but the output was not only scored using the segment metric discussed above, but also a *dependency metric* that was intended to be more semantically relevant. The system outputs were automatically converted to *semantic dependency graphs*, which were then compared to the gold standard. Figure 5.6 shows the semantic dependency graph representing the sentence *This geography is important in understanding Dublin*, taken from the paper by Baker, Ellsworth, and Erk (2007).

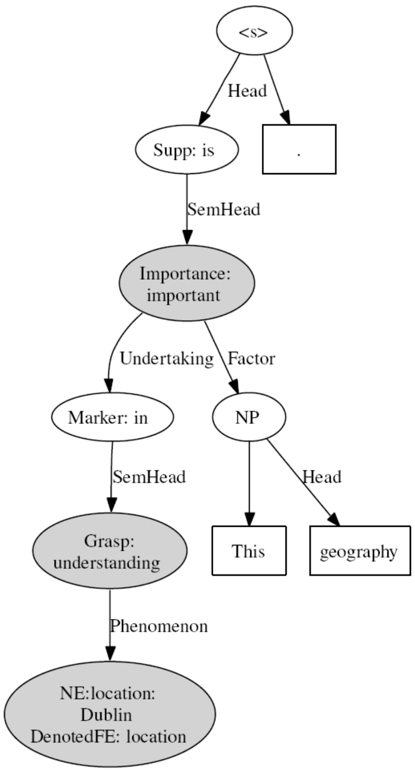


Figure 5.6: *Semantic dependency graph representation used in the FSSE task.*

In addition, there were strictness options in the evaluation of labeling. In the *hard* evaluation metric, a label was either completely correct or incorrect. In the *soft* evaluation metric, a frame or semantic role label could be counted as "partially" correct, depending on the distance in the FrameNet ontology.

5.4.2 Evaluation in the CoNLL-2008 Shared Task

In the CoNLL-2008 Shared Task on joint analysis of syntactic and semantic dependencies, the evaluation metric scored the performance of syntactic dependency parsing and semantic analysis. The global measure by which the systems were ranked was the average of the syntactic and semantic scores.

For the evaluation of semantic role annotation, predicate–argument structures were conceptually represented as in Figure 5.1: A predicate link whose label is the sense identifier, and labeled argument links from the predicate to its arguments. Predicate and argument links were equally weighted in scoring.

5.4.3 Evaluating Dependency-based Systems with Segment Metrics

For the FSSE evaluation, the annotation of semantic role structures consisted of labeled segments. A naïve method to create argument segments given an argument dependency node would be to create a set of segments for all contiguous word groups in the yield[4] of the argument node. However, when the argument node dominates the predicate, the created segment is not compatible with the conventions used in segment-based annotation. Figure 5.7 shows such an example. In segment-based annotation, we would annotate *The man* as an argument of *saw*. In a dependency representation, we would identify *man* as the argument, but its yield would be the whole phrase *The man I saw yesterday*.

Note, however, that on a conceptual level this is a trivial non-issue that does not affect the interpretation of the sentence. Regardless of bracketing, the semantics of the sentence is roughly the following (using VerbNet semantic labels):

$$\exists x, e : \text{man}(x) \wedge \textsc{See}(e) \wedge \textsc{Experiencer}(e, \text{I}) \wedge \textsc{Stimulus}(e, x) \wedge$$
$$\wedge \textsc{Time}(e, \text{Yesterday}) \wedge \text{painter}(x)$$

Algorithm 5.3 shows how the segments are constructed from the argument dependency nodes. For each argument node, the algorithm computes the yield Y, the set of dependency nodes to include in the bracketing. This set is then partitioned into contiguous parts, which are then converted into segments. In most cases, the yield is just the subtree dominated by the argument node. However, if the argument dominates

[4]The yield of a dependency node n is the set $Y = \{d : n \text{ dominates } d\}$.

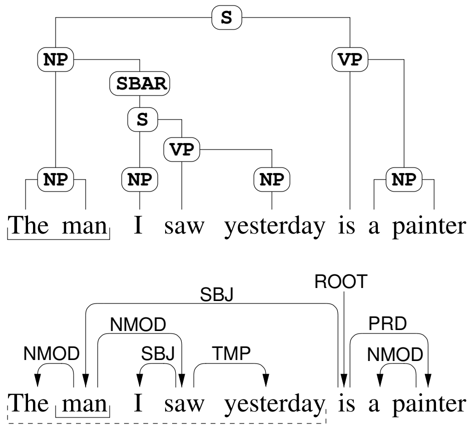

S
NP
SBAR
S
VP
VP
NP
NP
NP
NP
The man I saw yesterday is a painter
ROOT
SBJ
NMOD
PRD
NMOD
SBJ
TMP
NMOD
The man I saw yesterday is a painter

Figure 5.7: *Example of a situation where an annotated semantic role segment does not match the yield of any dependency node.*

the predicate, then the branch containing the predicate is removed. Also, FrameNet allows arguments to coincide with the predicate; in this case, the yield is just the predicate node.

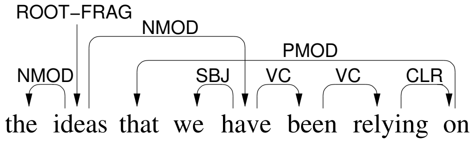


Figure 5.8: *Example of a dependency tree containing a predicate* relying *with three arguments:* the ideas, we, *and* on ... that.

To illustrate Algorithm 5.3, consider Figure 5.8. In this sentence, the predicate *relying* has three arguments: *the ideas*, *we*, and *on ... that*. The simplest of them is *we*, which does not dominate its predicate and which is not discontinuous. A more complex case is the discontinuous argument headed by *on*, where the yield $\{on, that\}$ is partitioned into two subsets that result in two separate segments. Finally, the dependency node *ideas* dominates the predicate. In this case, the algorithm removes the subtree headed by *have*, so the remaining yield is $\{the, ideas\}$.

Algorithm 5.3 Segment creation from argument dependency nodes.

```
function CREATE-SEGMENT($p, a, T$)
input Predicate node $p$, argument node $a$, dependency tree $T$
  if $a$ does not dominate $p$ in $T$
    $Y \leftarrow \{n; a \text{ dominates } n\}$
  else if $p = a$
    $Y \leftarrow \{p\}$
  else
    $c \leftarrow$ the dependent of $a$ that dominates $p$
    $Y \leftarrow \{n; a \text{ dominates } n\} \setminus \{n; c \text{ dominates } n\}$
  end if
  $S \leftarrow$ partition of $Y$ into contiguous subsets
  return $\{(\text{min-index } s, \text{max-index } s); s \in S\}$
```

5.5 Results

This section describes the results of the evaluations.

5.5.1 Results in FSSE

The FSSE system, processed using Algorithm 5.3 to give segment output, was evaluated on three unseen texts.

Table 5.5 shows the results for frame detection averaged over the test texts. The strictness column indicates whether hard or soft frame matching was used by the evaluation script, and the metric column whether a segment or dependency metric was used. Our system outperformed the second-best participant in the evaluation, the constituent-based system by Bejan and Hathaway (2007), on all metrics.

Strictness	Metric	Recall	Precision	$F1$
Hard	Segment	0.528	0.688	0.597
Soft	Segment	0.581	0.758	0.657
Hard	Dependency	0.549	0.715	0.621
Soft	Dependency	0.601	0.784	0.681

Table 5.5: *Results for frame detection in the FSSE task.*

Table 5.6 shows the importance of a proper implementation of the predicate identifier and classifier. The hard segment metric is used in these evaluations. For the predicate identifier, we compare a system using a rule-based filter with one that marks all lemmas listed in FrameNet as predicates. We see that the filter lowers recall very slightly, while the precision is greatly improved. For the frame assignment problem, we see that when a classifier is used, the system performs much better than when the most common sense is always picked.

Pred. id.	Frame assignment	Recall	Precision	$F1$
Filtering	Classifier	0.528	0.688	0.597
Filtering	First sense	0.457	0.596	0.517
No filtering	Classifier	0.535	0.488	0.510
No filtering	First sense	0.461	0.420	0.440

Table 5.6: *Effect of filtering rules and sense disambiguation on frame detection results.*

Finally, Table 5.7 shows the results for the evaluation of complete annotation. Our system had the top scores in this evaluation as well. However, the scores for all systems were low, reflecting the complexity of the task and the difficulty of the test set.

Interestingly, the scores are higher with the dependency metric than with the segment metric, while the other teams generally had higher scores with the segment metric. We believe that the reason for this is that we used a dependency parser, and that the rules that we used to convert dependency nodes into segments may have produced some errors. It is possible that the figures would have been slightly higher if our program produced semantic dependency graphs directly.

Strictness	Metric	Recall	Precision	$F1$
Hard	Segment	0.364	0.530	0.432
Soft	Segment	0.391	0.570	0.464
Hard	Dependency	0.384	0.561	0.456
Soft	Dependency	0.411	0.600	0.488

Table 5.7: *Results for the complete system in the FSSE task.*

5.5.2 CoNLL-2008 Results

We evaluated the CoNLL-2008 baseline system on WSJ section 23 and a part of section K of the Brown corpus. We used the standard evaluation script for the shared task. Table 5.8 shows the result of the semantic part of the evaluation. In addition to the conventional precision, recall, and F1 measures, the table also shows the number of perfectly detected propositions (Prop).

Corpus	P	R	F1	Prop
WSJ	81.32	82.32	81.81	52.67
Brown	67.40	69.99	68.67	33.88
WSJ+Brown	79.77	80.98	80.37	50.64

Table 5.8: *Results on the CoNLL-2008 corpora.*

These figures are higher than for all participating systems in the CoNLL-2008 Shared Task except the best-performing (which will be described in Chapter 7).

Chapter 6

Comparing Syntactic Representations for Automatic Role-semantic Analysis

In Chapter 5, we described a dependency-based semantic role labeler that achieved a good result in two evaluations. However, a comparison with previously published results may not be very informative, since results are also influenced by menial engineering and optimization details. In this chapter, we carry out an evaluation that we believe is more fair: We implement dependency- and constituent-based semantic role labelers that are designed to be as similar as possible, and we evaluate them on a number of different parsers. While the purpose of the previously described system was to demonstrate the feasibility of dependency-based semantic role analysis, the current chapter instead aims to understand the impact of the syntactic representation on the subtasks of semantic role labeling. Additionally, using the same experimental setup, we evaluate the effect of the design of the dependency representation that we described in Chapter 4.

6.1 Experimental Setup

To study the influence of syntactic representation on SRL performance, we developed a framework that could be easily parametrized to process either constituent or dependency input[1]. This section describes its implementation. As the role-semantic paradigm, we used FrameNet (Baker, Fillmore, and Lowe, 1998).

6.1.1 Systems

We applied the SRL architecture on the output of six different parsers. All parsers were trained on sections 02–21 of the WSJ part of the Penn Treebank, either directly for the constituent parsers or through the constituent-to-dependency converter described in Chapter 4. The resulting SRL systems are identified as follows:

LTH Parser. A dependency-based system using the parser described in 7.3.

MaltParser. A dependency-based system using MaltParser (Nivre et al., 2007).

MSTParser. A dependency-based system using MSTParser (McDonald, Crammer, and Pereira, 2005).

C&J Reranker. A constituent-based system using the reranking parser (the May 2006 version) by Charniak and Johnson (2005).

Charniak. A constituent-based system using Charniak's parser (Charniak, 2000).

Collins. A constituent-based system using Collins' parser (Collins, 1997).

MaltParser is an incremental greedy classifier-based parser based on support vector machines, while MSTParser use exact edge-factored search with a linear model trained using an online learning algorithm. MaltParser and MSTParser have achieved state-of-the-art results for a wide range of languages in the 2006 and 2007 CoNLL Shared Tasks on dependency parsing. Charniak's and Collins' parsers are widely used constituent parsers for English, and the C&J reranker is the best-performing publicly available constituent parser at the time of writing according to published figures. Charniak's parser and the C&J reranker

[1]The implementation is available for download at the following web page: `http://nlp.cs.lth.se/fnlabeler`.

come with a built-in part-of-speech tagger; all other systems used the Stanford tagger (Toutanova et al., 2003).

Following Gildea and Jurafsky (2002), the SRL problem is traditionally divided into two subtasks: identifying the arguments and labeling them with semantic roles. We did not consider predicate identification and disambiguation in the experiments described in this chapter. Although state-of-the-art SRL systems use sophisticated statistical models to perform the two subtasks jointly, as described in Chapter 7, we implemented them as two independent support vector classifiers as in Chapter 5 to be able to analyze the impact of syntactic representation on each subtask separately.

Table 6.1 enumerates the features used by the classifiers. For a description of the features used in the dependency trees, see 5.3.5. The differences in the feature sets reflect the structural differences between constituent and dependency trees: The constituent-only features are based on phrase tags and the dependency-only features on grammatical functions labels.

Features	Argument identification	Argument classification
PREDLEMMA	C,D	C,D
PREDPOS	C,D	C,D
VOICE	C,D	C,D
POSITION	C,D	C,D
ARGWORD/POS	C,D	C,D
LEFTWORD/POS	C,D	C,D
RIGHTWORD/POS	C,D	C,D
PREDGOVWORD/POS	C,D	
CONSTITUENTSUBCAT	C	C
CONSTITUENTPATH	C	C
PHRASETYPE	C	C
GOVCAT	C	C
RELPATH	D	D
DEPLABELS	D	D
CONTROLLERHASOBJ	D	
PREDREL	D	
FUNCTION		D

Table 6.1: *Classifier features. The features used by the constituent-based and the dependency-based systems are marked C and D, respectively.*

6.1.2 Features in Constituent-based SRL Systems

In the constituent-based SRL systems, we used a number of features that are not directly definable in a dependency-based system. The examples refer to Figure 6.1.

CONSTITUENTSUBCAT. Subcategorization frame: corresponds to the phrase-structure rule used to expand the phrase around the target. For *give* in the example, this feature is `VP→VB NP NP`.

CONSTITUENTPATH. A string representation of the path through the constituent tree from the target word to the argument constituent. For instance, the path from *gave* to *she* is ↑`VP`-↑`S`-↓`NP`.

PHRASETYPE. Phrase type of the argument constituent, e.g. `NP` for *she*.

GOVCAT. Governing category: this feature is either `S` or `VP`, and is found by starting at the argument constituent and moving upwards until either a `VP` or a sentence node (`S`, `SINV`, or `SQ`) is found. For instance, for *she*, this feature is `S`, while for *the horse*, it is `VP`. This can be thought of as a very primitive way of distinguishing subjects and objects.

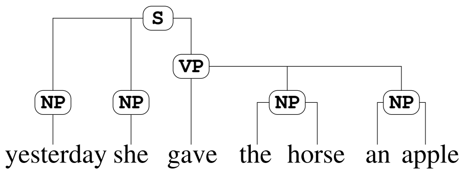


Figure 6.1: *Example to illustrate features in constituent-based SRL.*

6.2 Comparison of Constituents and Dependencies for SRL

We carried out a number of experiments to compare the influence of the syntactic representation on different aspects of SRL performance. We used the FrameNet example corpus and running-text corpus, from which we randomly sampled a training and test set. The training set consisted of 134,697 predicates and 271,560 arguments, and the test set of 14,952 predicates and 30,173 arguments. This does not include null-instantiated arguments, which were removed from the training and test sets.

6.2.1 Argument Identification

Before evaluating the full automatic argument identification systems, we studied the effect of the span creation from dependency nodes (Algorithm 5.3). To do this, we measured the upper-bound recall of argument identification using the conventional span-based evaluation metric. We compared the quality of *pruned* spans (Algorithm 5.3) to *unpruned* spans (a baseline method that brackets the full subtree). Table 6.2 shows the results of this experiment. The figures show that proper span creation is essential when the traditional metrics are used: For all dependency-based systems, the upper-bound recall increases significantly. However, the dependency-based systems generally have lower figures for the upper-bound recall than constituent-based ones.

System	Pruned	Unpruned
LTH	83.9	82.1
Malt	82.1	80.2
MST	80.4	77.1
C&J Reranker	85.3	
Charniak	83.4	
Collins	81.8	

Table 6.2: *Upper-bound recall for argument identification.*

Our first experiment investigated how the syntactic representation influenced the performance of the argument identification step. Table 6.3 shows the result of this evaluation. As can be seen, the constituent-based systems outperform the dependency-based systems on average. However, the picture is not clear enough to draw any firm conclusion about a fundamental structural difference. There are also a number of reasons to be cautious: First, the dependency parsers were trained on a treebanks that had been automatically created from a constituent treebank, which probably results in a slight decrease in annotation quality. Second, while dependency parsing has advanced during recent years, constituent parsing is still at least one generation ahead: the best constituent parser (C&J) is a reranking parser utilizing global features, while the dependency parsers use local features only; we believe that a reranker could be used to improve the dependency parsers as well.

Differences between parsers using the same syntactic formalism are also considerable, which suggests that the attachment accuracy is

System	P	R	F1
LTH Parser	79.7	77.3	78.5
MaltParser	77.4	73.8	75.6
MSTParser	73.9	71.9	72.9
C&J Reranker	81.4	77.3	79.2
Charniak	79.8	75.0	77.3
Collins	78.4	72.9	75.6

Table 6.3: *Argument identification performance based on type of formalism.*

System	Accuracy
LTH Parser	89.6
MaltParser	88.5
MSTParser	88.1
C&J Reranker	88.9
Charniak	88.5
Collins	88.3

Table 6.4: *Semantic role classification accuracy.*

possibly the most important parameter when choosing a parser for this task.

6.2.2 Argument Classification

To evaluate the argument classification accuracies, we provided the systems with gold-standard arguments, which were then automatically classified. Table 6.4 shows the results.

Here, the situation is different: the best dependency-based system make 6.3% fewer errors than the best constituent-based one, a statistically significant difference at the 99.9% level according to a McNemar test. Apart from this outlier, there are no clear differences that can be attributed to syntactic formalism. However, this result is positive, because it shows clearly that SRL can be carried out in situations where only dependency parsers are available.

On the other hand, it may seem paradoxical that the rich set of grammatical functions used by the dependency-based systems did not lead to a clearer difference between the groups, despite the linguistic intuition that this feature should be useful for argument classification.

Especially for for the second- and third-best systems (Malt and MST versus Charniak and Collins), the performance figures are almost identical. However, all systems use lexical features of the argument, and we saw in 5.3.6 that these features are very important for argument classification. Given enough training data, one may say that the grammatical function is implicitly encoded in these features. This suggests that lexical features are more important for constituent-based systems than for dependency-based ones.

6.2.3 Robustness of SRL Classifiers

In this section, we test the hypothesis that the SRL systems based on dependency syntax rely less heavily on lexical features. We also investigate two parameters that are influenced by lexicalization: domain sensitivity and the amount of training data required by classifiers.

Tests of Unlexicalized Models

To test the hypothesis about the reliance on lexicalization, we carried out a series of experiments where we set aside the lexical features of the argument in the argument classifier. Table 6.5 shows the results.

As expected, there is a sharp drop in performance for all systems, but the results are very clear: When no argument lexical features are available, the dependency-based systems have a superior performance. The difference between MST and C&J constitutes an error reduction of 6.9% and is statistically significant at the 99.9% level.

System	Accuracy
LTH Parser	83.0
MaltParser	81.9
MSTParser	81.7
C&J Reranker	80.3
Charniak	80.0
Collins	79.8

Table 6.5: *Accuracies for unlexicalized role classifiers.*

Training Set Size

Since the dependency-based systems rely less on lexicalization, we can expect them to have a steeper learning curve. To investigate this, we trained semantic role classifiers using training sets of varying sizes and compared the average classification accuracies of the two groups. Figure 6.2 shows the reduction in classification error of the dependency-based group compared to the constituent-based group (again, all systems were lexicalized). For small training sets, the differences are large; the largest observed error reduction was 5.4% with a training set of 25,000 instances. When the training set size increases, the difference between the groups decreases. The plot is consistent with our hypothesis that the grammatical function features used by the dependency-based systems make generalization easier for statistical classifiers. We interpret the flatter learning curves for constituent-based systems as a consequence of lexicalization – these systems need more training data to use lexical information to capture grammatical function information implicitly.

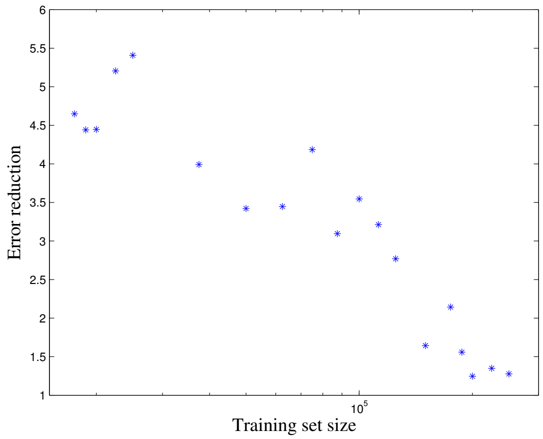


Figure 6.2: *Error reduction of average dependency-based systems as a function of training set size.*

Out-of-domain Test Sets

We finally conducted an evaluation of the semantic role classification accuracies on an out-of-domain test set: the FrameNet-annotated Nuclear Threat Initiative texts from SemEval task (Baker, Ellsworth, and Erk, 2007). Table 6.6 shows the results. This corpus contained 9,039 predicates and 15,343 arguments. The writing style is very different from the FrameNet training data, and the annotated data contain several instances of predicates and frames unseen in the training set. We thus see that all systems suffer severely from domain sensitivity, but we also see that the dependency-based systems are more resilient – the difference between MST and C&J is statistically significant at the 97.5% level and corresponds to an error reduction of 2%. The experiment reconfirms previous results (Carreras and Màrquez, 2005) that the argument classification part of SRL systems is sensitive to domain changes, and Pradhan, Ward, and Martin (2008) argued that an important reason for this is that the lexical features are heavily domain-dependent. Our results are consistent with this hypothesis, and suggest that the inclusion of grammatical function features is an effective way to mitigate this sensitivity.

System	Accuracy
LTH Parser	71.1
MaltParser	70.1
MSTParser	70.1
C&J Reranker	69.5
Charniak	69.3
Collins	69.3

Table 6.6: *Classification accuracy on the NTI texts.*

6.3 Effect of the Design of Dependency Representation

While the previous section compared the constituent and dependency formalisms with respect to SRL performance, this section investigates how SRL is affected by the structure of the dependency representation, which we discussed at great length in Chapters 3 and 4. We now apply the dependency-based SRL system evaluated in the previous section on the output of MaltParser in two different dependency formalisms.

As the baseline, we used the Penn2Malt tool.[2] This constituent-to-dependency converter implements the widely used method described by Yamada and Matsumoto (2003). While Yamada's method only produces unlabeled links, Penn2Malt also outputs function labels. The set of grammatical functions contains 12 labels.

The evaluation is carried out similarly to the previous section. Starting with the upper-bound recall on the training set (Table 6.7), we see that the system using the dependency representations designed in this work (referred to as LTH Converter in the table) outperforms the baseline significantly.

System	Pruned	Unpruned
LTH Converter	82.1	80.2
Penn2Malt	81.2	79.2

Table 6.7: *Upper-bound recall for argument identification, based on dependency type.*

It should be noted that this is not because the LTH dependency structures are easier to predict for a parser; in fact, the opposite is true. Table 6.8 shows the results of a parser evaluation on section 23 of the WSJ part of the Penn Treebank. As we can see, the LTH format is more demanding for parsers, both for attachment and for function labeling.

Converter	Labeled	Unlabeled
Penn2Malt	90.3	91.4
LTH Converter	87.8	90.7

Table 6.8: *Parsing accuracy for MaltParser on WSJ section 23 using Penn2Malt and the LTH converter.*

Moving on to argument identification performance, shown in Table 6.9 we see that the system based on the LTH-processed treebank outperforms the Penn2Malt-based system soundly.

It is also interesting to note that the difference is greater than the difference in upper-bound recall. It thus seems that the modified head

[2]`http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html`

selection procedure and dependency arc labeling used by the LTH labeler result in better features for the statistical classifier.

System	P	R	F1
LTH Converter	77.4	73.8	75.6
Penn2Malt	76.1	72.1	74.0

Table 6.9: *Argument identification performance for different conversion methods.*

Similarly to what we saw in 6.2.2, we see no significant difference in the case of argument classification (Table 6.10). Again, we believe that this is due to the fact that lexicalization can compensate for an inexpressive grammatical function set if the training set is large enough. The figures in Table 6.11, which show classification accuracies when removing argument lexicalization, are consistent with this hypothesis.

System	Accuracy
LTH Converter	88.5
Penn2Malt	88.4

Table 6.10: *Semantic role classification accuracy for different conversion methods.*

System	Accuracy
LTH Converter	81.9
Penn2Malt	80.0

Table 6.11: *Unlexicalized role classification accuracy for different conversion methods.*

The benefit of relying less on lexicalization is once more evident in the plot of error reduction as a function of training set size in Figure 6.3. For clarity, the plot also includes an interpolated quadratic polynomial to show the general tendency more clearly. For small training sets, the semantic role labeler using LTH dependency structures has a higher performance.

However, in contrast with the experiment in 6.2.2, the difference in accuracy is insignificant when we move to role classification on the NTI

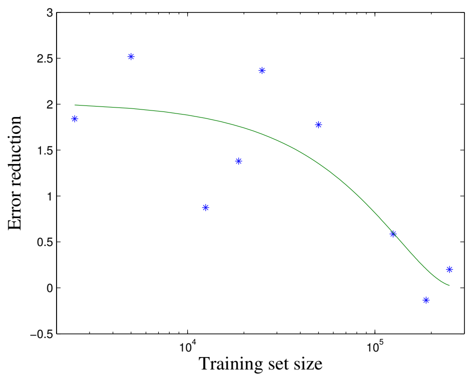


Figure 6.3: *Error reduction of the LTH-based semantic role labeler as a function of training set size.*

corpus. This result may appear slightly unexpected, but a possible explanation is that the LTH dependency structures, while generally more informative, also make the *parser* more brittle. However, it is hard to test this hypothesis empirically without a gold-standard syntactic annotation of the NTI corpus.

System	Accuracy
LTH Converter	70.1
Penn2Malt	70.0

Table 6.12: *NTI role classification accuracy for different conversion methods.*

6.4 Discussion

We have described a set of experiments investigating the relation between syntactic representation and semantic role labeling performance, specifically focusing on a comparison between constituent- and dependency-based SRL systems. The influence of the syntactic formalism on SRL has only been considered in a few previous articles. For instance, Gildea and Hockenmaier (2003) reported that a CCG-based parser gives improved results over the Collins parser. The most closely related work is by Miyao et al. (2008), who compared the effect of parsers and syntactic representation based on the performance of information extraction systems. They reported results that were very similar to ours, that constituent and dependency systems can both serve as the syntactic input for a system doing semantic processing.

We reconfirm the result from Chapter 5 that dependency-based systems can perform more or less as well as constituent-based systems. This comparison is also fairer since we designed the systems to be as similar as possible, to rule out any influence of engineering details on the result. For the argument classification task, dependency-based systems are slightly better on average, while the constituent-based systems perform slightly higher in argument identification.

Our second main result is that for argument classification, dependency-based systems rely less heavily on lexicalization, and we suggest that this is because they use features based on grammatical function labels. These features make the learning curve steeper when training the classifier, and improve robustness to domain changes.

The experiments also show that a careful design of the dependency representation, as described in Chapters 3 and 4, is important for the performance of the semantic role labeler.

Chapter 7

Extensions of the Classifier-based Model

In Chapter 5, we modeled the problem of automatic analysis of text as a *sequential* task that is solved using a greedy search procedure. We applied a dependency parser, followed by predicate identification and word sense disambiguation, then argument identification and labeling. Using this model, we were able to obtain decent results.

However, this architecture is problematic from both a linguistic and from a statistical view. It is clear that all the subtasks are highly interdependent, and it is sensible to ask whether they could be solved more accurately if the statistical model takes this into account. Intuitively, semantic interpretation should help syntactic disambiguation, and joint syntactic–semantic analysis has historically been prominent in linguistic theory; deep-linguistic formalisms such as LFG and HPSG are examples of frameworks where syntactic and semantic analysis are performed in tandem.

The chapter starts with an introduction to modern statistical methods for learning predictors of complex structured data. Introducing more complex models is a challenge from an engineering point of view, since the problem structure has to be exploited to maintain tractability.

We extend the CoNLL-2008 baseline system for PropBank and NomBank analysis described in Chapter 5 by applying these methods to the complete syntactic–semantic problem, and to the subproblems of dependency parsing and predicate–argument structure prediction, respectively. The resulting system took part in the CoNLL-2008 Shared Task, where it obtained the best result among 22 participants.

7.1 Discriminative Modeling of Non-atomic Variables

Prediction of structured data has a long tradition in NLP. The statistical models have typically been generative and employed maximum-likelihood estimation, such as probabilistic context-free grammars for constituent parsing, or hidden Markov models for sequence labeling.

The generative models had the drawback of lack of expressivity, since the features that they could model were limited. This led to the introduction of discriminative classifier-based models, which can use arbitrary features. They might be probabilistic, such as maximum-entropy Markov models (McCallum, Freitag, and Pereira, 2000) or not (Punyakanok and Roth, 2001). Classifier-based models have been very successful in a number of tasks such as sequence labeling (Kudo and Matsumoto, 2003) and dependency parsing (Nivre et al., 2007).

However, unlike generative models, classifier-based models suffer from a problem referred to as *label bias* (Bottou, 1991) – local decisions are viewed in isolation when estimating the parameters of the models, and this may lead to suboptimal performance when viewed in a larger context. Therefore, a number of approaches have been proposed to unify the two types of models. The first widely known was the *conditional random field* (Lafferty, McCallum, and Pereira, 2001), which is the generalization of logistic (or maximum entropy) models to arbitrary data, but which is typically applied in tasks that can be modeled using graphical models, such as sequence labeling.

Since then, a large number of publications on the topic have appeared. The approaches can be grouped into two broad categories:

- *Decomposition-based* methods, which assume that the non-atomic output variable y can be meaningfully decomposed into a set of atomic parts $y_1, \ldots, y_n$, which are logically and statistically correlated. Problem-specific independence assumptions are then used to make optimization tractable. This group includes the above-mentioned conditional random fields, as well as the max-margin Markov (Taskar, Guestrin, and Koller, 2004) and PCFG (Taskar et al., 2004) models. A recent efficient optimization algorithm for both types of models is the extragradient method (Collins et al., 2008).

- *Prediction-based* methods, which repeatedly asks the user to make a prediction during learning. This group includes online learning algorithms such as the generalized perceptron (Collins, 2002) and its extensions, which may be margin-based such as MIRA (Crammer and Singer, 2003) and the online passive–aggressive algorithm (Crammer et al., 2006), or probabilistic (Johansson, 2007). Batch algorithms in this group include SVM$^{\text{struct}}$ (Tsochantaridis et al., 2005).

It can be noted that when using search based on locally trained classifiers, it is to a certain extent possible to compensate for the inherent limitations of the models by using a more complex feature set (Liang, Daumé III, and Klein, 2008). For instance, this has made it possible to obtain competitive performance in a sequential labeling task such as part-of-speech tagging (Giménez and Màrquez, 2003). Another way to overcome the bias problems in classifier-based search is based on iterative retraining (Wang, Lin, and Schuurmans, 2007; Daumé III, Langford, and Marcu, 2006).

7.1.1 Online Linear Learning

In a discriminative framework, we model the prediction problem for a given input $\boldsymbol{x}$ as finding the output $\hat{y}$ from a candidate set $\mathcal{Y}$ that maximizes a function $F(\boldsymbol{x}, y)$.

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} F(\boldsymbol{x}, y)$$

The learning problem consists of finding the function F so that the cost of the predictions is as low as possible according to a *cost function* (or loss function) ρ. In this work, we consider linear scoring functions of the following form:

$$F(\boldsymbol{x}, y) = \boldsymbol{w} \cdot \boldsymbol{\Psi}(\boldsymbol{x}, y)$$

where $\boldsymbol{\Psi}(\boldsymbol{x}, y)$ is a numeric feature representation of the pair $(\boldsymbol{x}, y)$ and $\boldsymbol{w}$ a high-dimensional vector of feature weights.

A widely used framework for fitting the weight vector is the max-margin model (Taskar, Guestrin, and Koller, 2004; Taskar et al., 2004), which is a generalization of the well-known support vector machines (Boser, Guyon, and Vapnik, 1992) to general cost-based prediction problems. This approach has been shown theoretically and practically to lead to good generalization to unseen data. In this model, we search for the smallest weight vector $\boldsymbol{w}$ that satisfies the constraint that the difference in F between the correct output y_i and an incorrect output y_{ij} should be at least $\rho(y_i, y_{ij})$. For a training set $\mathcal{T} = \{\langle \boldsymbol{x}_i, y_i \rangle\}$ where the output space for the input $\boldsymbol{x}_i$ is $\mathcal{Y}_i$, we state the learning problem as a constrained quadratic optimization program:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \|\boldsymbol{w}\|^2 \\ \\ \text{subject to} & F(\boldsymbol{x}_i, y_i) - F(\boldsymbol{x}_i, y_{ij}) \geq \rho(y_i, y_{ij}), \quad \forall \langle \boldsymbol{x}_i, y_i \rangle \in \mathcal{T}, y_{ij} \in \mathcal{Y}_i \end{array}$$

In practice, the optimization problem is also regularized to reduce overfitting, which leads to the introduction of a parameter C as in SVMs.

For most realistic types of problems, the number of constraints precludes a direct solution of the optimization program. In our case, for instance, the number of constraints is equal to the number of incorrect parse trees for each sentence. There are algorithms that can handle fairly large-scale problems either by iteratively selecting constraints (Tsochantaridis et al., 2005) or by relying on problem factorization (Taskar, Guestrin, and Koller, 2004; Collins et al., 2008).

Since the large number of training examples and features in our case make an exact solution of the max-margin optimization problem impractical, we used the online passive–aggressive algorithm (Crammer et al., 2006), which simplifies the optimization process in two ways:

- The weight vector $\boldsymbol{w}$ is updated incrementally, one example at a time.
- For each example, only the most violated constraint is considered.

The algorithm is a margin-based variant of the perceptron. Algorithm 7.1 shows pseudocode for the algorithm.

As stated above, we model the prediction problem for a given input $\boldsymbol{x}$ as finding the $\hat{y}$ that maximizes a function $F(\boldsymbol{x}, y)$. Conceptually, we solve this problem by applying F to all possible outputs for the input $\boldsymbol{x}$. However, except for very small candidate sets, it is not feasible to enumerate all the possible outputs. When applying a learning method such as the passive–aggressive algorithm, we thus need to rely on the structure of the problem we are solving.

To use the passive–aggressive algorithm, the user needs to implement the following four functions.

- A feature representation function $\boldsymbol{\Psi}(\boldsymbol{x}, y)$.
- A cost function $\rho(y_i, y_j)$ that returns a numerical measure of how far the predicted output y_j is from the correct answer y_i.

Algorithm 7.1 The online passive–aggressive algorithm

```
function PASSIVE–AGGRESSIVE($\mathcal{T}, N, C$)
input Training set $\mathcal{T} = \{(\boldsymbol{x}_t, y_t)\}_{t=1}^{T}$
      Number of iterations $N$
      Regularization parameter $C$
  Initialize $w$ to zeros
  repeat $N$ times
    for $(\boldsymbol{x}_t, y_t)$ in $\mathcal{T}$
      let $\tilde{y}_t = \arg\max_y F(\boldsymbol{x}_t, y) + \rho(y_t, y)$
      let $\tau_t = \min\left(C, \frac{F(\boldsymbol{x}_t,\tilde{y}_t)-F(\boldsymbol{x}_t,y_t)+\rho(y_t,\tilde{y}_t)}{\|\boldsymbol{\Psi}(\boldsymbol{x},y_t)-\boldsymbol{\Psi}(\boldsymbol{x},\tilde{y}_t)\|^2}\right)$
      $\boldsymbol{w} \leftarrow \boldsymbol{w} + \tau_t(\boldsymbol{\Psi}(\boldsymbol{x}, y_t) - \boldsymbol{\Psi}(\boldsymbol{x}, \tilde{y}_t))$
  return $\boldsymbol{w}_{\text{average}}$
```

- An optimizer that finds $\hat{y} = \arg\max_y F(\boldsymbol{x}_t, y) + \rho(y_t, y)$ during learning.
- A predictor that finds $\hat{y} = \arg\max_y F(\boldsymbol{x}_t, y)$ at test time.

7.2 Predicting Syntactic–Semantic Dependency Graphs

As we argued in the introduction to the chapter, syntax and semantics are interdependent and it is possible that syntactic and semantic analysis could be improved by integrating the two steps. In the framework described in 7.1, we model the problem of finding a syntactic tree $\hat{y}_{syn}$ and a semantic graph $\hat{y}_{sem}$ for a sentence $\boldsymbol{x}$ as maximizing a function F_{joint} that scores the joint syntactic–semantic structure:

$$\langle \hat{y}_{syn}, \hat{y}_{sem} \rangle = \arg\max_{y_{syn}, y_{sem}} F_{joint}(\boldsymbol{x}, y_{syn}, y_{sem})$$

7.2.1 Overview of the Problem and Previous Work

We can view the problem of joint syntactic and semantic analysis as a special case of multi-governor dependency parsing. Unfortunately, the optimization problem of maximizing F_{joint} is intractable in this case. If the number of heads is not bounded but we assume that the graph is cycle-free, the problem is equivalent to FEEDBACK ARC SET, which was one of Karp's original 21 NP-complete problems (Karp, 1972).[1] Even if we assume a bound on the number of heads – which is true in our case since there can only be one semantic link per predicate – the problem is NP-hard (Chickering, Geiger, and Heckerman, 1994). In the literature, there have been a few methods for multi-governor dependency parsing. For instance, McDonald and Pereira (2006) used a greedy search starting from the optimal single-governor tree, and Sagae and Tsuji (2008) applied a greedy shift–reduce algorithm.

However, formulating the joint syntactic–semantic analysis problem as multi-governor dependency parsing is too general, since the predicate–argument structures are unnested. This fact was exploited by Lluís and Màrquez (2008), who extended Eisner's search method (Eisner, 1996) for projective dependency trees to simultaneously predict syntactic and semantic dependency structures.

On the other hand, even with the restrictions on the form of the semantic dependency graphs, exact search is clearly intractable if we want to model the interaction between syntax and semantics realistically. As we saw in Chapter 5, the path through the dependency tree from the predicate to the argument is the most important feature. Since semantic relations can be non-local, the path feature is too complex to be used in exact search. Lluís and Màrquez (2008) used the output of a dependency parser to be able to extract this feature.

Except for the above-mentioned work by Lluís and Màrquez (2008), some interesting approximations of the joint syntactic–semantic search problem that were used in the CoNLL-2008 Shared Task include an incremental shift–reduce algorithm (Henderson et al., 2008) that maintains two separate stacks for syntactic and semantic dependencies, respectively. Another work based on shift–reduce parsing is Samuelsson et al. (2008), who added semantic information to the syntactic labels. This output was also combined with the output of a conventional classifier-based semantic role labeler in a "blending" step.

The problem of joint syntactic–semantic analysis has of course also been described before the CoNLL-2008 Shared Task. The pioneering work by Gildea and Jurafsky (2002), which anticipated many of the issues later explored in more detail by others, attempted to integrate the Collins constituent parser with a semantic role labeler, but failed to improve significantly over the original sequential method. A simpler approach to the problem that has been attempted is to add semantic la-

[1]To make matters worse, Kann (1992) showed that this problem is APX-hard, meaning that that there is a constant k, such that there is no polynomial-time approximation algorithm that always find a solution at most k times more costly than the optimal result.

bels to constituent labels (Musillo and Merlo, 2006; Merlo and Musillo, 2008). Finally, the work by Collobert and Weston (2008), which jointly predicts chunks, named entities, and semantic role segments, also needs to be mentioned.

7.2.2 Our Method

The intractability of the problem, which we described in the previous section, forces us to resort to a simplification. We chose a strategy based on *reranking*: A candidate pool is generated by simple subsystems, while the final output is selected from the candidate pool by a complex model. This allows us to have arbitrarily complex features in the scoring function F_{joint}, since we just apply it to every candidate. The reranking method is commonly used in speech recognition (Schwartz and Austin, 1993), and has also recently been applied in constituent parsing (Collins, 2000; Charniak and Johnson, 2005), where it has led to some of the best published results.

To generate the candidate pool, we used the following strategy:

- In a first step, a *syntactic submodel*, a dependency parser, generates a m-best list of syntactic trees.

- In a second step, a *semantic submodel* generates n semantic analyses for every syntactic tree.

The following two sections describes the submodels, and we will return to the integration step in 7.5.

7.3 The Syntactic Submodel

To implement the syntactic submodel, we once again applied the conceptual framework and design pattern described in 7.1: The process of syntactic parsing of a sentence $\boldsymbol{x}$ is modeled as finding the parse tree $\hat{y}_{syn} = \arg\max_{y_{syn}} F_{syn}(\boldsymbol{x}, y_{syn})$ that maximizes a scoring function F_{syn}. This is the well-known discriminative model for dependency parsing previously described by McDonald et al. (2005), *inter alia*.

We trained this model using Algorithm 7.1. The value of C was 0.01, and we defined the syntactic cost ρ_{syn} as the sum of link costs, where the link cost was 0 for a correct dependency link with a correct label, 0.5 for a correct link with an incorrect label, and 1 for an incorrect link.

7.3.1 Features and Search

The feature function $\boldsymbol{\Psi}_{syn}$ is a factored representation, meaning that we compute the score of the complete parse tree by summing the scores of its parts, referred to as *factors*:

$$\boldsymbol{\Psi}_{syn}(\boldsymbol{x}, y_{syn}) \cdot \boldsymbol{w} = \sum_{f \in \text{factors}(y_{syn})} \psi(\boldsymbol{x}, f) \cdot \boldsymbol{w}$$

The advantage of this decomposition is that although the number of parse trees is superexponential[2] in the number of words in the input sentence, the number of possible factors is just polynomial. Since the scoring function is the sum of factor scores, we can apply dynamic programming to find the optimal parse tree.

We used a second-order factorization (McDonald and Pereira, 2006; Carreras, 2007), meaning that the factors are subtrees consisting of four links: the governor–dependent link, its sibling link, and the leftmost and rightmost dependent links of the dependent. Figure 7.1 shows an example of a factor.

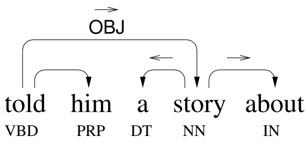


Figure 7.1: *Second-order factor in the dependency parser.*

Tables 7.1 and 7.2 show the first-order and second-order feature sets. These feature sets are taken from McDonald et al. (2005) and Carreras (2007), respectively. The abbreviations used in the tables are explained in Table 7.3. Note that all features are implicitly combined with the LABEL and DIRECTION features.

This factorization allows us to express useful features, but also forces us to adopt the expensive search procedure by Carreras (2007), which extends Eisner's span-based dynamic programming algorithm (1996) to allow second-order feature dependencies. This algorithm has a time complexity of $O(n^4)$, where n is the number of words in the

[2]For a sentence with n words, the number of unlabeled dependency trees is given by $(n+1)^{n-1}$ (Cayley's formula). If we consider only projective trees, then the number of trees is equal to $\frac{1}{2n+1}\left(\begin{array}{c}3n\\n\end{array}\right)$ (for a proof, see Yuret (1998), *inter alia*).

Unigram	Bigram	Context
gw	gw+gp+dw+dp	{gp+bp+dp}
gp	gp+dw+dp	gp+gp$_{+1}$+dp$_{-1}$+dp
gw+gp	gw+dw+dp	gp$_{-1}$+gp+dp$_{-1}$+dp
dw	gw+gp+dp	gp+gp$_{+1}$+dp+dp$_{+1}$
dp	gw+gp+dw	gp$_{-1}$+gp+dp+dp$_{+1}$
dw+dp	gw+dw	
	gp+dp	

Table 7.1: *First-order features used in the syntactic submodel*

Sibling	Grandchild
gp+dp+sp	gp+dp+gcp+gcd
gp+sp	gp+gcp+gcd
dp+sp	dp+gcp+gcd
gw+sw	gw+gcw+gcd
dw+sw	dw+gcw+gcd
gp+sw	gp+gcw+gcd
dp+sw	dp+gcw+gcd
gw+sp	gw+gcp+gcd
dw+sp	dw+gcp+gcd

Table 7.2: *Second-order features used in the syntactic submodel*

sentence. The search was constrained to disallow multiple root links. To speed up computation, we also pruned the search space based on part-of-speech tag pairs: A dependency link $g \xrightarrow{l} d$ was considered only if a link $g' \xrightarrow{l} d'$ in the same direction had been observed in the training treebank, where g' and g belong to the same part-of-speech category, and the same for d' and d. Simplified pseudocode for the search algorithm is given in Appendix B.

To evaluate the $\arg\max$ in Algorithm 7.1 during training, we need to handle the cost function ρ_{syn} in addition to the factor scores. Since the cost function ρ_{syn} is based on the cost of single links, this can easily be integrated into the factor-based search.

The m-best syntactic parsing algorithm, needed to generate the candidate pool for the complete system, is a very simple implementation that maintains an m-best set at every entry in the dynamic programming table. This is not efficient – it increases parsing time by a factor $m^2 \log m$ – but we did not have time to implement the smart

Abbreviation	Explanation
gw	governor word
gp	governor part-of-speech tag
dw	dependent word
dp	dependent part-of-speech tag
bp	part-of-speech tag between gov. and dep.
gp_{+1}	part-of-speech tag right of governor
sw	sibling word
sp	sibling part-of-speech tag
gcw	grandchild word
gcp	grandchild part-of-speech tag
gcd	grandchild direction

Table 7.3: *Abbreviations of feature names.*

m-best generation algorithm by Huang and Chiang (2005).

7.3.2 Handling Nonprojective Links

Although only 0.4% of the links in the training set are nonprojective, 7.6% of the sentences contain at least one nonprojective link. Many of these links represent long-range dependencies – such as *wh*-movement – that are valuable for semantic processing. Nonprojectivity cannot be handled by span-based dynamic programming algorithms. For parsers that consider features of single links only, the Chu–Liu/Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967) can be used instead. However, this algorithm cannot be generalized to the second-order setting – McDonald and Pereira (2006) proved that this problem is NP-hard, and described an approximate greedy search algorithm.

To simplify implementation, we instead opted for the pseudo-projective approach (Nivre and Nilsson, 2005), in which nonprojective links are lifted upwards in the tree to achieve projectivity, and special trace labels are used to enable recovery of the nonprojective links at parse time.

The use of trace labels in the pseudo-projective transformation leads to a proliferation of edge label types: from 69 to 234 in the training set, many of which occur only once. Since the running time of our parser depends on the number of labels, we used only the 20 most frequent trace labels.

7.4 The Semantic Submodel

The semantic submodel had three main subcomponents, as mentioned previously.

- The pipeline of local logistic classifiers described in Chapter 5.
- A set of linguistically motivated constraints.
- A reranking model that scores complete propositions (predicates with arguments)

To have input for the second and third step, a candidate set was needed. We generated this set by a simple beam search procedure (using a width of 4) based on the output scores from each of the four steps in the classifier pipeline. Since our classifiers were logistic, their output values could be meaningfully interpreted as probabilities, which allowed us to combine the scores from subclassifiers into a score for the complete propositions.

7.4.1 Linguistically Motivated Global Constraints

The following three global constraints were used to filter the candidates generated by the pipeline.

CORE ARGUMENT CONSISTENCY. Core argument labels must not appear more than once.

DISCONTINUITY CONSISTENCY. If there is a label C-X, it must be preceded by a label X.

REFERENCE CONSISTENCY. If there is a label R-X and the label is inside a relative clause, it must be preceded by a label X.

7.4.2 Proposition Reranker

Toutanova, Haghighi, and Manning (2005) have shown that a global model that scores the complete proposition can lead to substantial performance gains. We therefore created a proposition reranker using the following global features in addition to the features from the pipeline:

CORE ARGUMENT LABEL SEQUENCE. The complete sequence of core argument labels. The sequence also includes the predicate and voice, for instance `A0`+*break.01*/Active+`A1`.

MISSING CORE ARGUMENT LABELS. The set of core argument labels declared in the PropBank frame that are not present in the proposition.

Similarly to the syntactic submodel, we trained the global SRL model using the online passive–aggressive algorithm. The cost function ρ was defined as the number of incorrect links in the proposition. The number of iterations was 20 and the regularization parameter C was 0.01. Interestingly, we noted that the proposition reranker outperformed the pipeline even when it used no global features. This shows that the global learning model can correct label bias problems introduced by the pipeline architecture.

7.5 Integrating Syntactic and Semantic Analysis

As outlined in 7.2.2, when the candidate pool has been generated by the two subsystems, we can apply a joint model

$$F_{joint}(\boldsymbol{x}, y_{syn}, y_{sem}) = \boldsymbol{w} \cdot \boldsymbol{\Psi}_{joint}(\boldsymbol{x}, y_{syn}, y_{sem})$$

Our baseline joint feature representation $\boldsymbol{\Psi}_{joint}$ contained only three features:

- The log probability of the syntactic tree, $\log P_{syn}(y_{syn}|\boldsymbol{x})$
- The log probability of the semantic structure according to the classifier pipeline, $\log P_p(y_{sem}|\boldsymbol{x}, y_{syn})$
- The log probability of the semantic structure according to the proposition reranker, $\log P_r(y_{sem}|\boldsymbol{x}, y_{syn})$

Figure 7.2 shows an overview of the complete system. The three arrows entering the syntactic–semantic reranker represent the three features in $\boldsymbol{\Psi}_{joint}$.

The probabilities from the syntactic model and the proposition reranker were obtained using the multinomial logistic function ("softmax").

$$P(y|\boldsymbol{x}) = \frac{e^{F(y,\boldsymbol{x})}}{\sum_i e^{F(y_i,\boldsymbol{x}))}}$$

Normalizing the probabilities with respect to the sentence length or number of predicates did not result in any measurable effect.

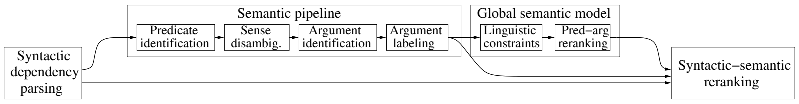


Figure 7.2: *Overview of the extended semantic role labeling system.*

To find the three weights in $\boldsymbol{w}$, we trained the model on the complete training set using cross-validation.

We carried out an initial experiment with a more complex joint feature representation, but failed to improve over the baseline. Time prevented us from exploring this direction conclusively.

7.5.1 Evaluation in the CoNLL-2008 Shared Task

The system participated in the CoNLL-2008 Shared Task. This section describes the results.

Overall Score

The submitted results on the development and test corpora are presented in the upper part of Table 7.4. The columns in the table are syntactic labeled accuracy, semantic labeled F1, macro-averaged syntactic and semantic F1, complete proposition F1, and number of perfectly analyzed sentences.

After the submission deadline, we corrected a bug in the predicate identification method. This resulted in improved results shown in the lower part of the table.

Corpus	Syn. acc.	Sem. F1	Macro F1	Prop.	Perfect
Test WSJ	90.13	81.75	85.95	54.12	12.46
Test Brown	82.81	69.06	75.95	36.90	12.68
Test WSJ + Brown	89.32	80.37	84.86	54.12	12.46
Test WSJ	90.13	83.09	86.61	57.34	13.12
Test Brown	82.84	69.85	76.34	37.11	13.15
Test WSJ + Brown	89.32	81.65	85.49	55.15	13.10

Table 7.4: *Results of our system in the CoNLL-2008 Shared Task (post-deadline bugfix results lower).*

Our results were the highest in the closed challenge of the shared task. Table 7.5 shows the results of the second-best system (Che et al., 2008).

Corpus	Syn. acc.	Sem. F1	Macro F1	Prop.	Perfect
Test WSJ	87.51	80.00	83.78	48.05	10.37
Test Brown	80.73	66.37	73.57	30.90	11.50
Test WSJ + Brown	86.75	78.52	82.66	48.05	10.37

Table 7.5: *Results of the second-best system in the CoNLL-2008 Shared Task.*

Syntactic Results

Table 7.6 shows the effect of adding second-order features to the parser in terms of accuracy as well as training and parsing time on a Mac Pro, 3.2 GHz. The training times were measured on the complete training set and the parsing time and accuracies on the development set. Similarly to Carreras (2007), we see that these features have a very large impact on parsing accuracy, but also that the parser pays dearly in terms of efficiency as the search complexity increases from $O(n^3)$ to $O(n^4)$. Since the low efficiency of the second-order parser restricts its use to batch applications, we see an interesting research direction to find suitable compromises between the two approaches, for instance by sacrificing the exact search procedure.

System	Training	Parse	Labeled	Unlabeled
1st order	65 min	28 sec	85.78	89.51
2nd order	60 hours	34 min	88.33	91.43

Table 7.6: *Impact of second-order features.*

Table 7.7 shows the dependency types most affected by the addition of second-order features to the parser when ordered by the increase in F1. As can be seen, they are all verb adjunct categories, which demonstrates the effect of grandchild features on PP attachment and labeling.

Label	ΔR	ΔP	ΔF_1
TMP	14.7	12.9	13.9
DTV	0	19.9	10.5
LOC	7.8	12.3	9.9
PRP	12.4	6.7	9.6
DIR	5.9	7.2	6.5

Table 7.7: *Labels affected by second-order features.*

Effect of Constraints and Proposition Reranking

To assess the effect of the components in the semantic submodel, we tested their performance on the top-scoring parses from the syntactic model. Table 7.8 shows the results on the development set, WSJ section 24. The baseline system (B) consists of the SRL pipeline only, i.e. the CoNLL-2008 baseline system described in Chapter 5. Adding linguistic constraints (C) results in a more precision-oriented system with slightly lower recall, but significantly higher F1. Even higher performance is obtained when adding the proposition reranker (R).

System	P	R	F1
B	80.74	77.98	79.33
B+C	82.42	77.66	79.97
B+C+R	83.64	78.14	80.40

Table 7.8: *Extended SRL results on the top-scoring parse trees.*

Effect of Syntactic–Semantic Integration

The final experiment concerned the integration of syntactic and semantic analysis. To have a system that is not greedy, we increase the number m of syntactic trees output by the dependency parser. The system chooses the output that maximizes the joint syntactic–semantic score, based on the top m syntactic trees. Table 7.9 shows the results on the development set. We see that syntactic–semantic integration improves both syntactic accuracy and semantic F1. This holds for the constraint-based SRL system as well as for the full system.

Sem model	m	Syn acc	Sem F1	Macro F1
B+C	1	88.33	79.97	84.17
B+C	16	88.42	80.42	84.44
B+C+R	1	88.33	80.40	84.39
B+C+R	16	88.47	80.80	84.66

Table 7.9: *Effect of syntactic–semantic integration.*

7.5.2 Comparing with Segment-based Systems

While the results presented so far are high compared to other dependency-based systems taking part in the CoNLL Shared Task, it is still not clear how well our system compares against previously published systems. However, these systems used the segment-based metric described in 5.4. We thus had to use Algorithm 5.3 to convert the semantic dependency structures to labeled segments. We then applied the CoNLL-2005 segment scorer to measure the quality of our output. In addition, we disabled predicate identification since in the 2005 task, the predicates were given; similarly, we did not evaluate anything related to NomBank since the 2005 task concerned PropBank only.

Table 7.10 shows the performance figures of our system on the WSJ and Brown corpora: precision, recall, F_1-measure, and complete proposition accuracy (Prop). These figures are compared to the best-performing system in the CoNLL-2005 Shared Task (Punyakanok, Roth, and Yih, 2008), referred to as Punyakanok in the table, and the best result currently published (Surdeanu et al., 2007), referred to as Surdeanu. To validate the sanity of the segment creation algorithm, the table also shows the result of applying segment creation to gold-standard syntactic–semantic trees. We see that the two conversion procedures involved (constituent-to-dependency conversion by the CoNLL-2008 Shared Task organizers, and the dependency-to-segment conversion in Algorithm 5.3) work satisfactorily although the process is not completely lossless.

During inspection of the output, we noted that many errors arise from inconsistent punctuation attachment in PropBank/Treebank. We therefore normalized the segments to exclude punctuation at the beginning or end of a segment. The results of this evaluation is shown in Table 7.11. This table does not include the results of the Surdeanu system since we did not have access to its output.

The results on the WSJ test set clearly show that dependency-

WSJ	P	R	F1	Prop
Our system	82.22	**77.72**	79.90	**57.24**
Punyakanok	82.28	76.78	79.44	53.79
Surdeanu	**87.47**	74.67	**80.56**	51.66
Gold standard	97.38	96.77	97.08	93.20

Brown	P	R	F1	Prop
Our system	68.79	61.87	65.15	32.34
Punyakanok	73.38	**62.93**	67.75	32.34
Surdeanu	**81.75**	61.32	**70.08**	**34.33**
Gold standard	97.22	96.55	96.89	92.79

WSJ+Brown	P	R	F1	Prop
Our system	80.50	**75.59**	77.97	**53.94**
Punyakanok	81.18	74.92	77.92	50.95
Surdeanu	**86.78**	72.88	**79.22**	49.36
Gold standard	97.36	96.75	97.05	93.15

Table 7.10: *Evaluation with unnormalized segments.*

WSJ	P	R	F1	Prop
Our system	**82.95**	**78.40**	**80.61**	**58.65**
Punyakanok	82.67	77.14	79.81	54.55
Gold standard	97.85	97.24	97.54	94.34

Brown	P	R	F1	Prop
Our system	70.84	63.71	67.09	**36.94**
Punyakanok	**74.29**	63.71	**68.60**	34.08
Gold standard	97.46	96.78	97.12	93.41

WSJ+Brown	P	R	F1	Prop
Our system	81.39	**76.44**	**78.84**	**55.77**
Punyakanok	**81.63**	75.34	78.36	51.84
Gold standard	97.80	97.18	97.48	94.22

Table 7.11: *Evaluation with normalized segments.*

based SRL systems can rival constituent-based systems in terms of performance – it clearly outperforms the Punyakanok system, and has a higher recall and complete proposition accuracy than the Surdeanu system. We interpret the high recall as a result of the dependency syntactic representation, which makes the parse tree paths simpler and thus the arguments easier to find.

For the Brown test set, on the other hand, the dependency-based system suffers from a low precision compared to the constituent-based systems. This may seem paradoxical in view of the result in 6.2.3, where we argued that the dependency-based systems were more resilient to domain changes. It is unlikely that the performance degradation stems from the syntactic parser; the increase in the number of errors of our parser is roughly the same as that reported for the Charniak parser used in the CoNLL-2005 Shared Task (Carreras and Màrquez, 2005). One possible explanation why the constituent-based systems are more robust in this respect is that they utilize a combination strategy, using inputs from two different full constituent parsers, a clause bracketer, and a chunker. Another possible reason[3] may be that the global inference used in the Punyakanok system is based on a set of hand-coded (and thus domain-independent) linguistic constraints, which may result in improved robustness. However, caution is needed when drawing conclusions from results on the Brown test set, which is only 7,585 words, compared to the 59,100 words in the WSJ test set.

7.5.3 Dependency-based Comparison

It has previously been noted (Pradhan et al., 2005b) that a segment-based evaluation may be unfavorable to a dependency-based system, and that an evaluation that scores argument *heads* may be more indicative of its true performance. We thus carried out a comparison using the evaluation script of the CoNLL-2008 Shared Task. In this evaluation method, an argument is counted as correctly identified if its head and label are correct. Note that this is not equivalent to the segment-based metric: In a perfectly identified segment, we may still pick out the wrong head, and if the head is correct, we may infer an incorrect segment. The evaluation script also scores predicate disambiguation performance; we did not include this score since the 2005 systems did not output predicate sense identifiers.

To score the output of a segment-based system using the dependency metric, we need to extract a head in every segment. However,

[3]Thanks to Dan Roth for this suggestion.

since CoNLL-2005-style segments have no internal tree structure, this is nontrivial. It is conceivable that the output of the parsers used by the Punyakanok system could be used to extract heads, but this is not recommendable because the Punyakanok system is an ensemble system and a segment does not always exactly match a constituent in a parse tree. Furthermore, the CoNLL-2008 constituent-to-dependency conversion method uses a richer structure than just the raw constituents: empty categories, grammatical functions, and named entities. To recreate this additional information, we would have to apply automatic systems and end up with unreliable results.

Instead, we chose to compute an *upper bound* on the performance of the segment-based system. Every semantic dependency in the gold standard was counted as correctly detected if the argument node was contained in a segment with the correct label.

Table 7.12 shows the results of the dependency-based evaluation. In the table, the output of the dependency-based system is compared to the upper bound on the result of the Punyakanok system.

WSJ	P	R	F1	Prop
Our system	**88.46**	**83.55**	**85.93**	**61.97**
Punyakanok	87.25	81.59	84.32	58.17

Brown	P	R	F1	Prop
Our system	77.67	**69.63**	73.43	**41.32**
Punyakanok	**80.29**	68.59	**73.98**	37.28

WSJ+Brown	P	R	F1	Prop
Our system	**87.07**	**81.68**	**84.29**	**59.22**
Punyakanok	86.94	80.21	83.45	55.39

Table 7.12: *Dependency-based evaluation.*

In this evaluation, the dependency-based system has a higher F1-measure than the Punyakanok system on both test sets. This suggests that the main advantage of using a dependency-based semantic role labeler is that it is better at finding the heads of semantic arguments, rather than finding segments. The results are also interesting in comparison to the multi-view system described by Pradhan et al. (2005b), which has a reported head F1 measure of 85.2 on the WSJ test set. The figure is not exactly compatible with ours, however, since that system used a different head extraction mechanism.

7.6 Discussion

We have described a system for syntactic and semantic dependency analysis based on PropBank and NomBank, and detailed the implementation of its subsystems. The system achieved the highest score in the CoNLL-2008 Shared Task on joint syntactic and semantic analysis. Crucial to our success was the high performance of the syntactic parser, which achieved a high accuracy. In addition, we reconfirmed the benefits of global inference in semantic analysis: both constraint-based and learning-based methods resulted in improvements over a baseline. Finally, we showed that integration of syntactic and semantic analysis leads to a modest improvement for both subtasks.

In addition, to assess the relevance of our work in comparison to previously published results, we compared it to a number of state-of-the-art systems. Our evaluations show that the performance of our system is close to the state of the art, and for some metrics even better. This holds regardless of whether a segment-based or a dependency-based metric is used. Interestingly, our system has a complete proposition accuracy that surpasses other systems by nearly 3 percentage points. Our system is the first semantic role labeler based only on syntactic dependency that achieves a competitive performance.

Evaluation and comparison are difficult issues since the natural output of a dependency-based system is a set of semantic links rather than segments, as is normally the case for traditional systems. To handle this situation fairly to both types of systems, we carried out a two-way evaluation: conversion of dependencies to segments for the dependency-based system, and head-finding heuristics for segment-based systems. However, the latter is difficult since no structure is available inside segments, and we had to resort to computing upper-bound results using gold-standard input; despite this, the dependency-based system clearly outperformed the upper bound on the performance of the segment-based system. The comparison can also be slightly misleading since the dependency-based system was optimized for the dependency metric and previous systems for the segment metric.

Our evaluations suggest that the dependency-based SRL system is biased to finding argument heads, rather than argument text snippets, and this is of course perfectly logical. Whether this is an advantage or a drawback will depend on the application – for instance, a template-filling system might need complete segments, while an SRL-based vector space representation for text categorization or a reasoning application might prefer using heads only.

Chapter 8

Conclusion

This chapter concludes the dissertation and gives a review of its main points. We also give a brief outlook on what research we intend to carry out in the future.

8.1 Contributions

The dissertation has investigated syntactic representation in the form of labeled relations between words, *dependencies*, and how dependency graphs can be used to automatically derive semantic information. The semantic structures that we have used are restricted to the representation of *semantic roles*, that is the logical relations that hold between predicates and their arguments, such as AGENT or INSTRUMENT. While dependency-syntactic analysis and semantic role labeling both have received much focus recently, there has been little work to unify these two fields. Our main effort has been to demonstrate that dependency syntax is suitable for the task of semantic role labeling.

We must emphasize that this work is not intended to take a side in the debate on constituents versus dependencies in general. Its purpose is not polemic, but to fill a gap in current research. In general, we believe that what is the most parsimonious representation depends on the language and on the particular task to solve – as stated previously, dependencies and constituents can be regarded as two *views* on the organizational structure of a sentence. For instance, dependency structures might not be the most expressive or economical representation for explaining and predicting anaphors, prosody, or the acceptable word orders in verb-second languages such as Swedish or

German. It is possible that a multistratal representation that may involve constituent structure or other structures may be the closest to the *true* representation, if such a thing can be conceived. What we do claim is that for the particular task of *explaining the syntax–semantics interface*, dependency representations are both expressive and economical, and thus deserve more attention than they have been given until now.

The first chapters were introductory. We first set out the main principle – *parsimony* – that has guided the design decisions made in this work. Then, we introduced the frameworks that we employed throughout the dissertation: first semantic roles, and then dependency syntax. We finally reviewed a number of guiding principles to use when designing dependency-syntactic representations.

Conceptually, we pointed out that since syntax is the system of signs that expresses semantic relations between the concepts denoted by surface words, our syntactic representations should be designed to reflect this fact. On the other hand, for reasons of scientific soundness, we emphasized that the criteria must be based on observable phenomena. Our dependency representation is thus *surface-syntactic*.

We described the creation of an English dependency treebank via automatic conversion from the Penn Treebank, the best-known constituent treebank for English. While the design was to a certain extent driven by the principles described in the introduction, the limits on what is possible to represent are set by the constituent annotation available.

The experiments described in this work clearly show that the task of semantic role labeling, which has traditionally been solved using constituent syntax as input, can be solved equally well using a dependency representation. Ours is the first published semantic role labeler using only syntactic dependency whose performance rivals that of its constituent-based counterparts, and this result refutes previous claims of insufficiency of dependency representations for this task. The system has been evaluated in two international competitions on semantic analysis, in both of which it achieved the best result. We suggested that the previous negative results published on dependency-based semantic role labeling have been caused by immature dependency parsers.

Since syntax is the surface encoding of the semantic relations between words, it is clear that the two problems of syntactic and semantic analysis are highly interdependent. This suggests that a parsing strategy that solves both tasks simultaneously could be fruitful. While the full problem is intractable in general, at least if we try to model the interaction between grammatical and semantic structures, we were able to implement a simple reranking-based method to integrate syntactic and semantic analysis. We showed that this resulted in a modest improvement in performance for both tasks.

The design of the dependency representation plays an important role: We have shown that a carefully designed dependency framework results in an improved semantic role labeling performance compared to a system based on previously published conversion methods.

In addition, we studied how the subtasks of semantic role analysis (argument identification and labeling) are affected by the choice of a dependency or constituent representation. We could see that constituent-based systems had a slightly higher average performance for the argument identification, although we are not sure how much of this effect is attributable to parser accuracy rather than syntactic representation. For argument classification, on the other hand, dependency-based systems were superior, and the difference becomes more marked when training sets are smaller or when there is a change of domain. We suggested that this is due to the fact that the grammatical function information available in dependency representations makes the classifiers less reliant on lexicalization. Lexical features are sparse and often tied to a certain domain.

8.2 Future Work

There are a number of possible directions in which this work could be extended in the future.

One particularly interesting open problem that was insufficiently explored in this work is the integration of syntactic and semantic analysis. As discussed previously, the computational intractability of the problem precludes exact optimization algorithms. Creativity is thus needed when devising approximate algorithms to solve the joint problem.

We carried out experiments where we integrated syntactic and semantic analysis by applying reranking to a candidate set generated by separate syntactic and semantic modules. However, the reranking strategy is generally agreed to have two important drawbacks:

Efficiency: Reranking requires generation of a set of candidates – the more candidates, the better result. This has a severe impact on the speed of the algorithms and makes the systems prohibitively slow for use in practical applications.

Coverage: Even when a large set of candidates is generated, there may be too little variation in the set and the correct answer is often not included.

Instead of traditional k-best reranking, it could be possible to base syntactic–semantic integration on the *forest reranking* method, which is a recent strategy that makes better use of the search space. It has been employed in machine translation (Huang and Chiang, 2007) and syntactic analysis (Huang, 2008) and led to improvements in both accuracy and efficiency. In this method, the nonlocal features that make dynamic programming impossible are computed as early as possible during search. This contrasts with conventional reranking, where nonlocal features are computed after all search has been carried out (i.e. the k-best candidate set has been computed). Another interesting method for coping with intractable search problems is the method by Smith and Eisner (2008), based on belief propagation.

Apart from issues in the algorithmic and statistical fields, there are also linguistic questions that need to be investigated. In the current work, the described systems have been trained and tested for the English language only. Although our systems are designed generically and have very little infrastructure specifically tied to English, this situation is of course scientifically problematic – it is impossible to tell which is the best algorithm or statistical model *in general* when evaluating on a single language only. It is therefore important to apply our systems on texts in other languages. One opportunity is the proposed CoNLL-2009 Shared Task, which aims to repeat the 2008 effort for other languages than English: Catalan, Chinese, Czech, German, Japanese, and Spanish. In fact, as repeatedly stated, multilinguality is one of the main motivations for this work, since we believe that dependency syntax is the most practical framework cross-linguistically for this task. If English were the only language worth consideration, our work would be less important since English is an exceptionally constituent-friendly language for which SRL works perfectly fine with constituents.

Finally, as a general recommendation to the field, the usefulness of semantic role representations in practical computer applications needs to be investigated thoroughly. After all, the scientific method requires us to avoid intermediate representations as far as possible. We can only make judgments on the scientific soundness of syntactic and semantic structures by demonstrating in practice that they make construction of applications simpler or result in statistically significant improvements in performance.

Bibliography

Babko-Malaya, Olga, Ann Bies, Ann Taylor, Szutik Yi, and Martha Palmer. 2006. Issues in synchronizing the English Treebank and PropBank. In *Proceedings of the Workshop on Frontiers in Linguistically Annotated Corpora*, pages 70–77, Sydney, Australia.

Baker, Collin, Michael Ellsworth, and Katrin Erk. 2007. SemEval-2007 task 19: Frame semantic structure extraction. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 99–104, Prague, Czech Republic.

Baker, Collin F., Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 86–90, Montréal, Canada.

Bejan, Cosmin Adrian and Chris Hathaway. 2007. UTD-SRL: A pipeline architecture for extracting frame semantic structures. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 460–463, Prague, Czech Republic.

Bies, Ann, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary Ann Marcinkiewicz, and Britta Schasberger. 1995. Bracketing guidelines for Treebank II style Penn Treebank project.

Blaheta, Don. 2004. *Function Tagging*. Ph.D. thesis, Brown University.

Boas, Hans C. 2002. Bilingual FrameNet dictionaries for machine translation. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-2002)*, volume IV, pages 1364–1371, Las Palmas, Spain.

Boas, Hans C. 2005. Semantic frames as interlingual representations for multilingual lexical databases. *International Journal of Lexicography*, 28(4):445–478.

Bornkessel, Ina, Matthias Schlesewsky, Bernard Comrie, and Angela D. Friederici, editors. 2006. *Semantic Role Universals And Argument Linking: Theoretical, Typological, And Psycholinguistic Perspectives*. Trends in Linguistics: Studies and Monographs 165. Walter de Gruyter, Berlin.

Boser, Bernhard E., Isabelle Guyon, and Vladimir Vapnik. 1992. A training algorithm for optimal margin classifiers. In *Computational Learning Theory*, pages 144–152, Pittsburgh, United States.

Bottou, Léon. 1991. *Une Approche théorique de l'Apprentissage Connexionniste: Applications à la Reconnaisance de la Parole*. Ph.D. thesis, Université de Paris XI.

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theory*, pages 24–41, Sozopol, Bulgaria.

Buch-Kromann, Matthias. 2006. *Discontinuous Grammar. A model of human parsing and language acquisition*. Ph.D. thesis, Copenhagen Business School.

Buchholz, Sabine and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City, United States.

Burchardt, Aljoscha, Katrin Erk, and Anette Frank. 2005. A WordNet detour to FrameNet. In *Proceedings of the GLDV 2005 workshop GermaNet II*, Bonn, Germany.

Burchardt, Aljoscha, 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 LREC-2006*, Genoa, Italy.

Cahill, Aoife, Michael Burke, Ruth O'Donovan, Stefan Riezler, Josef van Genabith, and Andy Way. 2008. Wide-coverage deep statistical parsing using automatic dependency structure annotation. *Computational Linguistics*, 34(1):81–124.

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

Carreras, Xavier and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of the Eigth Conference on Computational Natural Language Learning (CoNLL-2004)* *at HLT-NAACL 2004*, pages 89–97, Boston, United States.

Carreras, Xavier and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, United States.

Chang, Chih-Chung and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*.

Charniak, Eugene. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139, Seattle, United States.

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, United States.

Che, Wanxiang, 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 Natural Language Learning*, pages 238–242, Manchester, United Kingdom.

Chickering, David M., Dan Geiger, and David Heckerman. 1994. Learning Bayesian networks: The combination of knowledge and statistical data. Technical Report MSR-TR-94-09, Microsoft Research.

Chomsky, Noam. 1981. *Lectures on Government and Binding: The Pisa Lectures.* Foris Publications, Dordrecht, Holland.

Chu, Y.J. and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Collins, Michael. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain.

Collins, Michael. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the International Conference on Machine Learning*, Stanford, United States.

Collins, Michael. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 1–8, University of Pennsylvania, United States.

Collins, Michael, Amir Globerson, Terry Koo, Xavier Carreras, and Peter Bartlett. 2008. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *Journal of Machine Learning Research*, 9:1775–1822.

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

Collobert, Ronan and Jason Weston. 2007. Fast semantic extraction using a novel neural network architecture. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 560–567, Prague, Czech Republic.

Collobert, Ronan 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*, pages 160–167, Helsinki, Finland.

Crammer, Koby, Ofer Dekel, Joseph Keshet, Shai Shalev-Schwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 2006(7):551–585.

Crammer, Koby and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 2003(3):951–991.

Daumé III, Hal, John Langford, and Daniel Marcu. 2006. Search-based structured prediction. Submitted.

Davidson, Donald. 1967. The logical form of action sentences. In N. Rescher, editor, *The Logic of Decision and Action*. University of Pittsburgh Press, Pittsburgh.

Debusmann, Ralph, Denys Duchier, and Geert-Jan M. Kruijff. 2004. Extensible dependency grammar: A new methodology. In *Proceedings of the COLING 2004 Workshop on Recent Advances in Dependency Grammar*, pages 70–76, Geneva, Switzerland.

Diab, Mona, Alessandro Moschitti, and Daniele Pighin. 2008. Semantic role labeling systems for Arabic using kernel methods. In *Proceedings of ACL-08:HLT*, pages 798–806, Columbus, United States.

Dowty, David R. 1991. Thematic proto-roles and argument selections. *Language*, 67(3):574–619.

Edmonds, Jack. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

Einarsson, Jan. 1976. Talbankens skriftspråkskonkordans. Department of Scandinavian Languages, Lund University.

Eisner, Jason M. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 340–345.

Erk, Katrin. 2005. Frame assignment as word sense disambiguation. In *Proceedings of IWCS 6*, Tilburg, Netherlands.

Erk, Katrin and Sebastian Padó. 2006. Shalmaneser – a toolchain for shallow semantic parsing. In *Proceedings of LREC-2006*, Genoa, Italy.

Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Fellbaum, Christiane, editor. 1998. *WordNet: An electronic lexical database*. MIT Press.

Fillmore, Charles J. 1968. The case for case. In E. R. Bach and R. T. Harms, editors, *Universals in Linguistic Theory*. Holt, Rinehart, and Winston, pages 1–88.

Fillmore, Charles J. 1976. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language*, 280:20–32.

Fillmore, Charles J. and Beryl T. S. Atkins. 1992. Towards a frame-based organization of the lexicon: The semantics of RISK and its neighbors. In A. Lehrer and E. Kittay, editors, *Frames, Fields, and Contrast: New Essays in Semantics and Lexical Organization*. Hillsdale: Lawrence Erlbaum Associates, pages 75–102.

Fleischman, Michael, Namhee Kwon, and Eduard Hovy. 2003. Maximum entropy models for FrameNet classification. In *Proceedings of Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Sapporo, Japan.

Gildea, Daniel and Julia Hockenmaier. 2003. Identifying semantic roles using combinatory categorial grammar. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 57–64, Sapporo, Japan.

Gildea, Daniel and Daniel Jurafsky. 2000. Automatic labeling of semantic roles. In *Proceedings of the 38th Annual Conference of the Association for Computational Linguistics (ACL-00)*, pages 512–520, Hong Kong, China.

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

Gildea, Daniel and Martha Palmer. 2002. The necessity of syntactic parsing for predicate argument recognition. In *Proceedings of the 40th Annual Conference of the Association for Computational Linguistics*, pages 239–246, Philadelphia, United States.

Giménez, Jesús and Lluís Màrquez. 2003. Fast and accurate part-of-speech tagging: The SVM approach revisited. In *Proceedings of the International Conference RANLP-2003 (Recent Advances in Natural Language Processing)*, pages 158–165, Borovets, Bulgaria.

Hacioglu, Kadri. 2004. Semantic role labeling using dependency trees. In *COLING 2004: Proceedings of Proceedings of the 20th International Conference on Computational Linguistics*, pages 1273–1276, Geneva, Switzerland.

Hacioglu, Kadri and Wayne Ward. 2003. Target word identification and semantic role chunking using support vector machine. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, Edmonton, Canada.

Haghighi, Aria, Andrew Y. Ng, and Christopher D. Manning. 2005. Robust textual inference via graph matching. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Hajič, Jan. 1998. Building a syntactically annotated corpus: The Prague Dependency Treebank. In *Issues of Valency and Meaning*. pages 106–132.

Hajičová, Eva and Ivona Kučerová. 2002. Argument/valency structure in PropBank, LCS database and Prague dependency treebank: A comparative pilot study. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-2002)*, Las Palmas, Spain.

Henderson, James, Paola Merlo, Gabriele Musillo, and Ivan Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Natural Language Learning*, pages 178–182, Manchester, United Kingdom.

Hickl, Andrew, Jeremy Bensley, John Williams, Kirk Roberts, Bryan Rink, and Ying Shi. 2006. Recognizing textual entailment with LCC's GROUNDHOG systems. In *Proceedings of the Second PASCAL Recognizing Textual Entailment Challenge*.

Hirst, Graeme. 1983. A foundation for semantic interpretation. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, pages 64–73, Cambridge, United States.

Hockenmaier, Julia and Mark Steedman. 2007. CCGBank: A corpus of CCG derivations and dependency structures extracted from the

Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Huang, Liang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, United States.

Huang, Liang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT-2005)*.

Huang, Liang and David Chiang. 2007. Forest rescoring: Fast decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 144–151, Prague, Czech Republic.

Hudson, Richard. 1984. *Word Grammar*. Blackwell.

Hurford, James. 2003. The neural basis of predicate–argument structure. *Behavioral and Brain Sciences*, 23(6).

Johannessen, Janne Bondi. 1998. *Coordination*. Oxford University Press.

Johansson, Richard. 2007. Logistic online online learning methods and their application to incremental dependency parsing. In *Proceedings of the ACL 2007 Student Research Workshop*, pages 49–54, Prague, Czech Republic.

Johansson, Richard, Anders Berglund, Magnus Danielsson, and Pierre Nugues. 2005. Automatic text-to-scene conversion in the traffic accident domain. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1073–1078, Edinburgh, United Kingdom.

Johansson, Richard and Pierre Nugues. 2006. A FrameNet-based semantic role labeler for Swedish. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Session*, pages 436–443, Sydney, Australia.

Johansson, Richard and Pierre Nugues. 2007a. Extended constituent-to-dependency conversion for English. In *NODALIDA 2007 Conference Proceedings*, pages 105–112, Tartu, Estonia.

Johansson, Richard and Pierre Nugues. 2007b. Semantic structure extraction using nonprojective dependency trees. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 227–230, Prague, Czech Republic.

Johansson, Richard and Pierre Nugues. 2007c. Syntactic representations considered for frame-semantic analysis. In *Proceedings of the Sixth International Workshop on Treebanks and Linguistic Theories*, Bergen, Norway.

Johansson, Richard and Pierre Nugues. 2007d. Using WordNet to extend FrameNet coverage. In *Proceedings of the Workshop on Building Frame-semantic Resources for Scandinavian and Baltic Languages, at NODALIDA*, pages 27–30, Tartu, Estonia.

Johansson, Richard and Pierre Nugues. 2008a. Dependency-based semantic role labeling of PropBank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Honolulu, United States.

Johansson, Richard and Pierre Nugues. 2008b. Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *CoNLL 2008: Proceedings of the Twelfth Conference on Natural Language Learning*, pages 183–187, Manchester, United Kingdom.

Johansson, Richard and Pierre Nugues. 2008c. The effect of syntactic representation on semantic role labeling. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 393–400, Manchester, United Kingdom.

Johnson, Mark. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 136–143, Philadelphia, United States.

Kann, Viggo. 1992. *On the Approximability of NP-complete Optimization Problems.* Ph.D. thesis, Royal Institute of Technology.

Karp, Richard M. 1972. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*. New York, pages 85–103.

Kipper, Karin, Hoa Trang Dang, and Martha Palmer. 2000. Class-based construction of a verb lexicon. In *Proceedings of AAAI-2000 Seventeenth National Conference on Artificial Intelligence*, Austin, United States.

Klein, Dan and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 478–485, Barcelona, Spain.

Kudo, Taku and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 24–31, Sapporo, Japan.

Kuhlmann, Marco. 2007. *Dependency Structures and Lexicalized Grammars*. Ph.D. thesis, Saarland University.

Kuhlmann, Marco and Joakim Nivre. 2006. Mildly non-projective dependency structures. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Session*, pages 507–514, Sydney, Australia.

Lafferty, John, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, San Fransisco, United States.

Lecerf, Yves. 1960. Programme des conflits, modèle des conflits. *Bulletin bimestriel de l'ATALA*, 1(4):11–18.

Levin, Beth. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press.

Liang, Percy, Hal Daumé III, and Dan Klein. 2008. Structure compilation: Trading structure for features. In *Proceedings of the 25th International Conference on Machine Learning*, pages 592–599, Helsinki, Finland.

Lin, Dekang. 1998. A dependency-based method for evaluating broad-coverage parsers. *Natural Language Engineering*, 4:97–114.

Litkowski, Ken. 2004. Senseval-3 task: Automatic labeling of semantic roles. In *Proceedings of Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 9–12, Barcelona, Spain.

Liu, Ding and Daniel Gildea. 2008. Improved tree-to-string transducer for machine translation. In *ACL Workshop on Statistical Machine Translation (ACL08-SMAT)*, pages 62–69, Columbus, United States.

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

Magerman, David M. 1994. *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. thesis, Stanford University.

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

Màrquez, Lluís, Mihai Surdeanu, Pere Comas, and Jordi Turmo. 2005. A robust combination strategy for semantic role labeling. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 644–651, Vancouver, Canada.

McCallum, Andrew, Dayne Freitag, and Fernando Pereira. 2000.

Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 591–598, Stanford, United States.

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

McDonald, Ryan and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–88, Trento, Italy.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 987–994, Vancouver, Canada.

Melli, Gabor, Yang Wang, Yudong Liu, Mehdi M. Kashani, Zhongmin Shi, Baohua Gu, Anoop Sarkar, and Fred Popowich. 2005. Description of SQUASH, the SFU question answering summary handler for the DUC-2005 summarization task. In *Proceedings of the Document Understanding Conference*, Vancouver, Canada.

Mel'čuk, Igor A. 1988. *Dependency Syntax: Theory and Practice*. State University Press of New York, Albany.

Mel'čuk, Igor A. 2003. Levels of dependency in linguistic description: Concepts and problems. In V. Agel, L. Eichinnger, H.-W. Eroms, P. Hellwig, H. J. Herringer, and H. Lobin, editors, *Dependency and Valency. An International Handbook of Contemporary Research*, volume 1. Walter de Gruyter, pages 188–229.

Merlo, Paola and Gabriele Musillo. 2005. Accurate function parsing. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 620–627, Vancouver, Canada.

Merlo, Paola and Gabriele Musillo. 2008. Semantic parsing for high-precision semantic role labeling. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8, Manchester, United Kingdom.

Meyers, Adam, Ralph Grishman, M. Kosaka, and S. Zhao. 2001. Covering treebanks with GLARF. In *Proceedings of the ACL/EACL 2001 Workshop on Sharing Tools and Resources for Research and Education*.

Meyers, Adam, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank project: An interim report. In *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 24–31, Boston, United States. Association for Computational Linguistics.

Minsky, Marvin and Seymour Papert. 1969. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge.

Miyao, Yusuke, Rune Sætre, Kenji Sagae, Takuya Matsuzaki, and Jun'ichi Tsuji. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *Proceedings of ACL-08: HLT*, pages 46–54, Columbus, United States.

Miyao, Yusuke and Jun'ichi Tsuji. 2008. Feature forest models for probabilistic HPSG parsing. *Computational Linguistics*, 34(1):35–80.

Moschitti, Alessandro, Paul Morărescu, and Sanda M. Harabagiu. 2003. Open domain information extraction via automatic semantic labeling. In *Proceedings of the 2003 Special Track on Recent Advances in Natural Language at the 16th International FLAIRS Conference*, St. Augustine, United States.

Musillo, Gabriele and Paola Merlo. 2006. Robust parsing for the Proposition Bank. In *Proceedings of the EACL'06 Workshop on Robust Methods in Analysis of Natural Language Data*, Trento, Italy.

Narayanan, Srini and Sanda Harabagiu. 2004. Question answering based on semantic structures. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 693–701, Geneva, Switzerland.

Nilsson, Jens, Joakim Nivre, and Johan Hall. 2006. Graph transformations in data-driven dependency parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 257–264, Sydney, Australia.

Nivre, Joakim, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Nivre, Joakim 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, United States.

Ohara, Kyoko Hirose, Seiko Fujii, Hiroaki Saito, Shun Ishizaki, Toshio

Ohori, and Ryoko Suzuki. 2003. The Japanese FrameNet project: A preliminary report. In *In Proceedings of Pacific Association for Computational Linguistics (PACLING-03)*, Halifax, Canada.

Padó, Sebastian and Mirella Lapata. 2005. Cross-lingual projection of role-semantic information. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 859–866, Vancouver, Canada.

Padó, Sebastian. 2007a. *Cross-Lingual Annotation Projection Models for Role-Semantic Information*. Ph.D. thesis, Saarland University.

Padó, Sebastian. 2007b. Translational equivalence and cross-lingual parallelism: The case of FrameNet frames. In *Proceedings of the NODALIDA Workshop on Building Frame Semantics Resources for Scandinavian and Baltic Languages*, pages 39–46, Tartu, Estonia.

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

Persson, Jacob. 2008. Textkategorisering med predikat-argumentstrukturer. Master's thesis, Lund University.

Persson, Jacob, Richard Johansson, and Pierre Nugues. 2008. Text categorization using predicate–argument structures. Submitted.

Postal, Paul M. 1974. *On Raising: An Inquiry into One Rule in English Grammar and its Theoretical Implications*. MIT Press, Cambridge, Massachusetts.

Pradhan, Sameer, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James Martin, and Dan Jurafsky. 2005a. Support vector learning for semantic argument classification. *Machine Learning*, 60(1):11–39.

Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky. 2005b. Semantic role labeling using different syntactic views. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 581–588, Ann Arbor, United States.

Pradhan, Sameer, Wayne Ward, and James H. Martin. 2008. Towards robust semantic role labeling. *Computational Linguistics*, 34(2):289–310.

Punyakanok, Vasin and Dan Roth. 2001. The use of classifiers in sequential interface. In *The Conference on Advances in Neural Information Systems (NIPS)*, pages 995–1001, Vancouver, Canada.

Punyakanok, Vasin, Dan Roth, and Wen-tau Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *IJCAI-05, Proceedings* *of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1117–1123, Edinburgh, United Kingdom.

Punyakanok, Vasin, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Rambow, Owen, Cassandre Creswell, Rachel Szekely, Harriet Tauber, and Marilyn Walker. 2002. A dependency treebank for English. In *Proceedings of LREC-2002*, Las Palmas, Spain.

Ratté, Sylvie. 1994. A brief note on thematic roles. Memo, Massachusets Institute of Technology, Brain and Cognitive Sciences Department.

Sagae, Kenji and Jun'ichi Tsuji. 2008. Shift–reduce dependency DAG parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 753–760, Manchester, United Kingdom.

Samuelsson, Yvonne, Oskar Täckström, Sumithra Velupillai, Johan Eklund, Mark Fišel, and Markus Saers. 2008. Mixing and blending syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Natural Language Learning*, pages 248–252, Manchester, United Kingdom.

Saussure, Ferdinand de. 1916. *Cours de linguistique générale*. Reprinted Payot, 1995, Paris.

Schmid, Helmut. 2006. Trace prediction and recovery with unlexicalized PCFGs and slash features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 177–184, Sydney, Australia.

Schubert, Klaus. 1987. *Metataxis. Contrastive Dependency Syntax for Machine Translation*. Foris Publications, Dordrecht, Holland.

Schwartz, R. and S. Austin. 1993. A comparison of several approximate algorithms for finding multiple (N-BEST) sentence hypotheses. In *Proceedings of ICSSP*, Minneapolis, United States.

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

Smith, David and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of the Conference on Empirical Methods in*

Natural Language Processing (EMNLP), Honolulu, United States.

Subirats, Carlos. 2008. Spanish FrameNet: A frame semantic analysis of the Spanish lexicon. In *Multilingual FrameNets in Computational Lexicography*. Mouton de Gruyter. Forthcoming.

Surdeanu, Mihai, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 8–15, Sapporo, Japan.

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

Surdeanu, Mihai, Lluís Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, 29:105–151.

Surdeanu, Mihai, Roser Morante, and Lluís Màrquez. 2008. Analysis of joint learning strategies for the semantic role labeling of Spanish and Catalan. In *Proceedings of the the 9th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, Haifa, Israel.

Taskar, Ben, Carlos Guestrin, and Daphne Koller. 2004. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16*, Vancouver, Canada.

Taskar, Ben, Dan Klein, Daphne Koller, and Chris Manning. 2004. Max-margin parsing. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 1–8, Barcelona, Spain.

Tesnière, Lucien. 1959. *Éléments de syntaxe structurale*. Klincksieck, Paris.

Toutanova, Kristina, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 589–596, Ann Arbor, United States.

Toutanova, Kristina, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 173–180, Edmonton, Canada.

Trautner Kromann, Matthias. 2003. The Danish Dependency Treebank and the DTAG treebank tool. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT)*, pages 217–220, Växjö, Sweden.

Tsochantaridis, Iannis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(Sep):1453–1484.

Vadas, David and James R. Curran. 2007. Adding noun phrase structure to the Penn Treebank. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*, pages 240–247, Prague, Czech Republic.

Žabokrtský, Zdeněk. 2000. Automatic functor assignment in the Prague dependency treebank. In *Proceedings of TSD 2000 (Text, Speech and Dialogue)*.

Žabokrtský, Zdeněk, Petr Sgall, and Sašo Džeroski. 2002. A machine learning approach to automatic functor assignment in the Prague dependency treebank. In *Proceedings of LREC-2002*.

Wang, Qin Iris, Dekang Lin, and Dale Schuurmans. 2007. Simple training of dependency parsers via structured boosting. In *Proceedings of IJCAI*, pages 1756–1762, Hyderabad, India.

Xue, Nianwen. 2008. Labeling Chinese predicates with semantic roles. *Computational Linguistics*, 34(2):225–255.

Xue, Nianwen and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 88–94, Barcelona, Spain.

Xue, Nianwen and Martha Palmer. 2007. Adding semantic roles to the Chinese treebank. *Natural Language Engineering*, to appear.

Yamada, Hiroyasu and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of 8th International Workshop on Parsing Technologies*, pages 195–206, Nancy, France.

Yuret, Deniz. 1998. *Discovery of Linguistic Relations Using Lexical Attraction*. Ph.D. thesis, Massachusetts Institute of Technology.

Appendix A

List of Dependency Labels

Table A.1 shows the complete list of dependency labels that we used in our experiments. Labels marked with an asterisk were used only in the CoNLL-2008 treebank, and labels marked with a dagger were used only in the LTH treebank.

Label	Description
ADV	General adverbial
AMOD	Modifier of adjective or adverbial
*APPO	Apposition
BNF	Benefactor complement (*"for"*) in dative shift
† CLR	Used in PTB for various non-object complements of verbs
CONJ	Between conjunction and second conjunct
COORD	Coordination
DEP	Unclassified
DIR	Adverbial of direction
DTV	Dative complement (*"to"*) in dative shift
EXT	Adverbial of extent
EXTR	Extraposed element in cleft
*HMOD	Between a head and a dependent inside a hyphenated word
*HYPH	Between a part of a hyphenated word and a following hyphen
IM	Between infinitive marker (*"to"*) and a verb
† IOBJ	Indirect object
LGS	Logical subject of a passive verb
LOC	Locative adverbial or nominal modifier
MNR	Adverbial of manner
*NAME	Name-internal link
NMOD	Modifier of nominal
OBJ	Object
OPRD	Predicative part of a small clause
P	Punctuation
PMOD	Modifier of preposition
*POSTHON	Posthonorific modifier of nominal
PRD	Predicative complement
PRN	Parenthetical
PRP	Adverbial of purpose or reason
PRT	Between verb and particle
PUT	Prepositional phrase complement of the verb *"put"*
ROOT	Root
SBJ	Subject
SUB	Between subordinating conj. and subordinated clause
*SUFFIX	Between possessor and possessive suffix
*TITLE	Between name and title
TMP	Temporal adverbial or nominal modifier
VC	Verb chain
VOC	Vocative

Table A.1: *List of atomic syntactic labels.*

Appendix B

Search Procedure for Second-order Dependency Parsing

This appendix shows simplified pseudocode of the search procedure for dependency parsing with second-order factors (Carreras, 2007), an extension of the algorithm by Eisner (1996).

For brevity, we present the algorithm to compute the *score* of the best dependency tree given a sentence. To compute the actual tree, backpointers are needed as well. For subroutines depending on link direction, we show only the implementation for the right direction.

The algorithm uses two dynamic programming tables:

$O[s][t][d][l]$	score of the link with left endpoint s, right endpoint t, direction d, and label l
$C[s][t][d][m]$	score of the span starting at s and ending at t, with a top link with direction d ending at m

The following scoring functions are used:

$\sigma(s, t)$	score of the link from the token at s to the token at t
$\sigma(s, t, l)$	score of the link from the token at s to the token at t with the label l
$\sigma_s(s, t, l, k)$	sibling score, i.e. score of the link $s \xrightarrow{l} t$ having an adjacent link $s \rightarrow k$
$\sigma_{gc}(s, t, l, k)$	grandchild score, i.e. score of the link $s \xrightarrow{l} t$ having a subordinated link $t \rightarrow k$

```
function SEARCH($\boldsymbol{x}$)
input sentence $\boldsymbol{x}$
    for $k \in [1, \ldots, |\boldsymbol{x}|]$
        for $s \in [0, \ldots, |\boldsymbol{x}|]$
            $t \leftarrow s + k$
            if $t < |\boldsymbol{x}|$
                COMPLETE-LINKS-RIGHT($s, t$)
                COMPLETE-LINKS-LEFT($s, t$)
                COMPLETE-SPANS-RIGHT($s, t$)
                COMPLETE-SPANS-LEFT($s, t$)
    return $\max_i C[0][|\boldsymbol{x}| - 1][\rightarrow][i]$

procedure COMPLETE-LINKS-RIGHT($s, t$)
input left index $s$, right index $t$
    $L \leftarrow$ allowed labels between $s$ and $t$
    $us \leftarrow \sigma(s, t)$
    for $l \in L$
        $O[s][t][\rightarrow][l] \leftarrow us + \sigma(s, t, l) +$ CREATE-LINK-RIGHT($s, t, l$)

procedure COMPLETE-SPANS-RIGHT($s, t$)
input left index $s$, right index $t$
    for $m \in [s + 1, \ldots, t]$
        $C[s][t][\rightarrow][m] \leftarrow$ CREATE-SPAN-RIGHT($s, t, l$)

function CREATE-LINK-RIGHT($s, t, l$)
input left index $s$, right index $t$, label $l$
    $max \leftarrow -\infty$
    for $r \in [s, \ldots, t - 1]$
        $L \leftarrow$ best sibling of $s \xrightarrow{l} t$ between $s + 1$ and $r$
        $ls \leftarrow C[s][r][\rightarrow][L] + \sigma_s(s, t, l, L)$
        $R \leftarrow$ best grandchild of $s \xrightarrow{l} t$ between $r + 1$ and $t$
        $rs \leftarrow C[r + 1][t][\leftarrow][R] + \sigma_{gc}(s, t, l, R)$
        if $ls + rs > max$
            $max \leftarrow ls + rs$
    return $max$

function CREATE-SPAN-RIGHT($s, t, m$)
input left index $s$, right index $t$, middle index $m$
    $L \leftarrow$ allowed labels between $s$ and $m$
    return $\max_{l \in L, k \in [m+1, \ldots, t]} O[s][m][\rightarrow][l] + C[m][t][\rightarrow][k] + \sigma_{gc}(s, m, l, k)$
```

Appendix C

Notation

$\langle \cdot, \ldots, \cdot \rangle$	Tuple
$[\cdot, \ldots, \cdot], h\|\boldsymbol{t}$	Sequence
$\boldsymbol{x} \cdot \boldsymbol{y}$	Scalar product in a Euclidean space
$g \xrightarrow{l} d$	Dependency from the token g to the token d with label l
$g \xrightarrow{l} \Delta$	Dependency from the token g to the subtree Δ with label l