Re-evaluating and Exploring the Contributions of Constituency Grammar to Semantic Role Labeling

by

Li Yang

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Linguistics)
in The University of Michigan
2009

To my grandfathers
and
my father

Acknowledgments

Michigan has left me so much to remember and so much to be grateful to for the rest of my life.

When I first came to Michigan, I knew little about football. In the past five years, Michigan not only taught me to be able to enjoy football but also taught me the Michigan pride. I have been and will continue to be a proud Michigan fan!

I chose to come to Michigan to pursue my doctoral degree because the university encourages creative work through interdisciplinary studies. I thank the Linguistics Department for backing up the encouragement by supporting my training in other fields such as computer science and statistics.

I chose to pursue my doctoral training in computational linguistics at Michigan Linguistics because of the strong linguistic theoretical and computational background of the professors. I am grateful to professors Rusty Barrett, Pam Beddor, San Duanmu, Sam Epstein, John Lawler, Rick Lewis, Peter Ludlow, and Sally Thomason for my training in phonetics, phonology, syntax, and semantics. I would like to thank professors Drago Redev and Steve Abney for my training in computational linguistics. I am grateful to Steve, my adviser, for shaping my way to do scientific work, for tolerating my mistakes, for giving me the intellectual freedom to explore research possibilities, and for the long discussions that shaped my dissertation work. I would like to thank professors Rich Thomason and George Michailidis for their patience and advice for my dissertation work. I also have special thanks to my B adviser, San, for giving me valuable advice on how to survive as an international student. I am also grateful to Sylvia Suttor for her timely assistance on the administrative side.

My experience at Michigan Linguistics would be incomplete without my cohort and fellow students. I will remember how my cohort, Dina Kapetangianni and Gerardo Gerardo Fern*á*ndez-Salgueiro, encouraged and supported each other in our challenging first two years in the program. I would like to thank all other fellow students that overlapped me in the program for the discussions and Happy Hours we had together. I have special thanks to Rob Felty for providing me with the Latex style file for the current dissertation.

My research experience at Michigan Linguistics would be incomplete without my group mates, Victoria Fossum, Kevin McGowan, Terrence Szymanski, and Yang Ye. I would like to thank them for discussing computational work with me.

My life in the past many years while I was pursuing my graduate education would be incomplete without my wife Zhaohui and son Yankang. I would like to thank my wife for putting up with me for the long hours working on projects in the lab. I would like to thank my son for getting used to not having his Dad around at his shows and competitions. My deepest debt is to Zhaohui and Yankang for their patience and love during my pursuit of the doctoral degree at Michigan.

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

The task of semantic role labeling identifies the semantic argument(s) of a predicate and assigns a semantic role label to each semantic argument. Following the tradition of the Proposition Bank (PropBank), the present work defines a semantic argument as one of the constituents that participate in the event that the predicate is involved in, including the syntactic arguments, adjuncts, modal verbs, negation adverbs, and discourse markers. Since the seminal work on semantic role labeling (SRL) by Gildea and Jurafsky (2000), the community has seen much effort dedicated to the task, which generated dozens of published SRL systems, including the ones participated in the shared tasks on CoNLL-2004 (Conference on Computational Natural Language Learning), CoNLL-2005, and CoNLL-2008. In the past eight years, although researchers approached the SRL problem from different perspectives, they all focused on determining the appropriate syntactic/semantic knowledge and machine learning system in order to tackle the challenges in SRL (Carreras and Marquez, 2005; Surdeanu et al., 2008a). One of the most important challenges is to develop features that generalize across syntactic configurations a verb appears in, configurations involving moved or displaced, or extra arguments that are observed specifically in PropBank. PropBank is the main lexical semantic resource for SRL systems since CoNLL-2004 and provides the training and test data for the current system as well.

In terms of searching for the proper syntactic/semantic knowledge, the SRL researchers explored features based on two formalisms, namely constituency grammar and dependency grammar. The SRL systems constructed between the years of 2000 and 2006 investigated a variety of features that constituency grammar provides for. The lack of improvement in the performance on the commonly used test data, such as Penn Treebank section 23, between 2005 and 2007 motivated SRL researchers to seek help from the grammatical relations between a child word and its head within the framework of dependency grammar (Johansson and Nugues, 2007a,b; Surdeanu et al., 2008a). However, the shift in feature representations, aiming to show that dependency grammar was more suitable for the feature design, was not convincingly supported by the results from CoNLL-2008. As a matter of fact, the researchers suggested, post CoNLL-2008, that determining the right syntactic/semantic knowledge for the SRL task still remained an open question (Surdeanu et al., 2008a). This conclusion implies that finding the right syntactic/semantic knowledge that generalizes across different syntactic configurations that the same verb may appear in remains a challenge.

The present work continues the research effort to discover and utilize the appropriate syntactic/semantic knowledge for the SRL task. Specifically, while seeking the right features to solve the SRL problem in general, the present project focuses on tackling the challenge from syntactic configuration variations by integrating three less thoroughly explored types of syntactic/semantic knowledge in the context of SRL into the feature and system designs that enable generalizing across the different syntactic configurations of a verb, including the knowledge about the context dependence among the semantic roles of the core semantic arguments of a verb, that about the structures where argument movement or displacement occurs, and that about the grammatical relations between the arguments and the predicate. A brief description of each type of knowledge follows.

The current work takes advantage of the context dependency among the semantic roles of the core semantic arguments of a verb. Previous SRL systems that based their feature designs on constituency grammar follow the **linking theory** in assuming that a linking relation exists between a semantic argument and its semantic role (Levin and Hovav, 1996). Relying on this relation, such systems intend to uncover the semantic role based on the syntactic features of the semantic argument. However, the revised linking theory, under the name of **theory of argument realization**, not only observes the argument-semantic role link but also states the strong dependency among the semantic roles of the core arguments of a verb (Levin and Hovav, 2005). Due to this strong dependency, when the core semantic roles are realized, they are associated with a fixed syntactic configuration relevant to the verb, where the corresponding semantic arguments occupy specific positions in the configuration. Such positional correlation between the semantic roles and the core arguments is formulated as a strong constraint on the core semantic arguments in the current project. Although several SRL systems utilized such dependency among the semantic roles, these systems did not distinguish between core and non-core semantic arguments and listed all the semantic arguments in relation to the verb. The current work takes into account that the dependency among semantic arguments referred to by the **theory of argument realization** (AR) is only among core semantic arguments. An investigation of the AR theory as well as the lexical resources of the Proposition Bank and the FrameNet confirms this interpretation. The current work hence imposes the strong **dependency constraint** on the feature design only for the **core semantic arguments** of a verb.

The preceding dependency constraint emphasizes that a core semantic argument of a given verb occupying a specific syntactic position is assigned a fixed semantic role. This constraint does not seem to apply to the core semantic arguments in several syntactic structures that have undergone transformations. For example, the semantic role of the semantic argument occupying the subject position of a verb in active voice does not match that of the subject of the same verb in passive voice. To resolve the position-role-mismatch between the active and passive voices, the present system defines a **base argument configuration** corresponding to the core semantic arguments' positions in one of the verb's argument structures. Then, the core semantic arguments in the passive voice are brought to conform with this **base argument configuration**. The present system handles the core semantic arguments in other transformed syntactic structures in the same manner. The operation to align the transformed core semantic arguments with a base argument configuration of a given verb requires knowledge of the syntactic structures where one or more core arguments have undergone transformations as well as knowledge of their positions in one of the verb's syntactic argument structures. Thus, the current system identifies twelve types of transformed syntactic structures from Penn Treebank/PropBank and makes use of the knowledge about the transformed structures.

The knowledge of the transformed structures not only entails the concepts depicted in constituency grammar, such as phrase types, head words, and governing nodes, voices of the verb, positions of arguments in relation to the predicate, etc., but also entails a variety of word-to-head relations (see section 3.3.2) described in dependency grammar, including relations such as subject, object, modifier of nominal, modifier of preposition, modifier of adjective, etc. For example, to reconstruct the arguments' positions from a passive sentence, one needs the knowledge of the phrase types, subject relation, and object relation. In order to correctly handle the transformed structures, instead of siding with a specific grammar framework, the current work draws upon the concepts from both formalisms.

Combining the context dependency constraint, knowledge about the structures where argument movement occurs, and the grammatical relations between the arguments and the predicate, the current project offers a unique solution to the varying syntactic configuration challenge for three reasons. Firstly, the current system translates the preceding three types of knowledge into the base argument configuration (BAC) features for the core semantic arguments of a verb that generalize across different configurations that a verb appears in. Secondly, the operations that derive the BAC features from the different configurations of the verb are straight-forward and efficient because the system only needs to encode which argument(s) has/have moved in twelve syntactic structures and what positions they should be in regarding their verb. Finally, although not explicitly extracting the grammatical relations defined in dependency grammar as features which previous constituency grammar based systems did not utilize, the current system relies on these grammatical relations to reconstruct the base argument configurations of the core arguments.

Incorporating the BAC features into the system, the current work presents a new approach to the SRL task. More specifically, the system starts with identifying the core and non-core semantic arguments for each verb in an input sentence. The system then classifies the identified semantic arguments. To classify the core semantic arguments, the system first checks if there is any transformation among the arguments. If there is, it aligns the arguments up with those in a base argument configuration of the verb. Then, for each core argument, the system extracts a contextual feature (the BAC feature) in relation to other core arguments and the verb form by enforcing the dependency constraint. If the verb has an unrealized argument/arguments, the system backs off to a more general feature, Level-I feature. In cases of unknown verbs, the system backs off to the most general set of features, Level-II features. For the non-core semantic arguments, the system extracts only the Level-II features. When the features of all the arguments are extracted, a regularized logistic regression model assigns corresponding semantic role labels to the arguments. A nearest-neighbor classifier and a naive Bayes classifier are also applied to the arguments for the purpose of comparisons with the logistic regression model.

Three sets of experiments were conducted to test the current feature and system designs. To examine the effectiveness and build the performance upper bound of the current system independently of the argument identifier, the first set of experiments ran the argument classifier on section 23 of the Penn Treebank with the semantic arguments pre-annotated. The system achieved 88.89% of F-measure with the logistic regression model and 89.41% of F-measure with a nearest neighbor classifier. Both F-measures are above the 82.92% F-measure by the baseline classifier which relies on a generic set of features found in most constituency grammar-based SRL systems. The F-measure achieved by the current feature and system designs is also above the 85.38% of F-measure from a state-of-the-art re-ranking-based joint modeling system by Surdeanu et al. (2007) on the same data set. This indicates the current feature and system designs are relatively effective in terms of handling the varying syntactic configurations challenge. The argument classifiers in this set of experiments are called the oracle systems.

The second and third sets of experiments showed that the argument classifier performs consistently in different settings. Both sets of experiments also show that the performance of the argument identifier affects the argument classifier. Specifically, in the third set of experiments, the argument identifier obtained an F-measure of 79.80% on section 23 parsed with the Charniak parser. Consequently, the argument classifier gained an F-measure of 70.86%. Since 70.86 is about 89% of 79.80 and is comparable to the F-measure of 89% of the oracle systems from the first set of experiments, the performance of the argument classifier remains consistent but is conditioned on that of the argument identifier. Therefore, the immediate future task for the current project is to refine the argument identifier.

The rest of the report proceeds as follows. Chapter 2 lays the foundation of the current work in three perspectives. First, chapter 2 defines semantic roles and the semantic role labeling task. Second, chapter 2 introduces the lexical resources that are commonly used for training and evaluation in the field of SRL. Third, following the definitions and introduction, chapter 2 provides for the linguistic theoretical background of the current project by describing the aspects of the **theory of argument realization** that are relevant to the current project. Specifically, the concept of *context dependence* is defined. In addition to the preceding definitions, Chapter 2 introduces the definitions of *base argument configuration* (BAC) and the BAC feature. Chapter 3 reviews the SRL systems since the year of 2000, highlighting how the situation where a verb has different syntactic configurations still remain a challenge for the researchers. The differences between the constituency grammar-based feature design and the dependency grammar-based feature design are also discussed in the same chapter. Chapter 4 describes the feature design of the current project for the core and non-core arguments, emphasizing how the current project addresses the issue with the varying syntactic configurations of a verb. Chapter 5 incorporates the feature design into a complete statistical learning-based SRL system. Chapter 6 describes the three sets of experiments conducted to test the current system.

Chapter 2
Definitions, lexical resources, argument structure, base argument configuration, and argument realization

Before the discussion of any semantic role labeling system, it is necessary to present the definitions of semantic roles, semantic arguments, and the task of automatic semantic role labeling (SRL), to describe the lexical resources that enable automatic SRL, to describe argument structure and define base argument configuration of a verb, and to provide for the linguistic theoretical background of the current SRL project. Thus, this chapter begins with defining traditional semantic roles, semantic arguments, and the automatic SRL task. Due to the fact that the Proposition Bank (PropBank) and the FameNet database are the two standard lexical resources for training and evaluating existing SRL systems, the chapter proceeds with a description of both lexical resources, highlighting that both resources distinguishing between verb-specific core semantic roles and non-core semantic roles general to all verbs. The chapter follows the description of the lexical resources with defining the argument structure of a verb and presents the concept of base argument configuration for this project. The chapter then moves forward to present the linguistic theoretical background of the current system, the **theory of argument realization** (AR) that accounts for the **context dependence** between the semantic role of a core argument with those of other core arguments. Having observed the practical handling by the lexical resources and the theoretical rendering by the AR theory of the core and non-core semantic roles/arguments, the chapter then concludes that the current system employs verb-specific context dependence features for core semantic arguments and generic features general to non-core arguments.

2.1 Definitions

To lay the background for the upcoming discussions in later sections and chapters, this section presents the definitions for **traditional semantic roles**, **semantic arguments**, **core semantic arguments**, **non-core semantic arguments**, **semantic argument identification task**, and **semantic role labeling task**.

2.1.1 Traditional semantic roles

Semantic roles are semantic labels assigned to the grammatically relevant aspects of a verb's meaning that identifies the role that each of the verb's semantic arguments plays in the event it denotes (Levin and Hovav, 2005, p. 35). The grammatically relevant aspects of a verb's meaning are responsible for mapping lexical semantics to syntax in the framework of argument realization (Levin and Hovav, 2005, p. 9). The list of fixed roles associated with the semantic arguments of a verb is referred to as a **semantic role list**, also known as a *case frame* (Fillmore, 1968) or a *theta-grid* (Stowell, 1981). The concepts of semantic role and list has a long history dated back to the Sanskrit grammarian Pāṇini. The well-known theories in modern times on semantic roles are found in the work of Fillmore's **Case Grammar**, Gruber and Jackendoff's **thematic relations** (Levin and Hovav, 2005, p. 35). Although there is consensus on the functions of semantic roles among lexical semanticists, the actual roles subsumed in the semantic role list of a verb may differ slightly from one lexical semanticist to another. Regardless of such differences, the list in (1) below from Fillmore (1968, p. 376) gives a good example of the definitions of some of the common semantic roles.

(1) a **Agent**, the instigator of the event
b **Counter-Agent**, the force or resistance against which the action is carried out
c **Object**, the entity that moves or changes or whose position or existence is in consideration [1]
d **Result**, the entity that comes into existence as a result of the action
e **Instrument**, the stimulus or immediate physical cause of an event
f **Source**, the place from which something moves
g **Goal**, the place to which something moves
h **Experiencer**, the entity which receives or accepts or experiences or undergoes the effect of an action

The semantic roles defined in this section will be referred to as traditional semantic roles in later chapters because the recently developed lexical resources that the current project relies on defines a large number of fine grained semantic roles specific to verb senses. Such lexical resources include the Proposition Bank (PropBank) (section 2.2) and FrameNet (section 2.3). Verb-specific semantic roles defined in PropBank instead of the above traditional semantic roles are the ones that the current system utilizes for training and testing.

[1] This is a quote from Fillmore (1968, p. 376) which is equivalent to the *Theme* role by its definition.

The arguments participating in the event that the verb predicate denotes are in fact **verb semantic arguments**, as opposed to the **syntactic arguments** of the verb. The next section defines the concept of **verb semantic arguments** which is critical to the design of the current project.

2.1.2 Verb semantic arguments, core and non-core semantic arguments

The preceding section brings up the concept of verb semantic arguments, as opposed to the syntactic arguments of a verb. A precise definition of **verb semantic argument** is due in this section. According to Culicover and Jackendoff (2005, p 173), entities that are intrinsically involved in the event that the predicate verb denotes are called **semantic arguments** of the verb. As part of its meaning, the verb stipulates the number of semantic arguments. The semantic roles defined in the preceding section are then the labels assigned to the semantic arguments of a verb. For example, the verb *expect* with the meaning of *anticipate* as shown in (2) specifies its two semantic arguments with the semantic role labels of *expecter* and *thing expected* shown in (3). (4) shows that the semantic arguments *expecter* and *thing expected* are realized as the syntactic arguments, *subject NP* and *object NP* respectively.

(2) Verb: *expect*; Meaning: anticipate

(3) Number of semantic arguments: 2; Corresponding semantic role labels: *expecter*, *thing expected*

(4) $[_{Subject\_NP\_expecter}$ Portfolio managers$]$ expect $[_{Object\_NP\_thing\_expected}$ further declines in interest rates$]$.

The illustration above shows that semantic arguments of a verb are associated with specific semantic role labels determined by the verb. The rest of the section explains how the distinction between core and non-core semantic arguments arises.

Computational linguists, such as Palmer et al. (2003) and M‘arquez et al. (2008), incorporate into semantic arguments the traditional syntactic arguments, adjuncts, model verbs, discourse markers, and negation adverbs. In other words, the preceding grammatical units can all be realized as semantic arguments. An outcome of the integration is the development of a lexical semantic resource, the Proposition Bank (PropBank), the main resource for training and testing semantic role labeling systems. More details on PropBank are given in section 2.2. PropBank formally names the subset of semantic arguments realized as syntactic arguments **core arguments**. But the semantic arguments realized as adjuncts, model verbs, discourse markers, and negation adverbs are not given a formal type in PropBank but are sometimes referred to as **non-core arguments** (Gildea and Hockenmaier, 2003; Vickrey and Koller, 2008). The current project also refers to these semantic arguments as **non-core arguments**. In recent semantic role labeling literature, researchers tend to use **arguments** in replace of **semantic arguments**. The current project follows this trend. Thus, unless otherwise specified each occurrence of *argument* refers to a semantic argument in the rest of the report.

The results that follow from the distinction between core and non-core semantic arguments are that core semantic arguments, realized as syntactic arguments, are specific to a verb and that non-core semantic arguments, realized as syntactic adjuncts, model verbs, discourse markers, or negation adverbs, are general to all verbs.

This following example illustrates the core and non-core arguments. (5) lists the core argument *MEETER* of the verb *meet* and some of the non-core arguments, *LOCATION*, *TEMPORAL*, and *PURPOSE*. In (6), the *MEETER* argument is realized as the *subject NP* syntactic argument. The *LOCATION TEMPORAL*, and *PURPOSE* arguments are realized as the *PP* locative adjunct, *NP* temporal adjunct, and infinitival *VP* adjunct respectively.

(5) Verb: *meet*; Meaning: *get together*; Core argument: **MEETER**; Non-core arguments: **LOCATION**, **TEMPORAL**, **PURPOSE**, etc.

(6) [$_{NP_MEETER}$The economic and foreign ministers of 12 Asian and Pacific nations] will meet [$_{PP\_LOCATION}$ in Australia] [$_{NP\_TEMPORAL}$ next week] [$_{VP\_PURPOSE}$ to discuss global trade as well as regional matters such as transportation and telecommunications].

2.1.3 Automatic semantic role labeling task

Informally, the task of automatic semantic role labeling entails the subtask of **identifying** the semantic arguments that are realized as constituents, including arguments, adjuncts, modal verbs, discourse markers, and negation adverbs, and the subtask of **assigning** the correct semantic role label to each *argument*. The latter task is also known as the **argument classification** task. For example, the constituents that are realized semantic arguments in (7) are identified and shown in (8). Each identified argument/constituent is labeled with the correct semantic role in (9).

(7) Peter broke the jar with a rock.

(8) [$_{NP}\underline{\text{Peter}}$] broke [$_{NP}\underline{\text{the jar}}$] [$_{PP}\underline{\text{with a rock}}$].

(9) $[_{Agent}$Peter$]$ broke $[_{Object}$the jar$]$ $[_{Instrument}$with a rock$]$.

Formally, the SRL task in current project is defined in (10)

(10) Given sentence S and verbs $v_1, \cdots, v_m$ in S, for each verb v_i, the semantic role labeling task is to assign a unique semantic role, r_j, to the corresponding semantic argument a_j, of v_i.

2.2 PropBank

The Proposition Bank provides for the training and evaluation data for a majority of the published SRL systems. The SRL system in the present work also relies on these data from training and testing. Therefore, a depiction of the lexical resource is due. The Proposition Bank (PropBank), developed at the Computer and Information Sciences Department at the University of Pennsylvania in 2004, is the semantic annotation of the Wall Street Journal sections of Penn Treebank-2. Specifically, PropBank adds a layer of predicate-argument information, or semantic role labels, to the syntactic structures of the Penn Treebank. PropBank defines semantic roles on a verb-by-verb basis, consisting of roles specific to a verb's sense and roles general applicable to any verb. In the following, each type of role is described.

2.2.1 Verb specific roles

PropBank annotates the senses of each verb. For each verb with a specific sense, PropBank numbers their semantic arguments or roles, beginning with zero and ending with five, the notation of which is Arg_i. These numbered roles specific to a verb sense are called **core semantic roles**, comparable to FrameNet's core frame elements. Each semantic role is given a definition. A set of roles corresponding to a verb with a specific sense is referred to as **roleset**. Each roleset may be associated with a set of syntactic frames indicating allowable syntactic variations in the expression of the roleset. The roleset with its associated syntactic frames is called a **frameset**. The collection of frameset entries for a verb is referred to as the verb's **frames file**.

The examples in (11), (12) and (13) correspond to the three framesets in the frame file for the verb *yield*. The roleset, semantic roles and their definitions, and an annotated sentence are given for each of the three senses of *yield*. Each example shows only one set of the syntactic frames/variations associated with *yield*.

(11) Frameset of verb: **yield** | WordNet sense & ID: 01, *result in*

> **Roleset**: {Arg0, Arg1}
>
> Arg0: thing yielding
>
> Arg1: thing yielded
>
> Ex: [$_{Arg0}$The top money market funds] are currently *yielding* [$_{Arg1}$well over 9%].

(12) Frameset of verb: **yield** | WordNet sense & ID: 02, *give way*

> **Roleset:** {Arg0, Arg1, Arg2}
>
> Arg0: thing giving way
>
> Arg1: what's lost
>
> Arg2: what's preferred
>
> Ex: [$_{Arg0}$John] *yielded* [$_{Arg1}$the right-of-way] [$_{Arg2}$to the Mack truck].

(13) Frameset of verb: **yield** | WordNet sense & ID: 03, *give a dividend*

> **Roleset:** {Arg0, Arg1, Arg2}
>
> Arg0: thing providing a dividend
>
> Arg1: dividend, earnings
>
> Arg2: recipient
>
> Ex: [$_{Arg0}$A new, 12-year Canada Savings Bond issue] will *yield* [$_{Arg2}$investors] [$_{Arg1}$10.5%] $\cdots$

2.2.2 General roles

While *Arg0* to *Arg5* are specific to each verb sense, general roles apply to any verb sense. General roles are adjunct-like arguments (ArgsMs) or adverbial subordinate clauses to the verb, distinguished by one of the function tags shown in Table 2.1. General roles are also referred to as **non-core roles**, comparable to Framenet's non-core frame elements. The verb-level negation NEG and modal verbs MOD are also annotated as general roles although they are not considered as adjuncts.

LOC: location	CAU: cause
EXT: extent	TMP: time
DIS: discourse connectives	PNC: purpose
ADV: general purpose	MNR: manner
NEG: negation marker	DIR: direction
MOD :modal verb	

Table 2.1: Subtypes of the ArgM roles

2.2.3 PropBank Statistics

The current PropBank contains frames (frame files) for over $3,300$ verbs distributed over $4,500$ framesets, indicating that each verb has approximately 1.36 senses. Specifically, 21.6% (7213342) of the frames (frame files) contains more than one frameset, meaning 21.6% or 721 of the verbs have a single sense, while less than 100 verbs have four or more senses. Each sense of a polysemous is assigned to an appropriate frameset, with an inter-annotator agreement of 94% (Palmer et al., 2005).

2.3 FrameNet

The FrameNet lexical database provides for the training and test data for a number of SRL systems, including the seminal work of Gildea and Jurafsky (2000), and for the SRL tasks on Senseval-2003 and Semeval-2007. The main components of FrameNet are described in this section because the lexical resource is referred to in the literature review in Chapter 3. However, the current project does not use FrameNet for training or testing.

The FrameNet lexical database, being developed by the Berkeley FrameNet project since 1997, provides online lexical semantic resources for English. Built by virtue of Frame Semantics (Fillmore, 1968), the FrameNet database records the semantic and syntactic combinatory possibilities, *i.e.* valences, for each word in each of its senses. Frames, frame elements, lexical units, relationships between frames and frame elements (FE), and others comprise the FrameNet database (Ruppenhofer et al., 2006). Description of the current status of the FrameNet project will follow that of the preceding components.

2.3.1 Frames and Frame Elements

Following Frame Semantics, the FrameNet database has frames and their component frame elements as the basic units. A frame is a script-like conceptual structure that consists of a definition of this frame, a list of frame elements, relationships with other frames, and lexical items subsumed by this frame.

The frame definition describes the participants and propositions of a specific type of event. Frame elements (FEs) are the semantic roles specific to a frame. The two main sub-types of frame elements are the **core** and **peripheral** classes. Frame elements that are not core FEs are referred to as non-core FEs.

Core frame elements uniquely define a frame and are conceptually necessary to this frame. That is, each frame has a unique list of core frame elements assigned to it. Each core frame element has a unique definition. No two frames share the same list of core frame elements. For example, the list of core frame elements for the *Appeal* frame in table 2.2 are different from those for the *Employing* frame in table 2.3. Since the list of frame elements are unique to a frame, they are conceptually necessary to the frame.

Appeal Frame	
Definition	A Survivor manages to avoid being negatively affected, despite encountering a Dangerous_situation.
Core Frame Elements	
Dangerous_situation	The Dangerous_situation is one in which people are likely or expected to be negatively affected.
Survivor	The Survivor lives through a Dangerous_situation.

Table 2.2: *Appeal* frame definition

Peripheral FEs are the semantic roles representing notions such as TIME, PLACE, MANNER, MEANS, DEGREE, *etc*, while not distinctively characterizing a frame. The FEs that are not core FEs are called non-core FES. Each frame element is associated with a semantic role and its definition, and a semantic type. Table 2.3 lists the definition, core FEs, and non-core FEs of the **Employing** frame. Examples for the core the non-core elements are also listed.

2.3.2 Frame-to-frame Relations

The FrameNet database resembles all ontologies in that a network of relations are provided between frames. Eleven inter-frame relations are defined, including **Inheritance**, **Subframe**, **Has Subframes**, **Precedes**, **Is Preceded by**, **Using**, **Is Used by**, **Perspective on**, **Is Perspectivized in**, and **Is Causative of**. The most important relations according to Ruppenhofer et al. (2006) are **Inheritance**, **Using**, **Subframe**, and **Perspective on**.

The **Inheritance** relation is an IS-A relation. The **Inheritance** relation describes the

Employing Frame	
Definition	An Employer employs an Employee whose Position entails that the Employee perform certain Tasks in exchange for Compensation.
Core Frame Elements	
Employee	The FE Employee denotes the person who is obligated to perform some Task in order to receive Compensation. E.g., ***I*** *was EMPLOYED by an international corporation.*
Employer	The Employer is the person or institution that gives Compensation to an Employee. E.g., ***I*** *EMPLOYED him as Chief Gardener for ten years.*
Field	The FE Field identifies the field in which the Employee is employed. E.g., *He was EMPLOYED* ***in finance*** *fourteen years ago.*
Position	The FE Position indicates a particular type of employment. E.g., *I'm not EMPLOYED* ***as your waitress****!*
Task	The Task indicates the action/duty that the Employee is obligated to do for the Employer. E.g., *I am EMPLOYED* ***to collect the trash.***
Non-core Frame Elements	
...	...
Duration	The FE Duration identifies the amount of time for which the Employee continues in employ. Semantic Type: **Duration** E.g., *I EMPLOYED him as Chief Gardener* ***for ten years.***
Manner	The FE Manner identifies the way in which an Employer hires an Employee. Semantic Type: **Manner** E.g., *The three young men were given medals and* **hastily** COMMISSIONED.
Place	The FE Place identifies the location where the Employer hires the Employee. Semantic Type: **Locative_relation** E.g., *Actually, he EMPLOYED me* ***at the downtown office .***
Time	The FE Time identifies the time at whcih the Employer hires the Employee. Semantic Type: **Time** E.g., *Charles has been an EMPLOYEE of Microsoft* ***since 1998!***
...	...

Table 2.3: Employing Frame Definition

scenario where the child frame is a subtype of the parent frame, with each FE in the parent frame bound to a corresponding FE in the child. For example, the **Absorb_heat** frame inherits from the **Becoming** frame, which in return inherits from the **Event** frame.

With the **Using** relation, the child frame presupposes the parent frame as background, not all parent FEs necessarily bound to child FEs. The **Arriving** frame presupposes the **Path_shape** frame.

In the **Subframe** relation, the child frame is a subevent of a complex event represented by the parent. The **Arriving** frame is a subframe of **Traversing**. The **Departing** frame is also a subframe of **Traversing**.

The **Perspective on** relation describes the situation where the child frame provides a particular perspective on an un-perspectivized parent frame. The **Containing** frame perspectivize the **Containment** frame from the point of view of the *Container*.

The relevant inter-frame relations of the **Employing** frame are shown in table 2.4. The hierarchical semantic relations shown between the frames above also resemble the hierarchical relations among the words' senses in WordNet (Fellbaum, 1998).

Employing Frame (cont.)	
Inter-frame Relationships	
Relationship	Related Frame
Subframe of	Employer's _scenario
Precedes	Firing
Is Preceded by	Hiring
Perspective on	Employment_continue
Subsumed Lexical Units	
Lexical Units	*commission.v, employ.v, employee.n, employer.n, employment.n, personnel.n, staff.n, worker.n*

Table 2.4: Employing Frame Definition (cont. from table 2.3)

2.3.3 Lexical Units

Table 2.4 lists the lexical units subsumed by the **Employing** frame. Each lexical unit is defined by a specific sense of a word, its part-of-speech, and lexical entry. The sense of each word, i.e. lexical unit, is either the definition from the Concise Oxford Dictionary, 10th Edition or a definition created by a FrameNet staff member. Each sense of a word is provided with approximately 20 annotated examples, illustrating all syntactic combinatorial possibilities of the word. Each lexical unit is linked to a frame, and hence to the other lexical units that subsumed by the same frame. This grouping of semantically similar words in the same frame makes the FrameNet database similar to a thesaurus.

In the FrameNet database, the information about each lexical unit is shown in two reports, the lexical entry report and the annotation report. The lexical entry report contains:

- the LU definition
- any support or governing words annotated for the LU
- a table of frame elements and their syntactic realizations
- a table of valence patterns (syntactic patterns for each possible group of semantic roles).

Table 2.5 displays the lexical entry report for the lexical unit **employ**. The frequencies of the lexical unit's syntactic realizations and valence patterns are recorded, which may be useful for a statistical learning system that learns from these syntactic realizations and patterns.

Employ Lexical Unit		
Part-of-Speech	**Verb**	
Frame	**Employing**	
Definition	give work to (someone) and pay them for it.	
Frame Elements and Their Syntactic Realizations		
Frame Element	Number Annotated	Realizations(s)
Employee	(70)	NP.Appositive (4)
		NP.Ext (8)
		NP.Obj (58)
Employer	(70)	NP.Ext (62)
		CNI.– (2)
		PP[by].Dep (6)
Position	(25)	NP.Obj (1)
		PP[as].Dep (6)
		...
Task	(52)	2nd.– (5)
		INI.– (23)
		vPto.Dep (16)
		...

Table 2.5: Lexical Entry Report of *employ*

The annotation report contains:

- a brief guide to the frame elements
- FrameNet semantic annotation of corpus examples of the LU

The annotation report for the lexical unit **employ** defined in table 2.5 lists the frame elements of **employ**, including the core FEs and non-core FEs shown in table 2.3. The annotation report also provides about 60 annotated sentences with **employ** in different syntactic structures. For each sentence, the parts-of-speech of the words, the constituents, and the grammatical relations of the constituents are also annotated. For example, the annotations of one of these sentences are shown in (14):

(14) Sent: Canal companies EMPLOYED divers to investigate underwater problems.
POS: NN1-NP0 NN2 VVD-VVN AJ0 TO0 VVI AJ0 NN2 PUN

2.3.4 Current Status

The FrameNet database release 1.3 (Ruppenhofer et al., 2006) includes more than $10,000$ lexical units, covering about 800 semantic frames. More than $6,000$ of the lexical units are fully annotated. More than $135,000$ annotated sentences are provided for the lexical units. The inter-annotator agreement of the frames is not available.

2.4 From base argument configuration to base argument configuration feature

2.4.1 Syntactic argument structure

The syntactic arguments of a verb refer to its subject and complement. Subjects are also known as external arguments, and complements internal arguments (Radford, 1997, p. 325). Syntactic argument structure refers to the number of external and internal arguments that a particular predicate verb requires in a clause (Carnie, 2002, p. 166). This definition indicates that the argument structure of a verb not only specifies the number of syntactic arguments it takes but also determines their positions in the structure. Based on the number of possible arguments and their positions in the argument structure of a verb in a single clause, Quirk et al. (1985, p. 53) summarize three main types of argument structures. These are:

2-element structure: $Subj. + V$

3-element structure: $Subj. + V + Obj./AdjP/Inf./PP/...complement$

4-element structure: $Subj. + V + Obj. + Obj./AdjP/Inf./PP/...complement$

In the approach of Quirk et al. (1985), the verb is counted as an element of an argument structure. (The verb in the three structures is also known as the 1-place predicate, 2-place predicate, and 3-place predicate respectively (Radford, 1997, p. 325).) It is possible that a verb has more than one argument structure. That is, it may appear in one or more of the three configurations. For example, the verb *want* appears in two different argument structures in (15) and (16).

(15) $[_{subject}$John$]$ wants $[_{object_complement}$a Michigan shirt$]$. $\Rightarrow Subj. + V + Obj.complement$

(16) $[_{subject}$John$]$ wants $[_{infinitival\_clause\_complement}$ to watch a Michigan football game $]$. $\Rightarrow$ $Subject + V + Infinitival\_clausal\_complement$

2.4.2 Base argument configuration

The preceding section shows that Quirk et al. (1985) represent argument structures by sequentially listing the argument types, such as *Subject* + *V* + *Object* etc. The current project represents argument structures in a different format. In this format, instead of listing argument types, such as the *subject* and *object complement* in (15) and (16), an argument structure of a verb is represented as a sequence consisting of the lemmatized verb with the syntactic categories of each of its syntactic arguments in the order in which they appear in the parse tree. Such representation of an argument structure of the verb is called a **base argument configuration** (BAC) of the verb. A verb may have one or more base argument configurations because a base argument configuration corresponds to an argument structure of a specific verb and the verb may have more than one argument structure, as examples (15) and (16) show above.

For example, the base argument configurations corresponding to the two argument structures of *want* in (15) and (16) are shown in (17) and (18).

(17) $[_{NP}$ John$]$ $[_{VP}$ wants $[_{NP}$ a Michigan shirt$]]$. $\Rightarrow NP - want - NP$

(18) $[_{NP}$John$]$ $[_{VP}$ wants $[_{S}$ to watch a Michigan football game$]]$. $\Rightarrow NP - want - S$

The base argument configuration $NP - want - NP$ in (17) of the predicate *want* is analogous to its argument structure in (16) except that the verb lemma is included in the BAC and the argument types are replaced with the syntactic categories.

2.4.3 Base argument configuration feature

The task of argument classification classifies an argument-verb instance and assigns a semantic role label to the instance based on its features. The base argument configuration (BAC) feature is one of the features for the instance. The **BAC feature** is essentially a listing of the arguments in the base argument configuration with respect to a given verb, while marking the argument in question as the current argument. For example, (19).1 is the instance created for the argument *John* in (17). Its BAC feature is shown in (19).2. And its position is marked by the string *cur*.

(19) (1) **Instance:** *John_want*

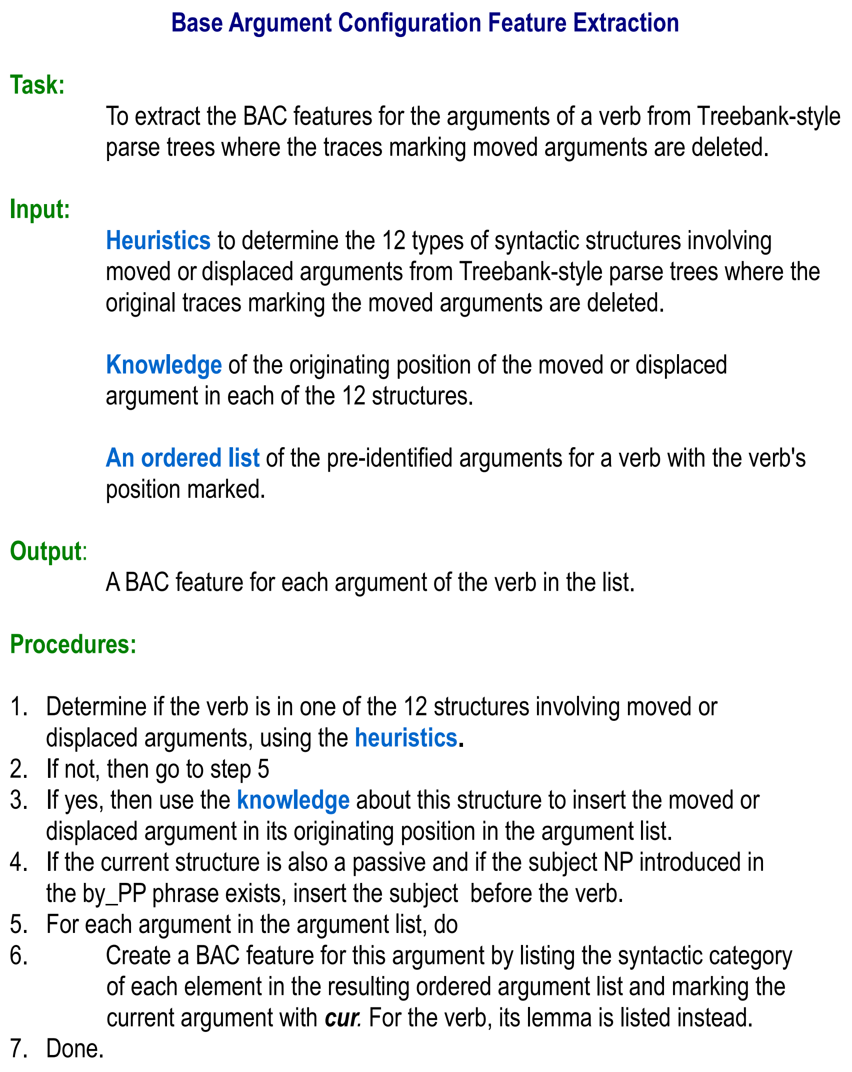
(2) **BAC feature:** *curNP_want_NP*

While a BAC feature captures the argument structure of a given verb, it serves two other important functions as well.

First, a BAC feature approximates the context dependence among the semantic roles of the core arguments of a given verb. As section 2.5.3.1 will show, there exists context dependence among the semantic roles of core arguments as illustrated in the theory of argument realization. While the semantic roles of the arguments are unknown and cannot be used as features, listing the syntactic categories of all core arguments functions as an approximation of context dependence. Since the syntactic categories of all the core arguments are present in the BAC feature, it serves as an approximation of context dependence among the arguments.

Second, a BAC feature generalizes across the syntactic structures involving moved or displaced arguments of a given verb. In real life data, such as that collected in the Penn Treebank, arguments of predicate verbs have frequently been displaced or moved out of the positions of their corresponding verb's argument structure. Creating the BAC features for the moved or displaced arguments of a given verb helps to generalize across the syntactic structures that the verb appears in because arguments positions in the base argument configuration overlap with the those in one of the verb's argument structures.

However, the BAC features cannot be extracted for the displaced arguments by simply listing the syntactic categories of the verb's arguments as is because the positions of the displaced arguments do not overlap with those in one of the verb's argument structures. The next section illustrates how base argument configuration features are created for the displaced arguments in passive voice and subject control.



Base Argument Configuration Feature Extraction

Task:

To extract the BAC features for the arguments of a verb from Treebank-style parse trees where the traces marking moved arguments are deleted.

Input:

Heuristics to determine the 12 types of syntactic structures involving moved or displaced arguments from Treebank-style parse trees where the original traces marking the moved arguments are deleted.

Knowledge of the originating position of the moved or displaced argument in each of the 12 structures.

An ordered list of the pre-identified arguments for a verb with the verb's position marked.

Output:

A BAC feature for each argument of the verb in the list.

Procedures:

1. Determine if the verb is in one of the 12 structures involving moved or displaced arguments, using the **heuristics.**
2. If not, then go to step 5
3. If yes, then use the **knowledge** about this structure to insert the moved or displaced argument in its originating position in the argument list.
4. If the current structure is also a passive and if the subject NP introduced in the by_PP phrase exists, insert the subject before the verb.
5. For each argument in the argument list, do
6. Create a BAC feature for this argument by listing the syntactic category of each element in the resulting ordered argument list and marking the current argument with ***cur***. For the verb, its lemma is listed instead.
7. Done.

Figure 2.1: The BAC extraction module

2.4.4 Creating BAC features for moved or displaced arguments

The current project observes twelve types of syntactic structures in Treebank/PropBank where moved or displaced arguments occur, including eight types of moved arguments, one type of displaced arguments, one type of arguments whose predicates are in co-ordinated structures, and one type of extra arguments associated with relative clauses (see section 4.3 for the illustration of each type). The module for extracting BAC features is summarized in figure 2.1. This module requires as input the heuristics to detect the twelve structures, the knowledge about the originating positions of the moved or displaced argument(s) of the verb in each structure, and the list of pre-identified arguments of the verb. Such heuristics and knowledge help with both the argument identification and classification tasks. The rest of this section illustrates the procedures to extract base argument configuration features for two of the twelve structures, including the passive structure and the subject control structure.

At this point, the author would like to draw the reader's attention to the fact that the forementioned twelve types of syntactic structures are the ones observed in TreeBank/PropBank with noticeably high frequencies. The current system is designed to handle these structures but does not handle the structures that involve other types of moved or displaced arguments that either appear in low frequencies in TreeBank or have not been observed or do not appear in Treebank at all.

2.4.4.1 Extracting BAC features for passive structures

This section illustrates how the BAC features are created for the arguments of a passive verb in examples (20).2 through (20).8.

(20) (1) $[_{NP}$ The problems$_i]$ $[_{VP}$were uncovered *trace$_i$* ... $[_{PP}$ by $[_{NP}$ the government$]]]$.

(2) $[_{NP}$ The problems$]$ $[_{VP}$ were uncovered ... $[_{PP}$ by $[_{NP}$ the government$]]]$.

(3) **Core argument list** (NP_*Subject*, uncovered_*passive*, PP$_{Adverbial}$)

(4) **Unmoved the subject** (uncovered_*passive*, NP_*Subject*, PP$_{Adverbial}$)

(5) **Replaced the PP argument** (PP$_{Adverbial}$, uncovered_*passive*, NP_*Subject*)

(6) **In base argument configuration** (NP, uncover, NP_*Subject*)

(7) **BAC feature for *The problems*** (NP-uncover-curNP)

(8) **BAC feature for the PP** (curNP-uncover-NP)

(20).1 shows the Penn Treebank version of an annotated passive sentence where the moved argument *the problems* is marked with an index along with its trace. The current project encodes its knowledge about the structures involving moved arguments using such annotations of traces. In this case, the current system learns from this and other passive clauses that the subject of the passive verb is a moved argument and it occupies the object position when the verb is active voice. At the same time, the PP-argument headed by *by* in passive voice occupies the subject position when the verb is in active voice. Such knowledge helps with extracting the BAC features more than argument identification.

The parsed sentence (20).2 without any trace marked is the actual input to the current system. The trace of the moved argument *the problems* is not shown for two reasons. First, the current system uses the CoNLL-2005 training data which consists of sections 2 to 22 of the Penn Treebank where all traces have been deleted. Second, for testing, the current system as well as many others rely on an automatic parser, such as Charniak parser, which does not output annotations for traces. The current system then relies on the knowledge learned from the Treebank annotations to identify the structures involving moved arguments, such as that about passive voice described in the last paragraph.

Following the procedures of the BAC feature extraction module in figure 2.1, a list of core arguments shown in (20).3 are extracted for the verb *uncovered* from the parse tree.

Since the verb is in passive voice, the subject *The problems* is a moved argument, and the adverbial *by the government* is a displaced argument, the BAC features cannot be extracted for the two arguments as is because the arguments in a base argument configuration are not moved or displaced. To extract the BAC features for the two arguments, the system first un-moves the subject by inserting it behind the verb as shown in (20).4.

Since *uncover* is in passive voice, there is an extra operation to resolve the PP-argument. That is, relying on the knowledge that in passive voice the PP-argument headed by *by* is the subject of the verb in active voice, the system inserts the PP-argument in the beginning of the argument list and rewrites the syntactic category as NP.

Through the above operations, the resulting base argument configuration of *uncover* is shown in (20).5. Now, the BAC feature for subject *The problems* is extracted as *NP-uncover-curNP* based on its position in the base argument configuration. And the BAC feature for the PP argument *by* is extracted as *curNP-uncover-curNP*.

2.4.4.2 Extracting BAC features for subject control structures

This section illustrates how the BAC features are extracted for the arguments in the subject control structure in examples (21).2 through (21).6, specifically for the arguments of the lower-clause verb *pull*. This section also shows that the knowledge about the arguments' positions in the subject control structure contributes to the argument identification task.

(21) (1) $[_{NP}$ John$_i]$ $[_{VP}$tried $[_{S}$ $[_{NP}$*trace$_i$*$]$ $[_{VP}$ to pull $[_{NP}$the wool $]]]]$ ...

(2) $[_{NP}$ John$]$ $[_{VP}$tried $[_{S}$ $[_{VP}$ to pull $[_{NP}$ the wool $]]]]$ ...

(3) **Args. list for *pull*:** (NP_John, pull, NP_the wool)

(4) **Base argument configuration for *pull*:** (NP, pull, NP)

(5) **BAC feature for *NP_John*:** (curNP-pull-NP)

(6) **BAC feature for *NP_the wool*:** (NP-pull-curNP)

(21).1 shows a Treebank version of a sentence with subject control, where the subject of the matrix verb is raised from its co-indexed trace. From this and other similar subject control structures, the system learns that the subject of the matrix verb is also the subject of the lower-clause verb. This knowledge is encoded in the system and is useful for identifying the subject of the lower verb.

(21).2, with the trace and co-indices deleted, represents the actual input sentence the current system sees. While the object of the lower-clause verb *pull* can be identified easily because its parent is *pull*, the system relies on the knowledge described in the preceding paragraph to identify the subject for the lower-clause verb *pull*. As a result, both arguments of *pull* are identified and shown in (21).3. Now, a base argument configuration corresponding to the predicate verb is derived and shown in (21).4. Extracting the BAC features for the arguments needs just to list the elements in the configuration. Finally, the two BAC features corresponding to the two arguments are shown in (21).5 and (21).6.

2.5 Semantic roles and argument realization

The goal of SRL is to identify the semantic roles of the arguments of the verb(s) in an unseen sentence, where no annotation of any kind is available. To achieve this goal, SRL systems, especially the early ones such as Gildea and Jurafsky (2000), presuppose that a verb's syntactic structures depend on its meaning and that semantic roles, representing the lexical meaning of the verb, can be uncovered by using the syntactic structures because semantic roles are mapped onto arguments by linking rules. Both presuppositions are made on the basis of the **linking theory** (Levin and Hovav, 1996). In addition to assuming this linking relationship between syntactic structures and semantic roles, SRL researchers also assume the dependencies among the arguments and their semantic roles, based on their knowledge and intuition about the English language. The researchers have done so without consulting the **theory of argument realization** (AR), the revised *linking theory* in Levin and Hovav (2005) that examines in more detail the relations among arguments and their semantic realization. An investigation of the main concepts of the AR theory is due for two reasons. First, the AR theory offer a linguistic theoretical background to the SRL task. Second, the AR theory meticulously studies the dependency among arguments and their corresponding semantic roles.

While presenting the AR theory as the linguistic theoretical background for the current project, this section also serves the purpose of connecting the practice of electronic lexical resources such as the PropBank and the FrameNet database with the linguistic theory in lexical semantics by showing that these lexical resources provide for a fine-grained solution to the grain-size problem that lexical semanticists previously had not been able to successfully solve using the traditional semantic roles such as those defined in section 2.1.1 . By bridging the gap between the lexical semantic theory and the unsolved grain-size problem, the lexical resources provide for useful information for automatic semantic role labeling systems.

The results from the above investigation of the AR theory and the comparison between the AR theory and modern lexical semantic resources are two-fold. On the one hand, the AR theory provides for the lexical semantic basis for the dependencies among the semantic roles of the corresponding co-arguments of a verb. On the other hand, due to the fact that FrameNet and PropBank lexical resources extend the AR theory, the aspects regarding non-core semantic roles in FrameNet and PropBank that are not discussed by the AR theory need to be considered in the current project. The comparison shows that the dependency among arguments in the AR theory really entails those among core arguments and that the semantic role of a non-core argument may be independent of that of other core or non-core arguments.

2.5.1 The theory of argument realization

As expressed in the work of Stowell (1981), Pesetsky (1982), and Chomsky (1986), many theories of grammar are built on the assumption that syntactic realization of arguments is largely predictable from the meaning of their verbs. In this line of research, *the theory of argument realization* studies the possible syntactic expressions of the arguments of a verb, comprising their category type and their grammatical function. Attempts to implement the program of predicting syntactic structures from the meaning of a verb, such as the Projection Principle, are often focused on generalizations concerning argument realization without grounding their work in an articulated theory of lexical semantic representations (Levin and Hovav (1996, 2005)). Aiming to successfully implement the program of predicting syntactic structures from the verb's meaning, the theory of argument realization takes the position of the Universal Alignment Hypothesis and extends it with two specific goals grounded in articulated theories of lexical representations. The Universal Alignment Hypothesis (UAH), proposed by Perlmutter and Postal (1984, p. 97), formalizes the above assumption behind many theories of grammar, such as Stowell (1981) and Pesetsky (1982) in (22).

(22) Universal Alignment Hypothesis: There exist principles of UG which predict the initial [grammatical] relation borne by each nominal in a given clause from the meaning

of the clause.

To implement the program laid out in the UAH, the AR theory strives to meet two goals:

Goal I For a verb with a specific core meaning/sense, identify the grammatically relevant aspects of the meaning/sense.

Goal II Explicate the components' connection to the range of argument realization options regarding the verb.

Sections 2.5.2 and 2.5.3 elaborate on the two goals respectively.

2.5.2 SRs representing grammatical aspects of verb meaning

Identifying the grammatically relevant components of meaning serves as the foundation for the theory of argument realization. The identification task entails the task of investigating the nature of a lexical representation that can encode the grammatically relevant facets of verb meaning (Levin and Hovav, 2005). Different theories have been proposed to accomplish this task. Some of these are complex theories such as Talmy's components and conflation patterns, Jackendoff's conceptual structure (Jackendoff, 1990), and Pustejovsky's generative lexicon Pustejovsky (1995). Perhaps the simplest and yet most commonly adopted forms of lexical representation for the grammatically relevant aspects of verb meaning are **semantic role lists** (Levin and Hovav, 2005). This tradition can be traced back to Fillmore's work in the early 1970's. The definitions of some semantic roles given by Fillmore are introduced in section 2.1.1. The rest of the section reviews the reason, illustrated in Fillmore 1970 and adopted widely in the field of lexical semantics (Saeed, 2003; Levin and Hovav, 2005), that semantic roles make good lexical representation encoding the relevant aspects of verb meaning.

Fillmore (1970) investigates the relation between lexical meaning of the verbs and the syntactic structure of the sentences with these verbs as the matrix verbs. In his investigation, Fillmore observes that verbs with different core meanings each appear with a specific set of syntactic structures. The issue that Fillmore is interested in is what makes the set of syntactic structures associated with the verb with a given sense different from the set associated with the verb with another sense.

To illustrate the situation where a core sense of a given verb is associated with a specific set of syntactic structures, Fillmore gives the widely-quoted examples of the verbs *hit* and *break*. Table 2.6 paraphrases the original examples given in Fillmore (1970, p. 126) and Levin and Hovav (2005, p. 37):

break (changing of state)	***hit (contacting surface)***
Peter broke the jar with a rock.	Peter hit the stump with a stick.
A rock broke the jar.	A stick hit the stump.
The jar broke.	*The stump hit.
The jar was broken.	The stump was hit.
John broke Peter's leg.	John hit Peter's leg.
*John broke him on the leg.	John hit him on the leg.

Table 2.6: Examples of *break* and *hit* with their respective core senses

The left column in table 2.6 lists the examples corresponding with the core sense of the verb *break*, involving *changing the state* of the relevant argument in bold fonts. The right column lists the examples corresponding with the core sense of the verb *hit*, involving *contacting the surface* of the relevant argument in bold fonts. The syntactic structures for each set of examples are shown in table 2.7 as grammatical relations of the noun arguments of the corresponding matrix verb.

break (changing of state)	***hit (contacting surface)***
Subject broke Object PP_instrument	Subject hit Object PP_instrument
Subject broke Object	Subject hit Object
Subject broke.	*Subject hit.
Subject was broken.	Subject was hit.
Subject broke Object	Subject hit Object
*Subject broke Object PP_place.	Subject hit Object PP_place.

Table 2.7: Syntactic structure shown as grammatical relations.

Table 2.7 displays the structural similarities and differences in the data with *break* and *hit* as the matrix verbs. It seems that both verbs license structures including *subject-verb-object*, *subject-verb-object-PP_instrument*, and the *passive*. At the same time, while *break* licenses the intransitive structure *subject-verb* but does not appear with the *subject-verb-object-PP_place* configuration, *hit* does not appear with the intransitive configuration but occurs with *subject-verb-object-PP_place*.

To account for the similarities and differences in the syntactic configurations of the two verbs, Fillmore starts with assigning semantic roles to the arguments in table 2.7. The results of the assignment are shown in table 2.8. The *OBJECT* role in table 2.8 is defined in section 2.1.1 as *the entity moves or changes or whose position or existence is in consideration*. It is different from the grammatical role/relation of *object*. From the role assignments, Fillmore summarizes two semantic role lists that characterize the role assignments:

List a *break:* {(AGENT)* (INSTRUMENT)* OBJECT}

break (changing of state)	***hit (contacting surface)***
AGENT broke OBJECT INSTRUMENT	AGENT hit PLACE INSTRUMENT
AGENT broke OBJECT	SUBJECT hit PLACE
OBJECT broke	*Subject hit
OBJECT was broken	PLACE was hit
AGENT broke OBJECT	AGENT hit PLACE
*Subject broke Object PP_place.	AGENT hit (possessor) PLACE

Table 2.8: Semantic role assignment for arguments in tables 2.6 and 2.7

List b *hit:* {(AGENT|INSTRUMENT|AGENT INSTRUMENT){1}PLACE}

Each list subsumes three semantic roles. The order of the roles in each list is irrelevant. Role list *a* contains three roles, including AGENT, INSTRUMENT, and OBJECT. Role list *b* includes three roles, namely, AGENT, INSTRUMENT, and PLACE. Each role list is more than just listing the roles themselves. It also states the rules about the presence of the roles in the actual semantic role assignment. Explanation about these rules follows.

In list *a*, the regular expression (ROLE)* indicates that this ROLE may be present or not in the final semantic role assignment. Thus, either the AGENT role, or the INSTRUMENT role, or both may be absent from the final role assignment. The ROLE without any parentheses around it must be present in the role assignment. In other words, the OBJECT role must be present in the final assignment. These rules precisely predict the roles appearing in the final role assignment shown in table 2.8. For example, the role assignment for the arguments *Subject broke* of the data *The jar broke* is *OBJECT broke*, where only the OBJECT role is present but the AGENT and INSTRUMENT roles are absent. On the other hand, there does not exist any valid role assignment for the data *John broke him on the leg* because the semantic role, that is, list *a*, associated with the *changing-of-state* sense of the verb *break* does not permit a PLACE role for the argument *on the leg*.

Similarly in list *b*, the regular expression (AGENT|INSTRUMENT|AGENT INSTRUMENT){1} indicates that at least one of the AGENT and INSTRUMENT roles must be present in the final assignment. At the same time, the PLACE role must always be present in the final assignment. This is in fact the case shown in the role assignment in table 2.8. There is not any valid role assignment for the data *The stump hit* because neither the AGENT role nor the INSTRUMENT role has any argument to assign to.

From the above, one can see that the semantic role lists and the associated rules about the presence of the roles relate the semantic roles to the actual role assignment for the arguments shown in table 2.8. Due to this, Fillmore concludes that the structural differences between *break* and *hit* that originated in the lexical differences between the two are captured by the semantic role lists. Hence, the first goal of the theory of argument theory to identify the relevant components of meaning is met. Semantic roles are such components.

Discovering semantic roles as the lexical representation of the grammatical aspects of verb meaning has achieved the first goal of the AR theory. The other goal of the AR theory is to find the regularities that connect the role assignment in table 2.8 to their corresponding arguments shown in table 2.7. Fillmore himself is one of the first researchers who proposes argument selection rules that choose roles from the semantic role list to fill their corresponding argument positions shown in table 2.7. These rules are dubbed as the linking/mapping rules and are further developed by later researchers such as Carter (1988) and Pinker (1989) and will be discussed in section 2.5.3.

2.5.2.1 Additional examples

Section 2.5.2 shows how semantic lists of verbs contribute to the structural differences of verbs with different meanings. This section provides more evidence to support this point with three verbs of the same form, *finger*. The meaning, syntactic structures, and semantic roles corresponding to the core arguments of the verb, *finger*, in examples 23, 24, and 25, are listed in Table 2.9. Non-core arguments are not shown in the table.

(23) ... [$_{NP}$ the big first baseman] fingered [$_{NP}$ the gold bat he wears on a neck chain].

(24) To date, [$_{NP}$ scientists] have fingered [$_{NP}$ two of these cancer-suppressors].

(25) ... [$_{NP}$ Hugo Spadafora] publicly fingered [$_{NP}$ Mr. Noriega] [$_{PP}$ on drug trafficking charges].

Note that the part of speech of *v* in Table 2.9 is added to the core arguments only for the purpose of showing the position of the arguments in relation to the verb, which itself is not an argument.

Example	23	24	25
Meaning	to touch	to locate, find	to blame
Meaning ID	01	02	03
Core arguments	NP-v-NP	NP-v-NP	NP-v-NP-PP
Arguments w/ Sense	NP-v01-NP	NP-v02-NP	NP-v03-NP-PP
Semantic roles	toucher-entity touched	agent-entity found	assigner-entity-action blamed for

Table 2.9: Sense, syntactic structures, and roles for *finger*

As Table 2.9 shows, the concatenated phrase types *NP-v-NP* and *NP-v-NP* of the core arguments for verbs *finger*_01 and *finger*_02 respectively are identical. It is the semantic role sets *toucher-entity touched* and *agent-entity found* help to distinguish between *finger*_01 and _02. The core arguments *NP-v-NP-PP* suffices to distinguish *finger*_03 from other two senses of *finger*, while the role set *assigner-entity-action blamed for* is an extra help.

2.5.3 Argument realization: mapping semantic roles to arguments

Section 2.5.2 demonstrates that semantic roles make good representation of the grammatical aspects of a verb's meaning. To achieve the goal of discovering the regularities that map semantic roles to the verb's arguments, lexical semanticists start with *context dependence* between the semantic roles of the arguments of the verb.

2.5.3.1 Context dependence

Lexical semanticists observe and agree on a dependency relationship between the semantic role of an argument and those of other arguments of the same predicate verb. This relationship is referred to as **context dependence**. Context dependence entails that

> the options for the syntactic realization of a particular argument are often not determined solely by its semantic role, but also by the semantic roles borne by its co-arguments (Levin and Hovav, 2005, p.158).

In other words, a particular semantic role needs not consistently be associated with a specific syntactic realization, with the exact realization depending on co-occurring roles. (26) through (30) exemplify context dependence between the semantic role of an argument and that/those of other co-argument(s).

(26) An instrument can't be realized as subject in the presence of an agent.

a. ∗The chisel opened the door by Dana.

(27) A recipient can't be realized as subject in the presence of an agent.

a. Alex received a package.

b. Sam sent Alex a package.

c. ∗Alex sent a package by Sam. (*Sam being the sender*)

d. Alex was sent a package by Sam.

(28) An experiencer can't be realized as subject in the presence of an agent/causer.

a. The toddler (*deliberately) feared the lion.

b. The lion (deliberately) frightened the toddler.

(29) A moving entity (i.e., theme) can't be realized as subject in the presence of an agent.

a. Kelly moved the cat into the room.

b. The cat was moved into the room by Kelly.

c. ∗The cat moved into the room by Kelly.

d. The cat entered the room.

(30) A moving entity can't be an object in the presence of an entity that changes state.

a. Shelly smeared oil on the axle.

b. Shelly smeared the axle with oil.

c. ∗Shelly smeared oil the axle. (***Oil** being realized as a direct object.*)

d. ∗Shelly smeared the axle oil. (***Oil** being realized as an indirect object.*)

Example (27).c above is in active voice and is ungrammatical because *Alex*, the recipient, is realized as the subject while *Sam*, the agent, is present. Example (27).d is in passive voice and is grammatical although *Sam*, the agent, is present. Similarly, (30).c is in active voice and is ungrammatical because the moving entity, *the cat*, is the subject while *Kelly*, the agent, is present. At the same time, the passive in (30).b is grammatical although the moving entity, the *cat*, is the subject, while the agent *Kelly* is also present. These examples illustrate that the original rules in (27) and (30) both need to be constrained with the condition of *in active voice*.

The preceding examples demonstrate the dependency relationship among the semantic roles of the semantic arguments of the same verb. Two important observations one can make from the discussion above are as follows. First, context dependence applies to the semantic roles of the core semantic arguments but not to those of the non-core semantic arguments. Second, the semantic roles of the core semantic arguments of a given verb that are contextually dependent on each other are realized through a specific syntactic configuration of the semantic arguments of the verb, within which the positions of the arguments assume fixed positions relevant to the verb and other arguments. For example, (27).b and (27).d demonstrate how the *recipient* role is contextually dependent on the *agent* role. When realized, the same *recipient* role is associated with a specific syntactic configuration. The syntactic configuration in (31) corresponds to the realization of the *recipient* role of the argument *Alex* in (27).b. In (31), *Alex* is realized as the object in the configuration with respect to the verb *send*. The syntactic configuration in (32) corresponds to the realization of the *recipient* role of the argument *Alex*. In (32), *Alex* is realized as the subject in the configuration with respect to the passive verb *send*.

(31) [$_{NP_agent}$Sam]-send-[$_{NP_recipient}$Alex]- [$_{thing_sent}$a package]

(32) [$_{NP_recipient}$Alex]-sent_passive-[$_{thing_sent}$a package]-[by-[*NP_agent*Sam]

Due to the strong association between context dependence and syntactic configuration of the core arguments of a given verb, the current project utilizes the syntactic configuration of the core arguments as a clue to uncover the semantic role label of the semantic arguments of the verb. The current project refers to the process of doing so as imposing the context dependence constraint.

At the same time, one can conclude that the positions of the core arguments of a verb in the syntactic configurations discussed in the current section overlap with the argument positions in a base argument configuration of the verb (see section 2.4.2). Therefore, the current project utilizes the BAC feature to impose the context dependence among the core arguments (see section 4.2 for more detailed discussion).

2.5.3.2 Linking regularities

The second goal of the theory of argument realization is to account for context dependence using linking regularities. The concept of mapping rules or linking regularities that map semantic roles to the relevant verb arguments is first proposed by Fillmore (1968) and further developed by a number of researchers such as Carter (1988), Dowty (1987), and Pinker (1989). A definition of *linking/mapping rules* proposed by Pinker (1989, p. 74) goes as follows:

(33) Linking rules are regular ways of mapping ... arguments onto grammatical functions or underlying syntactic configurations by virtue of their thematic roles; they are the mechanisms that create the syntactic argument structure associated with a given thematic core.

The *linking rules* defined in 33 include two mechanisms, as proposed by Fillmore (1968), the **thematic hierarchy** (TH) for an argument and the argument **selection rule** corresponding to this TH. The **thematic hierarchy** is the ranking of semantic roles, which establishes prominence relations among them for a specific argument. An argument **selection rule** (SelR) selects a relevant semantic role to be a specific argument for the matrix verb.

For example, (34) a shows a sample Subject TH, and (34) b lists the corresponding subject SR for this TH. Together, (34) a and (34) b account for the grammatical and ungrammatical subject selections in (35). Specifically, in (35) b, the semantic roles present for selection are, AGENT, INSTRUMENT, and THEME. Based on TH (34) a and the corresponding SR (34) b, the argument *John* is selected as the subject because *John* the AGENT has the highest rank. At the same time, (35) f selects the argument *the key* which bears an INSTRUMENT role as the subject while the argument *John* which bears an AGENT role is available for select. This violates the thematic hierarchy (34) a and subject selection rule (34) b.

(34) (a) Subject TH: Agent $\succ$ Instrument $\succ$ Theme/Patient

(b) Subject SelR: The argument of a verb bearing the highest-ranked semantic role in the subject TH is its subject. Specifically, if there is an Agent, it becomes the subject; otherwise, if there is an Instrument, it becomes the subject; otherwise, the subject is the Theme/Patient.

(35) (a) $[_{AGENT}$John$]$ opened $[_{THEME}$the door$]$.

(b) $[_{AGENT}$John$]$ opened $[_{THEME}$the door$]$ with $[_{INSTRUMENT}$a key$]$.

(c) $[_{INSTRUMENT}$The key$]$ opened $[_{THEME}$the door$]$.

(d) $[_{THEME}$The door$]$ opened.

(e) *$[_{THEME}$The door$]$ opened by $[_{AGENT}$John$]$.

(f) *$[_{INSTRUMENT}$The key$]$ opened $[_{THEME}$the door$]$ by $[_{AGENT}$John$]$. Levin and Hovav (2005, p. 158)

Researchers, such as Fillmore (1968), Pinker (1989), and Levin and Hovav (2005), illustrate the concepts of thematic hierarchy, and argument selection rules with the TH for subject and the corresponding subject selection rule as shown in the examples above. Interestingly, none of these researchers gives examples of the TH and SRs for an argument other than the subject, such as the direct object and PP-complements/adjuncts. If one wishes to explain the mapping between a semantic role and the direct object and the PP-adjunct in (35), one can propose the THs for the direct object and the PP-adjunct and the corresponding SelRs in (36) and (37).

(36) (a) Direct Object TH: Theme

(b) Direct Object SelR: The argument of a verb bearing the highest-ranked semantic role in the direct object TH is its direct subject. Specifically, the semantic role theme becomes the direct object of the verb.

(37) (a) Instrument TH: Instrument

(b) Instrument SR: The PP-argument of a verb bearing the highest-ranked semantic role in the instrument TH is its PP-adjunct. Specifically, the semantic role instrument becomes the PP-adjunct of the verb.

Note that the direct object hierarchy and the instrument hierarchy each contain only one semantic role. The two hierarchies and their corresponding selection rules seem to cover the data in 35.

The examples in this section show that the two mechanisms, thematic hierarchy and selectional rules seem to meet the second goal of mapping semantic roles with the arguments of a verb for the theory of argument realization. However, the concept of thematic hierarchy faces **two challenges**. First, there is no agreement among lexical semanticists on what *the ranking of the arguments is derived from* (Levin and Hovav, 2005, p.157). Second, there is lack of agreement among lexical semanticists *over the appropriate formulation of the thematic hierarchy, in terms of both the roles constituting it and the ranking of these roles* (Levin and Hovav, 2005, p.157). These challenges make it difficult to represent the hierarchy within an automatic semantic role labeling system and to quantify the ranking among arguments and semantic roles.

The previous sections draw a picture of the theory of argument realization. The next section illustrates the grain-size problem with the semantic role list approach and focuses on computational linguists practical find-grained solution to the grain-size problem.

2.6 Chapter summary

While providing for the necessary information on the definitions of semantic roles, the semantic role labeling task, the lexical resources, and base argument configuration, the foregoing discussion focuses on the aspects of the **theory of argument realization** that are relevant to the SRL task, including the linking regularities on mapping arguments to the corresponding syntactic configurations through semantic roles, the representation of grammatical aspects of verb meaning by semantic roles, and the context dependency among the semantic roles of the core arguments. Section 2.5.3.1 presents the concept of context dependence between the semantic role of a core semantic argument and those of other semantic arguments of the same predicate verb and illustrates the strong correlation between context dependence and syntactic configuration of the core arguments of a given verb. The next chapter reviews the main trends in SRL and highlights the existing open research questions for the community. One of these is the challenge to be able to generalize across the different configurations of the same predicate verb. Chapter 4 presents a solution by utilizing the strong correlation between syntactic configurations and context dependence to uncover the semantic roles of the core arguments.

Chapter 3
Review of Semantic Role Labeling Systems

To provide a basis for sophisticated semantic interpretation tasks such as text summarization, fact-based question answering, and complex semantic and dialogue structure recognition, Gildea and Jurafsky (2000) were among the first researchers attempting to achieve this goal by building a semantic role labeling system, trained and evaluated on the FrameNet database. Gildea and Palmer (2002) extended the work by training and evaluating the earlier system developed in Gildea and Jurafsky (2000) using Propbank. The seminal work in Gildea and Jurafsky (2000) and Gildea and Palmer (2002) spawned a series of research work on semantic role labeling using both FrameNet and Propbank. The main events involving SRL include the **Senseval** (Evaluation Exercises for the Semantic Analysis of Text) series in 2004 and 2007, and the **CoNLL** (Conference on Computational Natural Language Learning) shared task series in 2004, 2005, 2008, and the upcoming 2009. The Senseval events focused on the use of the FrameNet database, and the CoNLL shared tasks encouraged using Propbank. In between the events, several systems using either FrameNet or Propbank were published.

Even the brief depiction of the work on SRL in the past eight years in the preceding passage reveals the great amount of effort the community has put in and the continuous interest the community shows in the SRL task. The SRL researchers have trained and tested the systems using either Propbank or FrameNet and have employed a wide variety of machine learning algorithms. Despite the differences in training data and in machine learning mechanisms, the SRL researchers have attempted one or more of the following four research issues since Gildea and Jurafsky (2000).

First of all, in the years between 2000 and 2008, researchers have investigated how feature designs based on the grammatical formalisms of constituency grammar and dependency grammar would help with the SRL task. The systems based on the constituency grammar formalism explored syntactic features local to the arguments as well as the dependencies among all the arguments of a verb. In the framework of constituency grammar, there are not any explicit labels for the dependencies among the arguments. The dependencies are rather the context dependence relations among the arguments and among their corresponding semantic roles. On the other hand, the systems based on the dependency grammar formalism explored explicitly the grammatical relations between words and their head words. However, finding the right syntactic information about the arguments of a predicate verb and integrating such information into an appropriate feature design remain an open issue throughout the years and up until now.

Second, in addition to investigating the appropriate feature representation in either the framework of constituency grammar or dependency grammar, SRL researchers explored a wide range of features, from the highly generic ones in earlier systems to the highly specific ones in later systems such as Punyakanok et al. (2005) and Toutanova et al. (2005). The latter features were designed to represent the syntactic structures that had not been discovered to be helpful for the SRL task by earlier systems. While introducing more features to cover the newly-discovered syntactic structures, incorporating these structure-specific features resulted in a larger feature space.

Third, in addition to the work on feature representation and feature expansion, SRL researchers have looked into how to use an appropriate statistical framework to learn from a chosen feature design. On top of building local models using different feature representations and feature expansions, researchers are concerned with how to build joint models to solve two tasks, how to jointly classify all the arguments of a verb using the context dependencies among them and how to overcome the parsing errors by jointly inferencing from the top-n parsing results from an automatic parser.

Finally, while most systems have focused on finding the proper syntactic features and on applying the right statistical framework accordingly, several systems explored how semantic knowledge, such as verb sense, and lexical semantic resources, such as WordNet, contributed to the SRL task. Unfortunately, since these systems did not demonstrate that verb sense or WordNet helped in their experiments, whether and how verb sense and lexical semantic resources can help remains an open question.

This chapter presents in detail the work on the first three research problems through several influential systems that based their feature designs on constituency grammar. At the same time, instead of showing the details of the systems whose feature designs are based on dependency grammar, this chapter introduces such systems by summarizing the systems on CoNLL-2008. Through the presentations, the current chapter achieves four goals.

First, the current chapter discovers the syntactic information that has been shown to be important to the SRL task but has not been fully explored by existing systems. Specifically, while illustrating how generic features are employed in the chosen SRL systems, this chapter focuses on examining how context dependencies among arguments are utilized in these systems built under the framework of constituency grammar. Through the examination, the current project determines to restrict the context dependency relationship to that between a core semantic role and other core semantic roles of the core arguments and extract features for the core arguments based on such dependencies. At the same time, the current system assigns generic features to non-core arguments because core arguments are verb specific but non-core arguments are not, based on the discussion in chapter 2.

Second, through examining the feature designs in existing SRL systems, the current chapter shows that the increased number of feature utilized by the systems is caused by the intentions to design specific features for the arguments in syntactic structures that involving moved or displaced arguments. These structure-specific features do not generalize the syntactic variations that a given verb appears in. This finding motivates the current project to create base argument configuration features that generalize across the syntactic variations that a verb appears in.

Third, through examining how features are integrated into the statistical learning schemes through local modeling and joint modeling in the CoNLL-2005 and CoNLL-2008 systems, this chapter shows that although joint inferencing from context dependency features and from the top-n parsing results helped the top systems on CoNLL-2005 it is not the case for CoNLL-2008 top systems where local modeling was the dominating design. This observation motivates the current system to build a local model while incorporating context dependence among the semantic roles of core arguments in the feature design.

Finally, by describing dependency grammar, this chapter aims to explore the feasibility of integrating the dependency relations in the framework of dependency grammar into the current SRL system because detecting and representing the syntactic structures involving argument movement require the knowledge of the grammatical dependency relations such as subject-verb, verb-object, modifier-head etc.

The above goals are achieved through a comprehensive description of three selected SRL systems based on constituency grammar and through a high level summary of the CoNLL-2008 systems. Due to the fact that the work in Gildea and Jurafsky (2000) and Gildea and Palmer (2002) laid the foundation for all later SRL systems, section 3.1 gives a detailed delineation of this fundamental piece of work. In the years since 2002, two systems, Punyakanok et al. (2005) and Toutanova et al. (2005), stood out for being the top two CoNLL-2005 systems. The two systems stand out also for extending the work of Gildea and Jurafsky (2000) by employing more detailed feature designs to cover the syntactic structures that affect the SRL task but have not been looked at before. Punyakanok et al. (2005) and Toutanova et al. (2005) still remain competitive among the SRL systems utilizing the constituency grammar formalism until today. A careful study of the feature designs in these systems and the statistical algorithms that integrate the features motivates the feature design and statistical framework of the current system. Therefore, section 3.2.1 highlights the system by Punyakanok et al. (2005) who extended Gildea and Jurafsky by applying linguistic constraints capturing several types of dependencies between arguments and by selecting the optimal SRL predictions from models built from different parsing results. At the same time, section 3.2.2 focuses on the system of Toutanova et al. (2005) who Toutanova et al. extended Gildea and Jurafsky by emphasizing the dependency between arguments using re-ranking-based inferencing and by incorporating new features specific to the structures such as relative clauses. Section 3.3 briefly introduces the CoNLL-2008 shared task and focuses on illustrating the dependency relations that CoNLL-2008 systems rely on. Based on the discussion in the earlier sections, section 3.4 summarizes that the context dependence among the semantic roles of the core arguments, several transformed syntactic structures, and dependency relations are further explored in the current project.

3.1 Gildea and Jurafsky (2000)

As the seminal work, Gildea and Jurafsky (2000)'s system laid the foundation for automatic SRL in a number of ways. First, the research problem of automatic SRL is formalized. Second, automatic SRL is formulated as a machine learning problem, consisting of the *argument identification* and *role labeling* subproblems. Third, the set of syntactic features utilized by Gildea and Jurafsky, in the formalism of constituency grammar, serves as the standard base features for many future SRL systems. Fourth, the authors build the system upon a linguistic theory, the **Linking Theory**, almost followed by all later SRL systems. Finally, Gildea and Jurafsky model the dependency among the semantic roles of a verb's arguments which removes the independence assumption of argument/constituent level feature design. To model the dependencies among the arguments and among their corresponding semantic roles, Gildea and Jurafsky (2002) combined argument identification and classification into one model so that the semantic roles of a target verb are assigned jointly and the most likely sequence is chosen. This combined approach is later referred to as **joint inferencing**. The rest of the section examines all the foregoing aspects of Gildea and Jurafsky's foundational work. Note that the rest of the discussion does not distinguish between Gildea and Jurafsky (2000), Gildea and Palmer (2002), and Gildea and Jurafsky (2002) since they are reports on the same system architecture.

3.1.1 Components of the learning-based system

Gildea and Jurafsky tackle the SRL problem with a machine learning approach consisting of the following two general steps:

Step I: Identify the arguments that are constituents that are semantic role recipients in the input sentences.

Step II: Label the arguments with a corresponding semantic role.

Each step of the system involves a learning and classification process which will be described in the following sections. These steps became the standard procedures for all later SRL systems.

3.1.2 Constituency grammar-based features

The features utilized in Gildea and Jurafsky (2000) and Gildea and Palmer (2002) are syntactic and lexical features based on the formalism of constituency grammar, which are made use of by the classifiers in both *argument identification* and *labeling* steps. The features have become the standard base features for later SRL systems. The six types of features that can be assigned to a candidate argument are listed below along with the shorthand for each feature:

(38) (a) the governing category (**gov**)
(b) the phrase type (**pt**)
(c) the head word (**h**)
(d) the path from the candidate to the predicate (**path**)
(e) the voice of the clause (**voice**)
(f) the position with respect to the predicate (**position**)

The features are for both the identification and classification tasks.

For instance, given input sentence 39,

(39) *The San Francisco Examiner issued a special edition around noon yesterday.*

the feature vector for its NP *the San Francisco Examiner* is instantiated as follows:

{ **gov**:*S*, **pt**:*NP*, **h**:*examiner*, **path**:$NP{\uparrow}S{\downarrow}VP{\downarrow}VBD$,
voice:*active*, **position**:*before* }

The linguistic theoretical background that motivates the syntactic features is reviewed in the next section.

3.1.3 Linguistic theoretical background

In the extension to the original work in 2000, Gildea and Palmer (2002) discuss the linguistic theoretical background behind their feature design. The following quote describes this background:

> The features used represent various aspects of the syntactic structure of the sentence as well as lexical information. The relationship between such surface manifestations and semantic roles is the subject of linking theory - see Levin and Hovav (1996) for a synthesis of work in this area. In general, linking theory argues that the syntactic realization of arguments of a predicate is predictable from semantics... exactly how this relationship works is the subject of much debate. Regardless of the underlying mechanisms used to generate syntax from semantics, the relationship between the two suggests that it may be possible to learn to recognize semantic relationships from syntactic cues, given examples with both types of information (Gildea and Palmer, 2002, p. 8) .

Relating to the feature design described in the preceding section, one can see that Gildea and Palmer suggest that the surface syntactic information can be used to recover the semantics beneath it. This is the point of view that has been adopted by almost all later SRL systems. Since the quote above is the only place where Gildea and Palmer discuss the linguistic theoretical background, a careful examination of *Linking Theory* may help to understand the feature engineering behind Gildea and Jurafsky's system as well as to shed light on alternative feature and system designs for SRL. Therefore, section 2.5.3 gives a detailed description of the *theory of argument realization*, a contemporary version of *Linking Theory*.

3.1.4 Identifying arguments

Arguments are the SR receiving constituents for a target verb ***t*** in an input sentence ***S***. Identifying the arguments is itself a training and classification process which follows the steps below:

Step 1: Parse input sentence ***S***.

Step 2: Identify constituent ***c*** from the parse.

Step 3: Calculate the likelihood of a test constituent **c** being a candidate given *path*, *t*, and *h*, using the following equation with linear interpolation:

$$P(\mathbf{c} \mid path, h, t) = \lambda_1 P(\mathbf{c} \mid path) + \lambda_2 P(\mathbf{c} \mid path, t) + \lambda_3 P(\mathbf{c} \mid path, t, h) \quad (3.1)$$

Step 4: Choose c to be a candidate if $P(\mathbf{c} \mid path, h, t)$ passes an arbitrary threshold.

The approach above correctly identifies 66% of the candidate constituents on the FrameNet test data in Gildea and Palmer (2002).

3.1.5 Labeling the candidates

The components of the labeling step are discussed in this section, including the base model, combining subset probabilities, and the back-off model.

3.1.5.1 The base model

The base model calculates the probability of a constituent filling each possible role given the syntactic and lexical features described above and the predicate t. Equation 3.2 performs this calculation.

$$P(r|h, pt, gov, position, voice, t) = \frac{\#(r, h, pt, gov, position, voice, t)}{\#(h, pt, gov, position, voice, t)} \tag{3.2}$$

This equation says to calculate the probability of a candidate c for filling the semantic role r one can simply divide the number of times r appears with t and with all five types of features by the total number of times r appears with t and all five types of features. The predicate itself and the five types of features form a full feature set for the predicate. However, it is likely that the full feature set for many candidates bearing some specific role cannot be observed in the training data, and as a result, their counts will be zero. For this reason, the base model combines probabilities of r conditioned on a variety of subsets of the full feature set, which have higher chances for being observed. Table 3.1 shows how the subsets are used in the calculation.

Feature Subset	Prob. Involving the Subset
$\{t\}$	$P(r\|t)$
$\{pt, t\}$	$P(r\|pt, t)$
$\{pt, gov, t\}$	$P(r\|pt, gov, t)$
$\{pt, position, voice\}$	$P(r\|pt, position, voice)$
$\{pt, position, voice, t\}$	$P(r\|pt, position, voice, t)$
$\{h\}$	$P(r\|h)$
$\{h, t\}$	$P(r\|h, t)$
$\{h, pt, t\}$	$P(r\|h, pt, t)$

Table 3.1: Subsets Probabilities

The probabilities are calculated from the empirical distributions in the training data. Specifically, each role and each conditioning subset and their counts in the above table are collected in a table. Then the probabilities are calculated by dividing the counts for each role by the total number of observations for each conditioning subset. For example, the probability $P(r|pt,t)$ is calculated as follows:

$$P(r|pt,t) = \frac{\#(r,pt,t)}{\#(pt,t)} \tag{3.3}$$

3.1.5.2 Combining subset probabilities

The probabilities of the subsets listed in table 3.1 are combined in several different ways to estimate the probability of a role given the full feature set, $P(r|h,pt,gov,position,voice,t)$, including equal linear interpolation, geometric mean method, and EM linear interpolation.

The basic combining method is equal linear interpolation, which averages the probabilities given by each of the subset probabilities. That is,

$$\begin{aligned} P(r|constituent) &= \lambda_1 P(r|t) + \lambda_2 P(r|pt,t) + \\ &\quad \lambda_3 P(r|pt,gov,t) + \lambda_4 P(r|pt,position,voice) + \\ &\quad \lambda_5 P(r|pt,position,voice,t) + \lambda_6 P(r|h) + \\ &\quad \lambda_7 P(r|h,t) + \lambda_8 P(r|h,pt,t) \end{aligned} \tag{3.4}$$

The equal linear interpolation method uses equal λ values as weights.

The geometric mean method is similar to equal linear interpolation when written in the log domain as follows:

$$\begin{aligned} P(r|constituent) &= \frac{1}{Z} exp\{\lambda_1 logP(r|t) + \lambda_2 logP(r|pt,t) + \\ &\quad \lambda_3 logP(r|pt,gov,t) + \lambda_4 logP(r|pt,position,voice) + \\ &\quad \lambda_5 logP(r|pt,position,voice,t) + \lambda_6 logP(r|h) + \\ &\quad \lambda_7 logP(r|h,t) + \lambda_8 logP(r|h,pt,t)\} \end{aligned} \tag{3.5}$$

The EM linear interpolation method chooses interpolation weights using the Expectation Maximization algorithm.

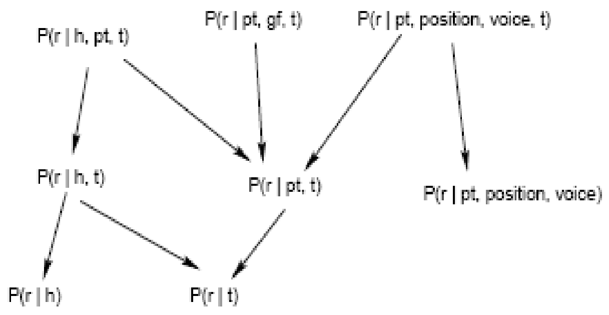


Figure 3.1: Backing off from more to less specific features

3.1.5.3 The backoff model

The backoff method builds a lattice over the subset probabilities in Table 3.1 from more specific conditioning features to less specific as shown in figure 3.1 (Gildea and Palmer, 2002, p. 17). The less specific probabilities are used only when no data are present for any more specific probabilities. This backoff approach is combined with both linear interpolation and geometric mean methods.

3.1.6 Modeling argument structure

Gildea and Palmer's system described in sections 3.1.2 through 3.1.5.3 assumes independence among arguments and classifies each argument independently of other arguments. However, the situation where a predicate with a specific meaning may take multiple argument structures challenges this assumption because it can be shown that the arguments in each argument structure are dependent on each other. The following example illustrates the dependency among a predicate's arguments. The left column in Table 3.2 lists three sample argument structures that verb *blame*, with the meaning of *assign responsibility for a fault or wrong to*, can take in FrameNet 1.3. Removing the *Cognizer* argument from the three arguments of *blame* without modifying its voice in *Dr. Farmery blames the Department of Health for causing undue alarm* would result in an ungrammatical sentence. Similarly, switching the positions of *Cognizer* and *Evaluee* would not work either.

To capture argument structures such as those shown in Table 3.2, Gildea and Palmer rely on two types of sentence-level features, *Frame Element Group* and *verb subcategorization*. The next two sections illustrate how the two sentence-level features are utilized in Gildea and Palmer's system.

3.1.6.1 Frame element group

Frame element group is a set of unordered semantic roles corresponding with the set of arguments a verb takes. The unordered set is used to capture the different roles associated with a verb's argument. The right column in Table 3.2 shows three sets of unordered semantic roles the three corresponding arguments of *blame* takes. The rest of this section describes how the *frame element group* feature is built into the system.

Example Sentences	*Frame Element Group* feature
Holman would characterise this as BLAMING [$_{Evaluee}$ the poor]	{Evaluee}
[$_{Cognizer}$ The bank] BLAMES [$_{Evaluee}$ problem debts from small business...in the south of England]	{Cognizer, Evaluee }
[$_{Cognizer}$ Dr Farmery] BLAMES [$_{Evaluee}$ the Department of Health] [$_{Reason}$ for causing undue alarm].	{ Cognizer, Evaluee, Reason}

Table 3.2: Sample argument structures of verb *blame*: 1

By combining the sentence-level *frame element groups* features with the constituent-level independent features defined in 3.1.2 and 3.1.5.1, the most likely overall semantic role assignments r^* for all candidate arguments/constituents of a sentence can be calculated using Equation 3.6

$$\begin{aligned} r^* &= \arg\max_{r_{1\ldots n}} P(r_{1\ldots n} \mid t) \prod_i P(f_i \mid r_i, t) \\ &= \arg\max_{r_{1\ldots n}} P(r_{1\ldots n} \mid t) \prod_i \frac{P(r_i \mid f_i, t)}{P(r_i \mid t)} \end{aligned} \tag{3.6}$$

$P(r_{1\ldots n} \mid t)$ is the prior for *frame element groups* of a predicate v. $P(r_i \mid f_i, t)$ gives the local probability of a candidate argument given the argument's local feature f_i, i.e. the ones defined in (38), and the predicate t. The final model in Equation 3.7 incorporates the identification model of Equation 3.1 into the model from Equation 3.6.

$$r^* = \arg\max_{r_{1\ldots n}} P(r_{1\ldots n} \mid t) \prod_i \frac{P(r_i \mid f_i, fe_i, t) P(fe_i \mid f_i)}{P(r_i \mid t)} \tag{3.7}$$

where fe_i is a binary variable indicating whether a constituent/argument is a candidate. When fe_i is true, the identification model in Equation 3.1 is plugged in for $P(fe_i \mid f_i)$; when fe_i is false, role assignment r_i for the ith candidate/argument is skipped.

3.1.6.2 Verb subcategorization

While the *frame element group* feature handles the different arguments and roles a predicate with a specific meaning can be associated with, the *verb subcategorization* feature is designed to capture the situation where the verb/predicate can assign different semantic roles to the same argument/syntactic position. The right column in Table 3.3 shows the subcategorization feature corresponding to each sentence. The subcategorization feature *VP → VB NP* captures the transitivity of the verb *open* that assigns the *Agent* role to its subject *He*. At the same time, the subcategorization feature *VP → VB* captures the intransitivity of the verb *open* that assigns *Patient* role to its subject *The door*.

Example Sentences	*Verb Subcategorization* feature
He opened the door.	VP → VB NP
The door opened.	VP → VB

Table 3.3: Sample subcategorization of verb *blame*

In their experiments, Gildea and Jurafsky combine the *subcategorization* feature with the *path* feature by adding the distributions $P(r|subcat, path, t)$ to the subsets probabilities for the base model shown in Table 3.1.

3.1.7 Performance

Gildea and Jurafsky (2000) and Gildea and Jurafsky (2002) report 82% accuracy in the task of labeling roles that are pre-identified by human annotators for the FrameNet database. They also report 65% precision and 61% recall in the task of simultaneously identifying role candidates and labeling on the FrameNet data.

3.1.8 Authors' discussion on feature design

Gildea and Jurafsky (2002) make several comments on feature design based on their experiments. First, they discover that combining the two sentence-level features, *frame element group* and *verb subcategorization* with the constituent-level features shown improves performance. Second, the use of the two sentence-level features makes some of the independent constituent-level features, such as *passive* and *position*, unnecessary. For example, removing the *passive* and *position* features when the two sentence-level features are used leads to the reported system accuracy of 82%. The authors' explanation proceeds as follows:

> ...these features, namely *passive* and *position*, overlap with the function of the *frame element group* features. Adding unnecessary features to the system can reduce performance by fragmenting the training data. (Gildea and Jurafsky (2002, p. 32))

It seems that Gildea and Jurafsky try to point out that a large number of unnecessary features may hinder performance by fragmenting the training data, the process that causes sparsity.

3.2 Extensions to the seminal work

Section 3.1 shows that Gildea and Jurafsky (2000) and Gildea and Jurafsky (2002) have conducted a comprehensive research work that pointed directions for future systems. While following the feature and system designs laid out in Gildea and Jurafsky (2000) and Gildea and Jurafsky (2002), later systems continued to investigate syntactic information and machine learning schemes other than those in the seminal work. The results from this effort are well represented in the systems on CoNLL-2005 and CoNLL-2008. Section 3.3 summarizes how CoNLL-2008 systems explored the effectiveness of the feature design based on dependency relations between words in the framework of dependency grammar. The present section illustrates how CoNLL-2005 systems extended the work of Gildea and Jurafsky (2002) by incorporating more knowledge about the dependencies between arguments into re-ranking based algorithms, by adding features to represent the syntactic structures that were not covered by earlier feature designs, and by integrating algorithms to overcome parsing errors.

Two of the top CoNLL-2005 systems, Punyakanok et al. (2005) and Toutanova et al. (2005), incorporated dependency constraints into their systems that Gildea and Jurafsky did not explore, and obtained gain in performance. Punyakanok et al. (2005), Surdeanu and Turmo (2005), and later systems such as Surdeanu et al. (2007) and Surdeanu et al. (2008b) experimented inferencing semantic roles by combining and re-ranking the results from multiple base models. To overcome the parsing errors from the automatic parser, Punyakanok et al. (2005) performed re-ranking and inferencing from the *k*-best parses from the automatic parser, the Charniak parser in their case.

The rest of the section gives each ofPunyakanok et al. (2005) and Toutanova et al. (2005) a detailed description, not only because they were the best performing systems on CoNLL-2005 but also because they still remains state-of-the-art among the systems utilizing constituency grammar. The main goal in the following discussion is to reveal two areas that have not been fully explored and are explored in the current project. First of all, the systems do not explicitly exclude non-core arguments from the context dependency features, although the discussion in chapter 2 shows that context dependency exists among core-arguments and is verb dependent and that non-core arguments are verb independent. Secondly, both systems designed features to cover the syntactic structures that were not covered by previous feature designs. As a result, more and more structure-specific features were created so that the feature space have been largely increased since Gildea and Jurafsky (2000). The following discussion also reveals that although Punyakanok et al. (2005) found joint inferencing from *k*-best parses could greatly improve the system performance Haghighi et al. (2005) dit not find that was the case.

3.2.1 Punyakanok et al. (2005)

Punyakanok et al. (2005) constructed a joint inferencing model that was the top-performing system and the only system integrating *hard* linguistic constraints on CoNLL-2005. Their system can be summarized in four stages: *pruning*, *argument identification*, *argument classification*, and *inference*. The readers are invited to read about the expanded feature sets in sections 3.2.1.2 and 3.2.1.3, the constraint-based features for several syntactic structures and context dependencies in section 3.2.1.4, and the joint inferencing from the *k*-best parses generated by Charniak parser in section 3.2.1.5.

3.2.1.1 Pruning

Following Xue and Palmer (2004), Punyakanok et al. proceed with a pruning stage that filters out the unlikely constituents of a predicate from the parse tree. The pruning stage consists of two heuristics. During initialization, the current node is set to be the predicate node. First, start the current node and collect its siblings as constituents unless the siblings are coordinated with the current node. If a sibling is a PP, also collect its immediate children. Siblings are constituents attached at the same level as the predicate. Second, move to the parent node and set it as the current node. Repeat the first heuristic until the top level node is reached.

The heuristics work effectively with correct parse trees. To handle the cases of incorrect *PP* attachment from automatic parsers, Punyakanok et al. consider any consecutive *NP* and *PP* as constituents, as well as the split of *NP* and *PP* inside the *NP* returned by the two heuristics.

3.2.1.2 Argument identification

Argument identification relies on a sparse network of linear binary classifiers, the weights of which are updated by a regularized variation of the Winnow multiplicative update rule. The activation score from a linear classifier can be converted to a conditional probability using the soft function in equation 3.8.

$$Prob(i) = \frac{e^{act_i}}{\sum_{1 \leq j \leq n} e^{act_i}} \tag{3.8}$$

If there are n classes of role labels and the raw activation score of class i, is act_i, then $Prob(i)$ is the posterior estimation for class i.

The constituents resulted from the *pruning stage* are classified using the linear binary classifiers trained on the following features. The conditional probabilities are calculated as needed.

3.2.1.3 Argument classification

Argument classification relies on a sparse network of linear binary classifiers constructed like those for *argument identification*. Conditional probabilities for classification are calculated using equation 3.8. In addition to the features introduced in the previous section, the features below are also used to built the classifiers:

- syntactic frame, the sequential pattern of the noun phrases and the predicate in the sentence, introduced by Xue and Palmer (2004).
- propositional phrase head, the head of the first phrase after the preposition inside PP
- NE indicating if the target argument is, embeds, overlaps, or is embedded in a named-entity along with its type

3.2.1.4 Inference

The inference stage optimizes the classification results for the set of arguments of a sentence from the *argument classification* stage by maximizing the objective function in equation 3.9 ,

$$\hat{c}^{1:M} = argmax_{c^{1:M} \in \mathscr{P}^M} \sum_{i=1}^{M} Prob(S^i = c^i) \tag{3.9}$$

subject to a series of linguistic and structural constraints as follows

- No overlapping or embedding arguments;

- No duplicate argument classes for A0-A5;
- Exactly one V argument per predicate;
- If there is a C-V, there must be V-A1-C-V pattern;
- If there is an R-arg, there must be arg somewhere;
- If there is a C-arg, there must be arg somewhere before it;
- Each predicate can take only core arguments that appear in its frame file.

The conditional probabilities $Prob(S^i = c^i)$ are output from the *argument classification* stage. The objective function performs sentence-level global optimization with respect to the constraints. The next section describes the joint inference model that implements equation 3.9.

3.2.1.5 Joint inference with Multiple SRL Systems

The sentence-level global optimization is combined with SRL outputs from several parses of the same sentence. This section introduces the model that carries out the joint inferencing. For each test sentence, its *k*-best parse outputs by the Charniak parser are used to build *k* separate SRL classifiers. A joint inference of the *k* classifiers on *k* argument sets, $\{S_1, \cdots, S_k\}$ is achieved by optimizing the same objective function as 3.9, shown in 3.10 :

$$\hat{c}^{1:M} = argmax_{c^{1:M} \in \mathscr{P}^M} \sum_{i=1}^{M} Prob(S^i = c^i) \tag{3.10}$$

where $S^{1:M} = \bigcup_{i=1}^{k} S_i$ and

$$Prob(S^i = c^i) = \frac{1}{k} \sum_{j=1}^{k} Prob_j(S^i = c^i)$$

where $Prob_j$ is the probability output by system j.

3.2.1.6 System Performance

Punyakanok et al.'s system obtained the highest score in the competition of the CoNLL-2005 shared task with F-measures of 79.44% over the TreeBank data, 67.75% over the Brown Corpus, and an overall of 77.92% over both data sets.

3.2.2 Toutanova et al. (2005)

Following Gildea and Palmer (2002), Toutanova et al. (2005) not only built a two-phase system that performs *identification* and *classification* separately but also built a joint inferencing model that combines identification and classification into one step. The system by Haghighi et al. (2005) was built on the same framework as Toutanova et al. (2005), but was augmented with two new features to model dependencies among core arguments and five new feature to model arguments' surface syntactic structures . Sections 3.2.2.1 through 3.2.2.3 introduce the system in Toutanova et al. (2005), and section 3.2.2.6 describes the extensions in Haghighi et al. (2005).

3.2.2.1 Identification

Identification is the task of identifying constituents from parsing as candidate semantic role bearing arguments. During the identification phase, Toutanova et al. classify a parse tree (t) node n_i as either ARG, an argument (including modifiers), or NONE, a non-argument, while the computed probability P_{ID} for identifying this n_i incorporated in both local and joint classifiers.

3.2.2.2 Local Classifiers

Let L be a set of semantic roles matching the nodes in the parse tree t. Let $Id(L)$ be the set of identified non-NONE arguments. The goal of a local classifier is to assign a semantic role label to l_i ($l_i \in L$) corresponding to the ith non-NONE argument in $Id(L)$, while satisfying two constraints. The first constraint is the independence constraint. The system assumes independence between each node n_i in the parse tree t , and hence assumes independence between the non-NONE arguments in $Id(L)$ and between the semantic roles in L. The second constraint is the non-overlapping constraint, which says that the arguments cannot overlap.

The local classifier calculates the probability of the semantic role labeling L given parse tree t and the predicate v by multiplying an identification model P_{ID} and a classification model P_{CLS} given the same t and v, shown in Equation 3.11.

$$P_{SRL}(L|t,v) = P_{ID}(Id(L) \mid t,v) \times P_{CLS}(L|t,v,Id(L)) \tag{3.11}$$

Enforcing the independence constraint between the non-NONE arguments and that between their labels is equivalent to independently maximizing the product of the probabilities of the two models in Equation 3.11 as shown in Equation 3.12,

$$P^{l}_{SRL}(L|t,v) = \prod_{n_i \in t} P_{ID}(Id(l_i) \mid t,v) \times \prod_{n_i \in t} P_{CLS}(l_i|t,v,Id(l_i)) \tag{3.12}$$

where l_i corresponds to the label for node n_i or the ith non-NONE argument in $Id(L)$. The base features, referred to as **local features**, utilized by Toutanova et al. (2005) are listed as follows:

- Standard Features (Gildea and Jurafsky, 2002)
  - phrase type: Syntactic Category of node
  - predicate lemma: Stemmed Verb
  - path: Path from node to predicate
  - position: Before or after predicate?
  - voice: Active or passive relative to predicate
  - head word of phrase
  - sub-cat: CFG expansion of predicate's parent
- Additional Features (Pradhan et al., 2004)
  - head POS
  - first/last phrase-type/word/POS
  - left/right sister phrase-type/POS:
  - parent phrase-type/POS
  - ordinal tree distance: Phrase Type with appended length of PATH feature
  - node-LCA partial path: Path from constituent to Lowest Common Ancestor with predicate node
- Selected Pairs (Xue and Palmer, 2004)
  - predicate lemma & path
  - predicate lemma & head word
  - predicate lemma & phrase type
  - voice & position

A dynamic programming algorithm, resembling the Viterbi algorithm for context-free grammar, is applied to satisfy the non-overlapping constraint. This algorithm starts from the leaves of the tree t and finds the best label assignment for the tree, using already computed labeling for its children. Each subtree t_i of t stores the most likely assignment up to t_i as well as the log-probability of the labeling of all nodes it dominates. Calculating the log-probabilities is a modification to Equation 3.12. Hence, the most likely labeling L for t is the maximum of the following two sums:

- The sum of the log-probabilities of the most likely assignment of the children subtrees t_1, ..., t_k plus the log-probability for assigning the node t to NONE.
- The sum of the log-probabilities for assigning all of t_i's nodes to NONE plus the log-probability for assigning the node t to ARG.

3.2.2.3 Joint classifiers

A discussion of the motivations for joint classifiers precedes the description of the joint classifiers. The joint classifiers are built for three reasons. First, there are dependencies between the roles that arguments bear. That is, there exist priorities among a verb's arguments with respect to the semantic roles they may bear. For example, an *instrument* cannot be assigned to subject in the presence of an *agent*, such as that in (40).

(40) $*[_{Instrument}$ The key] opened the door by $[_{Agent}$Susan].

Several systems, including Gildea and Palmer (2002), Thompson et al. (2003), and Pradhan et al. (2004), have modeled such dependencies and have shown performance gain for the modeling. Toutanova et al. (2005) continues the same tradition by defining two feature templates, *candidate argument sequence* and *whole label sequence*, that maintain the dependency across arguments. The *candidate argument sequence* (42) and *whole label sequence* (43) correspond to the dependency relationship between the arguments in example (41).

(41) $[_{NP1ARG1}$ Final-hour trading $]_{VBD1}$ accelerated $[_{PP1ARG4}$ $[_{TO1}$ to $[_{N2}$ 108.1 million shares]] $[_{NP3ARGM-TMP}$ yesterday]]

(42) [NP1-ARG1, VBD1-PRED, PP1-ARG4, NP3-ARGM-TMP]

(43) [voice:active ARG1, PRED, ARG4, ARGM-TMP]

The second reason for building joint classifiers is that Toutanova et al. show that the joint information about a verb's argument structure, such as the dative alternation, is important for the labeling task. Given examples (44) and (45) from Toutanova et al. (2005):

(44) $[_{ARG0}$ Shaw Publishing] offered $[_{ARG2}$Mr. Smith] $[_{ARG1}$ a reimbursement].

(45) $[_{ARG0}$Shaw Publishing] offered $[_{ARG1}$a reimbursement] to $[_{ARG2}$Mr. Smith].

(44) and (45) display different argument structures in spite of the same verb with the same meaning. The joint information between the verbs *offer* and their corresponding argument structure are captured by the respective feature templates in (46) and (47).

(46) [voice:active ARG0, PRED, ARG2, ARG1].

(47) [voice:active ARG0, PRED, ARG1, ARG2].

The elements in the templates are ordered to ensure the order of the arguments in the original sentence. The type of features are called *frame features* by the authors.

The third motivation for joint classifiers is that automatic parsers utilized for the identification phase do not achieve 100% accuracy in parsing, which affects the identification performance. Collins and Koo (2005) propose joint inference by re-ranking that improves parsing performance. Toutanova et al.'s follow this tradition and propose a re-ranking based joint model as follows.

Re-ranking based joint inference algorithm:

Let $\Phi(t,v,l)$ be a feature map from a tree t, target verb v, and joint label assignment L of the nodes of t, to the vector space $\mathbb{R}^s$. Let L_1, L_2, ... L_N denote top N joint label assignments.

Given: parsed training and test sentences, the corresponding local features, *candidate argument sequence* features, *whole label sequence* features, and *frame* features

Step 1: Generate top N label assignments to nodes in t according to a local model in Equation 3.12, repeated below:

$$P^l_{SRL}(L|t,v) = \prod_{n_i \in t} P_{ID}(Id(l_i) \mid t,v) \times \prod_{n_i \in t} P_{CLS}(l_i|t,v,Id(l_i))$$

Step 2: Learn a parameter vector W with w_i be the weight of the ith dimension of features in $\Phi(t,v,l)$, using a log-likelihood model;

Step 3: Calculate the probability of L using the re-ranking model:

$$P^r_{SRL}(l \mid t,v) = \frac{e^{\langle}\Phi(t,v,L),W\rangle}{\sum^N j = 1e^{\langle}\Phi(t,v,L_j),W\rangle} \tag{3.13}$$

Step 4: Calculate the final probability of L by combining the local model from Step 1 and the joint model from Step 3 as follows:

$$P_{SRL}(L \mid t,v) = (P^l_{SRL})(L \mid t,v)^{\alpha} P^r_{SRL}(l \mid t,v) \tag{3.14}$$

where α is tunable for the amount of influence of the local model on the final model.

Step 5: Calculate the probability of test label assignment L using Formula 3.15:

$$\arg \max_{L \in \{L_1,\ldots,LN\}} \alpha log P^l_{SRL}(L \mid t,v) + log P^r_{SRL}(l \mid t,v) \tag{3.15}$$

3.2.2.4 Repeated core argument label feature

The preceding section describes the two features intended to capture the dependencies among arguments, namely the *candidate argument sequence* feature in (42) and the *whole label sequence* feature in (43), (46), and (47). This section introduces the *repeated core argument* feature that was used in Toutanova et al. (2005) but was reported in Haghighi et al. (2005). This feature captures the cases where the WHNP in a relative clause and the noun phrase it refers to share the same semantic role. In example (ex:rep), the WHNP *which* and the NP it refers to *core businesses* both the arguments of *include* and both take the semantic role of *A2*, meaning *group*.

(48) The move also would allow the company to concentrate on $[_{A2}$core businesses$]$, $[_{A2}$which$]$ include ceramic tile ...

Although the authors themselves did not give an example for this feature, it indicates that the authors were aware of the syntactic structure of relative clauses and designed relevant features to the structure.

3.2.2.5 System performance

Toutanova et al. (2005) is ranked the second best in the competition of the CoNLL-2005 shared task, with F-measures of 78.63% over the TreeBank test data, 68.44% over the Brown Corpus, and an overall of 77.30% over both data sets.

3.2.2.6 Extensions by Haghighi et al. (2005)

The system in Toutanova et al. (2005) preceded that in Haghighi et al. (2005). Since both systems were built by the same authors, the time between the two systems allowed authors to perform error analyses and design new features to handle the error cases. Among several new features, the authors integrated the **projected path** local feature to represent the subject of the verb complement in the control-verb structure.

- Projected path - path from the maximal extended projection of the predicate to the argument. The maximal projection is the highest VP in the chain of VP's dominating the predicate.

In (49), the verb complement *widen* shares the same subject/argument *the trade gap* with the control verb *expected*. The projected path feature, $VP{\uparrow}S{\downarrow}NP$, from the highest VP *is ... widen* dominating the predicates *expected* and *widen*, to the argument *the trade gap*, is shared by both verbs and hence captures the fact that both verbs share the same subject/argument.

(49) $[_S\ [_{NP}$ The trade gap$]\ [_{VP}$ is $[\ [_{VP}$ expected $[_{VP}$ to widen $]\ ]\ ]\ ]\ ]$.

Together with the features listed below, Haghighi et al. (2005) were able to increase the overall F-measure of the $A0$ role from 81.02% to 83.08% for the local classifier.

- Missing subject - binary feature indicating if a predicate's subject is its left sister node
- Projected path & Missing subject - combination of the *missing subject* and the *projected path* features

Although Haghighi et al. (2005) tried to improve the system by jointly inferencing from the k-top parses from an automatic parser, the effort did not bring any gain in performance.

3.2.3 Section summary

The preceding two sections illustrate that Gildea and Jurafsky (2000) initiated the research work in essentially all the areas that later systems continue to research on, including experimenting with more syntactic features based on constituency grammar, integrating more constraints regarding the dependencies among arguments and their corresponding semantic roles, combining identification and classification into a joint inferencing model, and maximizing from different parsing results to overcome the parsing errors through a joint inferencing model. Thus, such trends indicate that SRL researchers paid attention to more detailed information/knowledge about arguments and semantic roles. This suggests that integrating detailed syntactic information is beneficial. Around 2006/2007 SRL researchers indeed started to investigate more detailed syntactic information but in an alternative grammar formalism, **dependency grammar**. Encouraged by the work such as Johansson and Nugues (2007a) who showed that the rich syntactic dependencies between words and their head words could improve SRL labeling, SRL researchers intended to investigate whether dependency grammar-based representation is *better* that constituency grammar-based representation. Before this new research trend displayed in CoNLL-2008 is reviewed, it is important to realize from the preceding discussion that previous systems have not distinguished between the feature representations of core and non-core arguments, that syntactic structure-specific features do not generalize across the all structures where movements have occurred, and that joint inferencing from the k-best parses does not necessarily help with the classification task.

3.2.4 Two variations to the preceding systems

As illustrated in the previous sections, the SRL systems that are based on the constituency grammar-based feature representation more or less follow the two paradigms proposed in Gildea and Jurafsky (2000), where a SRL system either follows the two-phase argument identification-classification approach or jointly inferences from identification and classification with or without the re-ranking option. This section introduces two systems that do not exactly follow the two foregoing paradigms. To avoid breaking the continuity in the previous sections, the systems are presented in this last sub-section of section 3.2.

3.2.4.1 SRL through sentence simplification

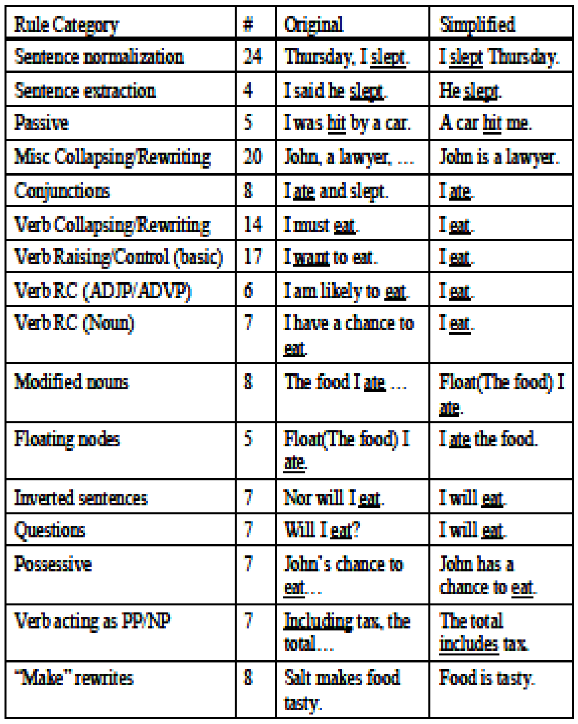
Vickrey and Koller (2008) is the first of the two systems that does not follow the two traditional SRL paradigms. The core components are summarized as follows:

1. For each verb in a sentence, apply 154 hand-written simplification rules to form a simple sentence corresponding to the verb.
2. Train a maxent SRL system using the simplified sentences upon a set of predefined features.
3. Given a new simplified sentence, predict the semantic roles of the constituents of each verb.

Note that the preceding approach differs from the traditional approaches in three aspects. First, a complex set of procedures transforms a sentence into simple sentences corresponding to each verb. A **simple sentence** is defined as a canonical form that is a format shared by all sentences. Figure 3.2 shows that such simplified format is close to the simple active statements the current project uses. As a matter of fact, this is the first application of transformation to the SRL task in literature. Second, the training and testing processes are conducted on the simplified sentences. Third, there is no procedure to identify the candidate arguments. Whatever constituents that are preserved through the simplification process are assigned a semantic role label.

Figure 3.2 lists a subset of the 154 sample simplification rules by category (Vickrey and Koller, 2008, p. 2). Figure 3.3 shows the rules applied to derive the simple sentence *I ate* from the original sentence *I was not given a chance to eat*.

Vickrey and Koller (2008) utilize three types of features for the simplified sentence. The first type of features consists of the rules that were used to obtain the simple sentence. The second type of features correspond to the label patterns of the arguments associated with a specific verb. The third type of features include common features found in literature, such as constituent label, head word of the constituent, etc. For example, table 3.4 lists the features for the simplified sentence *John gave me a sandwich* from the original passive



Rule Category	#	Original	Simplified
Sentence normalization	24	Thursday, I slept.	I slept Thursday.
Sentence extraction	4	I said he slept.	He slept.
Passive	5	I was hit by a car.	A car hit me.
Misc Collapsing/Rewriting	20	John, a lawyer, ...	John is a lawyer.
Conjunctions	8	I ate and slept.	I ate.
Verb Collapsing/Rewriting	14	I must eat.	I eat.
Verb Raising/Control (basic)	17	I want to eat.	I eat.
Verb RC (ADJP/ADVP)	6	I am likely to eat.	I eat.
Verb RC (Noun)	7	I have a chance to eat.	I eat.
Modified nouns	8	The food I ate ...	Float(The food) I ate.
Floating nodes	5	Float(The food) I ate.	I ate the food.
Inverted sentences	7	Nor will I eat.	I will eat.
Questions	7	Will I eat?	I will eat.
Possessive	7	John's chance to eat...	John has a chance to eat.
Verb acting as PP/NP	7	Including tax, the total...	The total includes tax.
"Make" rewrites	8	Salt makes food tasty.	Food is tasty.

Figure 3.2: Sample simplification rules

sentence *I was given a sandwich by John*.

This system obtains an overall of F-measure of 77.4% on WSJ section 23 using the preceding simplification-based approach.

3.2.4.2 SRL through combination strategies

Surdeanu et al. (2007) benefit from independent SRL systems by combining three base models into a complex system. The rest of the section starts with a brief description of each of the three base models and then follows up with a description of the three combination strategies. The performance of the three independent models and the best combination

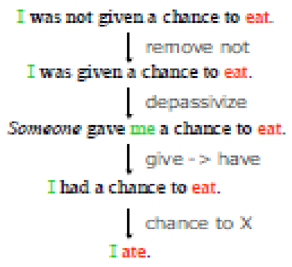

I was not given a chance to eat.
remove not
I was given a chance to eat.
depassivize
Someone gave me a chance to eat.
give -> have
I had a chance to eat.
chance to X
I ate.

Figure 3.3: Rules applied to derive *I ate*

Feature type	Examples
Pattern	{ ARG0 = Subj NP, ARG1 = PV NP2, ARG2 = PV NP1 }

Common	Role = ARG0, Head Word = John
	Role = ARG1, Head Word = sandwich
	Role = ARG2, Head Word = I
	Role = ARG0, Category = NP
	Role = ARG1, Category = NP
	Role = ARG2, Category = NP
	Role = ARG0, Position = Subject NP
	Role = ARG1, Position = Postverb NP2
	Role = ARG2, Position = Postverb NP1

Table 3.4: Sample features from Vickrey and Koller (2008)

strategy is shown in table 3.5.

The first and second base models model *the SRL problem as a sequential tagging task, where each semantic argument is matched to a sequence of non-embedding phrases* (Surdeanu et al., 2007, p.112). The first model utilizes partial syntax (chunks and clause boundaries). The second model instead uses full syntax instead. The third model follows the traditional approach, where semantic role labels are assigned to identified semantic arguments. All three models train the one-vs-all AdaBoost classifier. As shown in table 3.5, the F-measures of the base models are around 76%.

The true power of the system by Surdeanu et al. (2007) comes from the strategies that combine the base models. An overview of the system is shown in figure 3.4. A brief description of each combination strategy follows.

Inference with constraint satisfaction The first combination model has no parameters to estimate; it only makes use of the argument probabilities output by the individual models and constraints over argument structures to build the overall solution for each sentence.

Inference with local learning The second approach implements a cascaded inference model with local learning: first, for each type of argument, a classifier trained offline decides whether a candidate is or is not a final argument. Next, the candidates that passed the previous step are combined into a solution consistent with the constraints over argument structures.

Global learning The third inference model is global: a number of online ranking functions, one for each argument type, are trained to score argument candidates so that the correct argument structure for the complete sentence is globally ranked at the top.

Among the three strategies, the strategy of inference with local learning yields the highest performance, with an F-measure of 80.56% as shown in table 3.5.

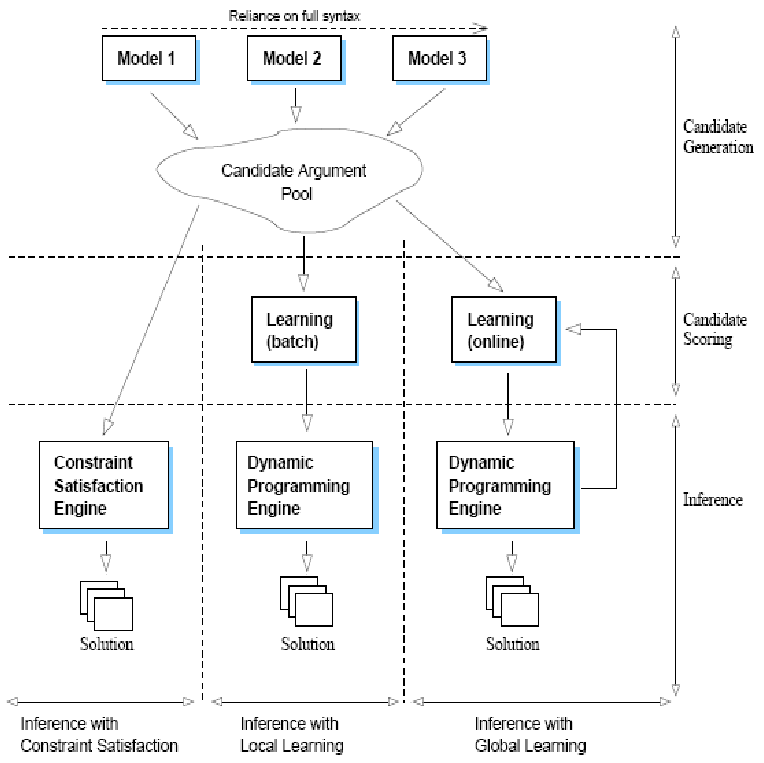

Reliance on full syntax
Model 1
Model 2
Model 3
Candidate Argument Pool
Candidate Generation
Learning (batch)
Learning (online)
Candidate Scoring
Constraint Satisfaction Engine
Dynamic Programming Engine
Dynamic Programming Engine
Inference
Solution
Solution
Solution
Inference with Constraint Satisfaction
Inference with Local Learning
Inference with Global Learning

Figure 3.4: Combination strategies

Model Name	Precision	Recall	F-measure
1	78.76	72.44	75.47
2	79.65	74.92	77.21
3	80.32	72.95	76.46
Local learning +1+2+3	87.47	74.67	80.56

Table 3.5: Performance of base and combined models

3.3 Exploring dependency grammar: onto CoNLL-2008

The CoNLL-2008 shared task, held between February, 2008 and May, 2008, subsumes two tasks. The first task is to perform semantic role labeling. The second task is to perform syntactic parsing for syntactic dependencies. The second task will not be discussed in the rest of the section because it is an irrelevant task to the current project.

The SRL task consists of two separate challenges. The first challenge is the **closed challenge** where participating teams are not allowed to utilize resources other than the training data and part-of-speech and dependencies between head words are the only syntactic representations in the training data. The teams will not have access to constituent-based syntactic representations. The second challenge is the **open challenge** where the teams are allowed to use any kind of external tools and resources, including constituent-based syntactic representations. At the same time, the shared task organizers encourage systems to utilize semantic resources, such as WordNet, VerbNet, named entities, or a word sense disambiguation system. The rest of this section describes the motivations for the two challenges, illustrates the dependency grammar-based features, and analyzes the system designs and results.

The purpose of describing the motivations of CoNLL-2008 systems, illustrating the dependency grammar-based feature representation, and discussing the systems designs is three-fold. First of all, the linguistic theoretical background in dependency grammar is provided, which helps with the feature design in the current system shown in section #. Secondly, the results from the CoNLL-2008 shared task do not show that dependency grammar-based representation better suits the SRL task than the constituency grammar-based representation. This further motivates the current system to investigate the constituency grammar for the SRL task. Finally, unlike the CoNLL-2005 systems, CoNLL-2008 systems could not verify that the joint modeling of the identification and classification tasks boost the performance. This observation motivates the current system to stick with a local modeling architecture.

3.3.1 Motivations for the challenges

Since CoNLL-2005, researchers continue to investigate the options in feature design and learning algorithms. However, the systems published since CoNLL-2005 until 2007, such as that by Gordon and Swanson (June 23-30, 2007) on ACL-2007, could not pass the F-measures achieved by the top performing systems from the CoNLL-2005 competition. This motivates researchers to investigate the causes.

The common data sets used for training and testing in the SRL systems are the PropBank and the FrameNet database. The kappa statistics or inter-annotator agreement for the PropBank over semantic roles is 93% or higher (Palmer et al., 2005). (The kappa statistics for the FrameNet database is not available. The availability of this statistics may not be relevant because most systems are built with the PropBank.) Therefore, the data per se should not be the cause for the lack of improvement in performance, and the SRL task on the PropBank data is a solvable problem. Realizing this, SRL researchers have turned to searching for other causes.

While searching for other causes, SRL researchers have raised doubt about the use of constituency grammar-based features and have shifted their interests to dependency grammar-based features, which are syntactic features based on dependency relationships between head words and modifiers in a sentence (see section 3.3.2 for a more detailed description). The work by Johansson and Nugues (2007a), Johansson and Nugues (2007b) and Surdeanu et al. (2008b) have generated enthusiasm in the field to replace the constituency grammar features with the dependency grammar features. Based on previous research work, the organizers of the CoNLL-2008 shared task of SRL formalize three research questions for the **SRL** task. First, due to the fact that the CoNLL-2005 top systems utilized joint inferencing/modeling to maximize for the best semantic role sequence and to handle parsing errors in the framework of constituency grammar, CoNLL-2008 encouraged the community to investigate joint inferencing in the framework of dependency grammar. The second research question is to show whether dependency grammar is a better representation for the SRL task than constituency grammar. The third research question is to investigate how to incorporate semantic information such as word sense disambiguation, and semantic resources such as WordNet and named entities. The third research question has been a heritage question since CoNLL-2005. In order to have a deep understanding of the CoNLL-2008 systems, it would be necessary to give a description of the linguistic theory behind the systems in the next section.

3.3.2 Dependency grammar and data representation

The organizers of CoNLL-2008 refer to the Treebank-style representation of language as constituency grammar-based representation and promote dependency grammar-based format for the CoNLL-2008 competition because they assume that dependency grammar provides for the researchers a simpler representation of the data and yet a richer set of grammatical relations. The following brief account helps to illustrate their rationale.

Dependency grammar, evolved in the work of Tesniere, Hudson, and Sgall (Nivre, 2007, p 46), traditionally accounting for syntactic structures in Classical and Slavic languages, includes two main components.

Component I Dependency grammar assumes that syntactic structure consists of lexical items.

Component II The lexical items are linked by binary asymmetrical relations called dependencies.

The preceding two components indicate that the dependency between the grammatical units is a relation strictly between two lexical items and that there are no intermediate nodes between two words that are related by some dependency relation. As a result, dependency grammar does not entail the concept of constituency or phrase structure. Figure 3.5 displays the difference between the constituent and dependency representations of the

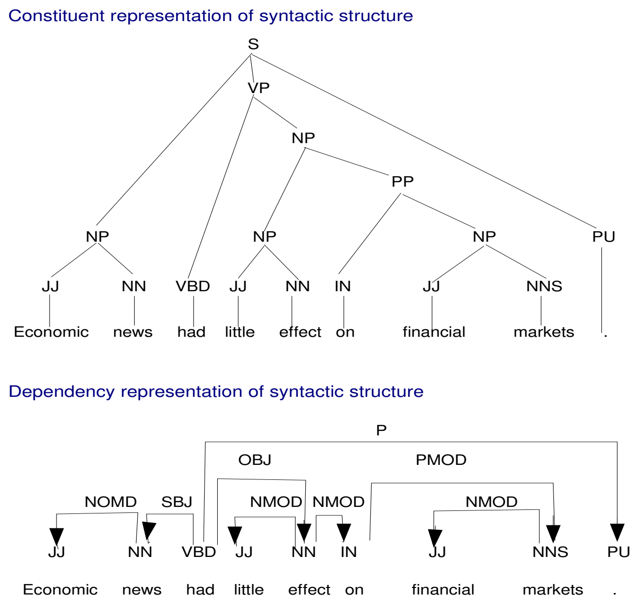


Figure 3.5: Constituent vs. dependency representation of syntactic structure

syntactic structure of the same English sentence (Nivre, 2007, p 11). In the lower panel of figure 3.5, a directed edge points to the head of a word. And the label on the edge represents a particular dependency relation between the child and the head. A dependency relation is in fact a grammatical relation between a head word and the child.

CoNLL-2008 systems in the closed challenge make extensive use of the dependency relations in their **feature engineering**. Table 3.6 lists all the dependency relations appearing in the CoNLL-2008 data, which are converted from the constituent representation in the Treebank (Surdeanu et al., 2008b) .

The next section summarizes the system designs and explains whether the research goals have been met.

Label	Description
NMOD	Modifier of nominal
P	Punctuation
PMOD	Modifier of preposition
SBJ	Subject
OBJ	Object
ROOT	Root
ADV	General adverbial
NAME	Name-internal link
VC	Verb chain
COORD	Coordination
DEP	Unclassified
TMP	Temporal adverbial or nominal modifier
CONJ	Second conjunct (dependent on conjunction)
LOC	Locative adverbial or nominal modifier
AMOD	Modifier of adjective or adverbial
PRD	Predicative complement
APPO	Apposition
IM	Infinitive verb (dependent on infinitive marker to)
HYPH	Token part of a hyphenated word
HMOD	Token inside a hyphenated word
SUB	Subordinated clause
OPRD	Predicative complement of raising/control verb
SUFFIX	Possessive suffix
DIR	Adverbial of direction
TITLE	Title (dependent on name)
MNR	Adverbial of manner
POSTHON	Posthonorific modifier of nominal
PRP	Adverbial of purpose or reason
PRT	Particle (dependent on verb)
LGS	Logical subject of a passive verb
EXT	Adverbial of extent
PRN	Parenthetical
EXTR	Extraposed element in cleft
DTV	Dative complement (to) in dative shift
PUT	Complement of the verb put
BNF	Benefactor complement (for) in dative shift
VOC	Vocative

Table 3.6: Dependency relations used on CoNLL-2008

3.3.3 Results from CoNLL-2008

This section summarizes how CoNLL-2008 systems answered the research questions on integrating WSD, joint inferencing, and incorporating the dependency grammar-based feature design into different machine learning schemes.

Twenty-four systems submitted results to CoNLL-2008. Five of the twenty-four systems participated in the open challenge, while the remaining nineteen in the closed challenge. Although systems in the open challenge were encouraged to utilize word sense disambiguation techniques and/or semantic lexical resources, such as WordNet, none of the systems did so. Thus, how word sense disambiguation can help the SRL task remains an open question.

During CoNLL-2005, the top four systems relied on joint inferencing/modeling to select the most likely semantic role sequence for a given predicate from multiple role sequences corresponding to one parse tree, e.g. Toutanova et al. (2005), or from *k*-best parse trees, e.g. Punyakanok et al. (2005), while combing argument identification and classification (see sections 3.1, 3.1.6, 3.2.1.5 and 3.2.2.3). In the CoNLL-2008 closed challenge, although nine out of the nineteen systems performed joint inferencing by combining argument identification and classification, only one out of the top five systems built a joint model (Johansson and Nugues, 2008; Surdeanu et al., 2008a). However, the improvement from the joint modeling is very small in Johansson and Nugues (2008). The other top four models built pipeline models that separate argument identification and classification steps. This shows that joint modeling is not a trivial task.

In terms of the choice of machine learning techniques, the systems choose from one of three approaches, namely, maximum entropy, support vector machines, and the perceptron algorithm as the base classifier.

The average F-measure of the top five CoNLL-2008 systems is 79.86% on the verb arguments, whereas that of the top five CoNLL-2005 systems is 77.47%. However, the results from the two events are not comparable because the CoNLL-2008 evaluation method only requires the systems to identify the head word of the arguments, but the CoNLL-2005 evaluation scheme requires argument boundary detection. The differences in the evaluation make it difficult to directly compare the results. In fact, the CoNLL-2008 research question on whether the dependency-based representation is better for SRL remains an open question.

3.4 Chapter summary

The preceding sections demonstrate SRL researchers' effort to discover the appropriate syntactic knowledge for the feature design and to develop the proper machine learning schemes for the chosen feature design. Along the line of discovering the right feature design for the SRL task, the researchers have explored features based on the constituency grammar formalism and the dependency grammar formalism respectively. In terms of the feature engineering based on constituency grammar, the researchers experimented with surface syntactic features as well as the context dependency features among the arguments of a verb, and concluded that the context dependencies and structure-specific features, such as *projected path* were helpful. In terms of the feature engineering based on dependency grammar, results from CoNLL-2008 indicate that the grammatical relations between words and their heads are helpful. But the researchers participated in CoNLL-2008 could not conclude that features based on dependency grammar are better than those based on constituency grammar.

Based on the above review of the SRL systems in the past eight years and on one's intuition of the English language, one can make several observations that reveal the areas that the current project looks into.

First, before SRL researchers shifted their interests to dependency grammar, they had to design new features to cover the syntactic structures that previous feature designs did not handle. They discovered these syntactic structures through error analyses. Section 3.2.1.4 shows how Punyakanok et al. (2005) imposed seven constraints on seven types of surface syntactic structures. Discovering that the previous systems had not properly handled the cases where two consecutive arguments could take the same semantic role in relative clauses and the noun phrase it modifies, Toutanova et al. (2005) created the *Repeated core argument label* feature (see section 3.2.2.4). Moreover, Haghighi et al. (2005) came up with the *projected path* feature to handle the moved subject in the control-verb structures (see 3.2.2.6). It seems to be the case that the features are more and more engineered towards specific syntactic structures. As a result, the feature space keeps expanding. In the original work of Gildea and Jurafsky (2002), fewer than 10 feature types were used. Punyakanok et al. (2005) used about 18 features for classification, in addition to 7 constraints. Haghighi et al. (2005) employed about 23 feature types. The increasing feature types indicate the discovering of more syntactic structures that researchers did not realize before. It is necessary to discover the syntactic structures that have not been explored before and design the appropriate features to cover accordingly. While designing structure-specific feature may improve the performance, it may also increase the feature space and fragment the training data. It is then necessary to be able to generalize from the different syntactic structures/configurations.

Second, as more syntactic structures that need specific feature engineering are discovered, it is important to know what other syntactic structures are out there that cannot be handled by the existing feature designs in literature.

Third, the above survey raises one's awareness of the important role context dependency plays in the SRL task. Gildea and Jurafsky (2000) first utilized this type of knowledge, without including the relative positions between the arguments and the verb. Example (51) shows the dependency feature that Gildea and Jurafsky (2000) would have had for the same verb *accelerated* in (50). This dependency feature is referred to as the *frame element group* feature by the authors in section 3.1.6.1. Toutanova et al. (2005) extended the dependency representation by including the relative positions and verb's voice in the *whole label sequence* feature. Example (53) lists all the arguments' labels for the verb *accelerated*, including the *ARGM-TMP* for the non-core argument *yesterday*. The *candidate argument sequence* feature also includes the verb's position in the sequence. Examples (52) lists all the arguments and their corresponding semantic role labels for the verb *accelerated*. According to the authors, the sequences (52) and (53) capture the dependency among the arguments. However, it is not clear that whether the authors implied that the non-core argument *yesterday* was also dependent on the core arguments. This may not be the case because in (54) *yesterday* is moved to the sentence initial position and its *ARGM-TMP* role has not been changed. Although the authors later reported in Haghighi et al. (2005) that the *whole sequence of core arguments* feature which only lists the core arguments of a verb were used in Toutanova et al. (2005), it is not clear how this feature had interacted with the *candidate argument sequence* and *whole label sequence* features where non-core arguments are listed. Therefore, if the dependencies among arguments are used as features to constrain the number of arguments the verb can take as well as the semantic role each argument position takes, then it may be necessary to handle the core and non-core arguments differently because some non-core arguments, such as the temporal argument *yesterday* above, are independent of other arguments' positions and semantic roles.

(50) $[_{NP1ARG1}$ Final-hour trading $]_{VBD1}$ accelerated $[_{PP1ARG4}$ $[_{TO1}$ to $[_{N2}$ 108.1 million shares]] $[_{NP3ARGM-TMP}$ yesterday]]

(51) [ARG1, ARG4, ARGM-TMP]

(52) [NP1-ARG1, VBD1-PRED, PP1-ARG4, NP3-ARGM-TMP]

(53) [voice:active ARG1, PRED, ARG4, ARGM-TMP]

(54) $[_{NP3ARGM-TMP}$ yesterday] $[_{NP1ARG1}$ Final-hour trading $]_{VBD1}$ accelerated $[_{PP1ARG4}$ $[_{TO1}$ to $[_{N2}$ 108.1 million shares]]]

As summarized in the three observations above, the current project investigates how the dependency among arguments can be appropriately represented, what syntactic structures need to be further explored and what features need to be designed accordingly, and how one can generalize from different syntactic structures. Previous systems discovered new syntactic structures through data and error analyses and relied on the intuition of the English language for context dependencies among arguments. Drawing upon the theory of argument realization, the current project splits the feature design for core and non-core arguments, defines a base argument configuration corresponding with the argument positions of the core arguments in an argument structure of the verb, identifies the syntactic structures whose core arguments' positions vary from the base argument configuration, reconstructs the base argument configurations for these variations to generalize across the structures of the same verb, and integrates the features into a machine learning framework.

The next chapter discusses the feature design that addresses the issues raised in the above observations.

Chapter 4
Feature Design

As discussed in chapter 3, although existing SRL systems differ in many ways, two tasks are common to all SRL systems. One of the tasks is to conceive a set of features to represent the training and test data in hand. The other is to select an appropriate machine learning model to integrate these features. In terms of feature engineering, the designs have evolved in three directions in the past eight years. First, in the years elapsed between 2000 and 2007, with the feature representation in the framework of constituency grammar, researchers focused on exploring how context dependencies among arguments could be utilized to help the SRL task (section 3.2). Second, in the same framework, SRL researchers headed down the direction of integrating structure-specific features as opposed to the structure-general features (section 3.2). Along the third direction, the community has witnessed the shift from the use of constituency grammar for feature representation to that of dependency grammar representation around 2006/2007 (section 3.3).

The work in all three directions was fruitful and yet can be further explored. The initial interests in trying to replace the constituency grammar representation with the dependency grammar representation culminated in the work of CoNLL-2008 systems. However, since the researchers could not conclude from the CoNLL-2008 results that the dependency grammar representation provides for a *better* solution (Surdeanu et al., 2008a), they are determined to further explore this direction on CoNLL-2009 (Hajič, 2009).

The work in the direction of utilizing context dependencies among arguments showed that such dependencies were helpful but did not distinguish between core and non-core arguments. While at the same time the **theory of argument realization** states that there exist context dependencies between the semantic role of a core argument and those of other core arguments, it does not mention any context dependency between a non-core semantic role and other core or non-core semantic roles. Similarly, the lexical resources, whether it be PropBank or FrameNet that all existing SRL systems rely on for training, define core semantic roles to be specific to the core arguments of a verb, while stating that the non-core semantic roles are not specific to any verb but general to all verbs. Separating the non-core arguments from the context dependencies among arguments seems to be necessary.

In the direction of choosing between more general or more specific features, in order to handle the syntactic structures that generic features failed to represent, instead of finding the common features shared by different syntactic structures, SRL researchers designed features specific to these structures which not only led to larger feature spaces but also did not necessarily generalize across different syntactic structures that the verb appears in.

The feature design of the current project investigates the areas that have not been much explored in these three directions. The three main areas the current project investigates are summarized as follows.

First, the current project employs the *base argument configuration* feature(s) of a given verb to generalize across twelve different syntactic structures involving moved or displaced arguments that the verb may appear in. The argument positions in a base argument configuration of a given verb correspond to those in one of the verb's argument structures. In the syntactic structures where arguments of a given verb are moved or displaced arguments, the argument positions are different from those in any of the verb's base argument configurations. In these cases, relying on the knowledge about these structures and about the arguments' originating positions, procedures are applied to create the base argument configuration features for these arguments. When the BAC features are successfully created for the moved or displaced arguments, they generalize across these syntactic structures. As a matter fact, the current project observes twelve types of syntactic structures involving moved and displaced arguments from the Penn Treebank data. The knowledge about theses structures and about the originating positions of the moved or displaced arguments are encoded in the system for the purposes of either identifying a verb's arguments or creating the BAC features for the argument classification task.

Second, through base argument configuration features, the current project imposes the context dependency constraint explicitly on the core arguments of a verb but not on the non-core arguments. A base argument configuration feature lists the full set of core arguments of a given verb. The presence of all core arguments helps to enforce the context dependence constraints which states that the semantic roles of core arguments are dependent on each other.

Finally, when deriving base argument configuration features, the current project utilizes both dependency grammar and constituency grammar, instead of siding with one of the two grammars. The procedures to identify the twelve types of syntactic structures involving moved or displaced arguments as well as the operations to reconstruct base argument configurations regarding these syntactic structures necessarily cross the boundaries between constituency grammar representation and dependency grammar. Although the BAC features for the core arguments do not directly include grammatical relations defined in dependency grammar, the procedures to derive the base argument configuration features do rely on grammatical relations such as subject-verb, verb-object, and modifier-head relations that constituency grammar-based SRL systems do not use. At the same time, syntactic categories, i.e. phrase types of the arguments, that do not appear in dependency grammar-based SRL systems, are listed in BAC features. Therefore, the feature design for the core arguments in the current project does not take sides in the debate on the effectiveness of the two grammatical formalisms, but instead draws upon both formalisms simultaneously.

The preceding passages outline the three areas in feature engineering in which the current project differs from existing systems. At the same time, the current project does recognize that although the BAC features generalize across different syntactic structures that a verb appears in they do not generalize across different verbs because the verb is built in base argument configurations and the core arguments are constrained on the verb itself. To reinforce the base argument configuration features, the current project also utilizes two levels of **backoff features**. The Level-I features are intended to handle unrealized arguments. The Level-II backoff features correspond with the commonly-used contextual features in literature that do not have to be dependent on the predicate verb.

For the non-core arguments, the current project employs the Level-II features.

The rest of the chapter proceeds as follows. Section 4.1 previews the architecture of the current SRL system through the semantic role labeling/testing phase, aiming to illustrate how the feature extraction for the core and non-core arguments defined in the current chapter fits into the complete system. Section 4.2 describes base argument configurations and the feature design for the core-arguments. Section 4.3 describes the twelve syntactic structures whose core arguments have either moved or displaced and discusses the approaches to constructing the base argument configuration features in each case. Section 4.4 describes the Level-I back-off features to handle unrealized arguments and the Level-II back-off features to handle the unseen verbs in the training data. Section 4.4 also points out that the Level-II features are general enough for any realized core arguments. Section 4.5 analyzes the non-core arguments and presents the corresponding features as a subset of the Level-II features.

4.1 System preview

Following the common approach in SRL, the current system starts with the argument identification module to identify the core and non-core semantic arguments for each verb in a given input sentence. The argument identification module selects a list of candidate arguments for each verb from all the constituents in the sentence. Then it runs a three-way classifier to classify each candidate argument as core, non-core, or non-argument. When the arguments are identified, the system runs the argument classification module to classify each argument and assigns it a semantic label. For both training and testing, the system uses the PropBank data (section 2.2) formatted for the CoNLL-2005 shared task of SRL.

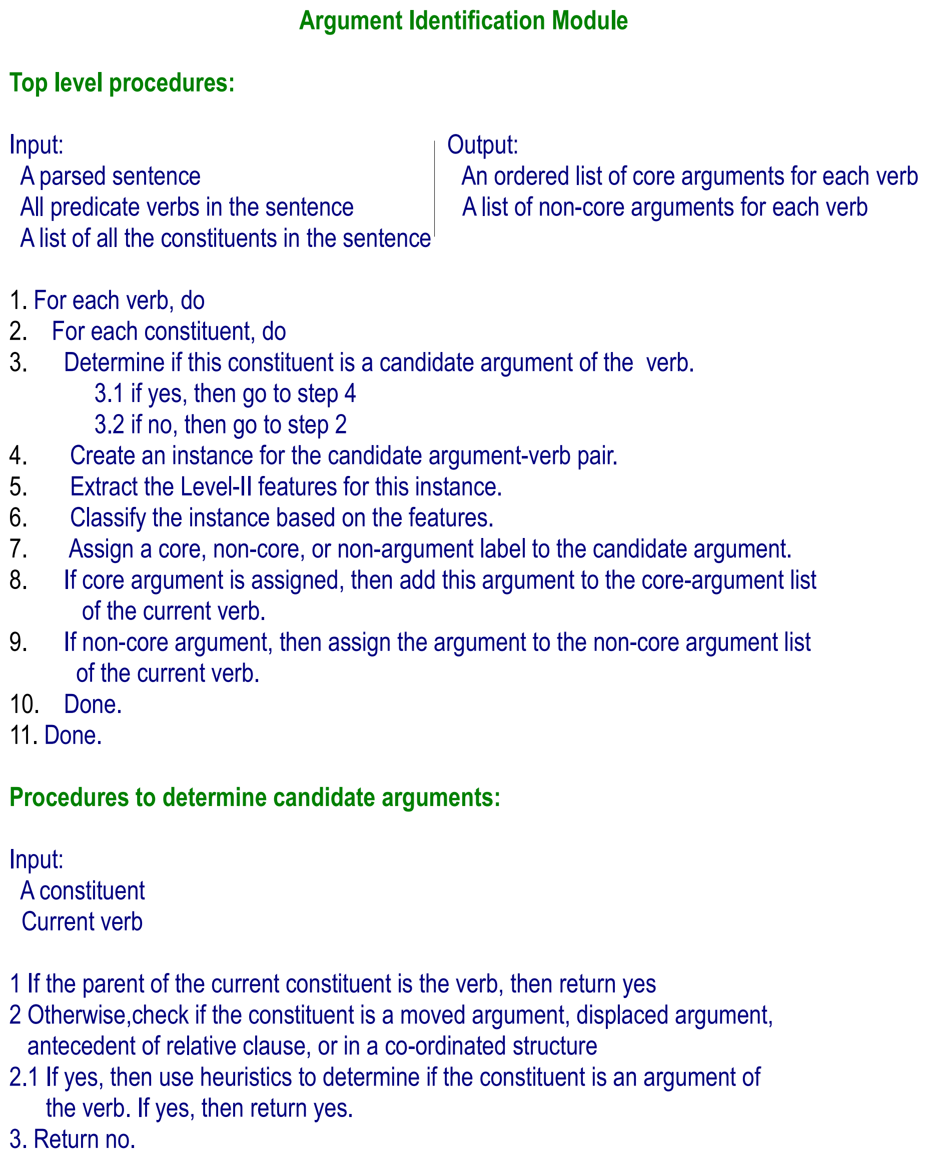
Table 4.1 shows the data format from CoNLL-2005. Column 1 lists the words in the input sentence. Column 2 gives the predicates' lemmas. Column 3 corresponds to the parse tree. The parent's index of each constituent is shown in column 4, where -1 indicates the root of the parse tree. Column 5 lists the semantic arguments and their role label for the first predicate *sow*. Column 6 corresponds with the semantic arguments and their role label for the second predicate *pollinate*. During training, the system is allowed to access the full table. During testing, the system does not have access to the content in columns 5 and 6. Instead, the system is supposed to identify the semantic arguments and predict their semantic role labels in columns 5 and 6.

0	1	2	3	4	5	6
idx.	**word**	**predicate**	**parse**	**parent idx.**	**sow** cand.	**pollinate** cand.
1	They	-	(S(NP-SBJ*)	2	(A0*)	*
2	**sow**	**sow**	(VP*	-1	(V*)	*
3	a	-	(NP(NP(NP*	-	(A1*	*
4	row	-	*)	2	*	*
5	of	-	(PP*	-	*	*
6	male-fertile	-	(NP*	-	*	(A0*
7	plants	-	*))))	5	*)	*)
8	nearby	-	(ADVP-LOC*)	2	(AM-LOC*)	*
9	,	-	*	2	*	*
10	which	-	(SBAR-1(WHNP-2*)	2	(C-A1*	(R-A0*)
11	then	-	(S(ADVP-TMP*)	12	*	(Non-core*)
12	**pollinate**	**pollinate**	(VP*	10	*	(V*)
13	the	-	(NP*	-	*	(A1*
14	male-sterile	-	*	-	*	*
15	plants	-	*)))))	12	*)	*)
16	.	-	*)	-	*	*

Table 4.1: Sample data format from CoNLL-2005

The rest of the section provides a brief summary of the training phase and a preview of the current novel system architecture through the semantic role labeling/testing phase, highlighting the feature extraction procedures for the candidate arguments during argument identification and for the core and non-core arguments/instances during semantic role labeling. The formal presentation of the system architecture can be found in chapter 5.

The training phase trains three classifiers, the three-way argument identifier, an argument classifier based on the BAC and Level-I features of the annotated instances in the training data, and a second argument classifier based on Level-II features only. The instances for the argument identifier and that for the argument classifier will be separately introduced in the rest of this section.



Argument Identification Module

Top level procedures:

Input:	Output:
A parsed sentence	An ordered list of core arguments for each verb
All predicate verbs in the sentence	A list of non-core arguments for each verb
A list of all the constituents in the sentence	

```
1. For each verb, do
2.    For each constituent, do
3.       Determine if this constituent is a candidate argument of the  verb.
             3.1 if yes, then go to step 4
             3.2 if no, then go to step 2
4.        Create an instance for the candidate argument-verb pair.
5.        Extract the Level-II features for this instance.
6.        Classify the instance based on the features.
7.        Assign a core, non-core, or non-argument label to the candidate argument.
8.       If core argument is assigned, then add this argument to the core-argument list
            of the current verb.
9.       If non-core argument, then assign the argument to the non-core argument list
            of the current verb.
10.   Done.
11. Done.
```

Procedures to determine candidate arguments:

Input:
A constituent
Current verb

```
1 If the parent of the current constituent is the verb, then return yes
2 Otherwise,check if the constituent is a moved argument, displaced argument,
  antecedent of relative clause, or in a co-ordinated structure
2.1 If yes, then use heuristics to determine if the constituent is an argument of
    the verb. If yes, then return yes.
3. Return no.
```

Figure 4.1: The Argument Identification module

4.1.1 Argument identification

During the testing/semantic role labeling phase, the complete system performs semantic role labeling through three steps, including parsing, argument identification (see section 5.2), and argument classification (see section 5.3).

Given an unannotated sentence as the input, the system starts with parsing the sentence with the Charniak parser.

Then, the system hands all the constituents in the parsed sentence down to the argument identifier. The argument identifier (shown in figure 4.1) proceeds with identifying the core and non-core semantic arguments for each predicate verb in the sentence from the constituents in three sub-steps.

First, the identifier selects the constituents whose parent is a predicate, including the syntactic arguments, adjuncts, modal verbs, discourse markers, and negation adverbs, as the candidate semantic arguments for the predicate. However, there are four situations where the predicate is not the parent of its argument on the parse tree. In these cases the identifier has to rely on heuristics to locate these arguments. These situations include the syntactic structures involving moved arguments, displaced arguments, shared arguments of the verbs in co-ordinated structures, and the extra arguments as the antecedents in relative clauses.(see section 4.3 for examples).

Next, the identifier extracts for each candidate/instance a set of generic features commonly used in literature for argument identification. This set of features are the Level-II features defined in section 4.4.3. In other words, this step prepares the instances and their features for the next step. Here, an instance is a candidate argument and verb pair. And the corresponding feature vector contains the Level-II features.

Finally, the identifier classifies the candidate semantic arguments as core, non-core, or non arguments, using the three-way classifier. The core arguments correspond to the syntactic arguments as discussed in section 2.1.2. And the non-core arguments correspond to the adjuncts, modal verbs, discourse markers, and negation adverbs (see section 2.1.2).

At this point, the argument identifier stops and outputs a list of core arguments and a list of non-core arguments for each verb in the input sentence. The procedures of the argument identifier are illustrated below. Figure 4.1 displays the main procedures of the argument identifier.

4.1.2 Identification examples

Table 4.2 shows an input sentence to the argument identifier. There are two predicates in the input sentence, *sow* and *pollinate*. The POS tags are shown in the third column. Column four shows the parse tree. Column five lists the parent head indices derived from the parse tree for the constituents. The negative index -1 indicates the root of the parse tree.

When receiving the parsed sentence shown in table 4.2, the argument identifier selects the constituents shown in table 4.3 for each of the two verbs. The candidate arguments of *sow* are listed in column 5. Those of *pollinate* are in column 6.

Next, the identifier selects the candidate arguments for each verb, which are shown in table 4.3. The identifier easily reads the candidate arguments off the parse tree for verb *sow* because their parent head indices all point to the index of *sow*, which is 2. Now, *sow* is

idx	word	POS	parse	parent idx
1	They	PRP	(S(NP-SBJ*)	2
2	**sow**	VBP	(VP*	-1
3	a	DT	(NP(NP(NP*	-
4	row	NN	*)	2
5	of	IN	(PP*	-
6	male-fertile	JJ	(NP*	-
7	plants	NNS	*))))	5
8	nearby	RB	(ADVP-LOC*)	2
9	,	,	*	2
10	which	WDT	(SBAR-1(WHNP-2*)	2
11	then	RB	(S(ADVP-TMP*)	12
12	**pollinate**	VBP	(VP*	10
13	the	DT	(NP*	-
14	male-sterile	JJ	*	-
15	plants	NNS	*)))))	12
16	.	.	*)	-

Table 4.2: Parsed input sentence and derived head indices

idx.	word	parse	head idx.	**sow** cand.	**pollinate** cand.
1	They	(S(NP-SBJ*)	2	(Arg*)	*
2	**sow**	(VP*	-1	(V*)	*
3	a	(NP(NP(NP*	-	(Arg*	*
4	row	*)	2	*	*
5	of	(PP*	-	*	*
6	male-fertile	(NP*	-	*	(Arg*
7	plants	*))))	5	*)	*)
8	nearby	(ADVP-LOC*)	2	(Arg*)	*
9	,	*	2	*	*
10	which	(SBAR-1(WHNP-2*)	2	(Arg*	(Arg*)
11	then	(S(ADVP-TMP*)	12	*	(Arg*)
12	**pollinate**	(VP*	10	*	(V*)
13	the	(NP*	-	*	(Arg*
14	male-sterile	*	-	*	*
15	plants	*)))))	12	*)	*)
16	.	*)	-	*	*

Table 4.3: Candidate arguments for *sow* and *pollinate*

associated with the list of candidate arguments, (*they*, *a row ... plants*, *nearby*, *which then ... plants*). However, while the candidate arguments, (*which*, *then*, *the male-sterile plants*) for *pollinate* can be easily collected from the parse tree because their parent head indices point to *pollinate*, the candidate argument *male-fertile plants*, spanning words 6 and 7, can only be selected with heuristics. In this case, the heuristics says that

> the antecedent of a relative clause needs to be collect as a candidate argument of the predicate in the relative clause.

Once the argument identifier chooses the candidate arguments for *sow* and *pollinate*, it creates instance for each candidate-verb pair, extracts the Level-II features for each instance, and classifies the candidate arguments as core, non-core, or non arguments. For example, for the candidate argument *a row of male-fertile plants*, the argument identifier creates an instance *a row ... plants: SOW*, extracts the features {head_plants, predicate_SOW, etc. }, and classifies it as a core argument.

The resulting core and non-core arguments for *sow* are shown in column 5 of table 4.4, and those of *pollinate* are shown in column 6.

idx.	**word**	**parse**	**head idx.**	**sow** cand.	**pollinate** cand.
1	They	(S(NP-SBJ*)	2	(Core*)	*
2	**sow**	(VP*	-1	(V*)	*
3	a	(NP(NP(NP*	-	(Core*	*
4	row	*)	2	*	*
5	of	(PP*	-	*	*
6	male-fertile	(NP*	-	*	(Core*
7	plants	*))))	5	*)	*)
8	nearby	(ADVP-LOC*)	2	(Non-core*)	*
9	,	*	2	*	*
10	which	(SBAR-1(WHNP-2*)	2	(Core*	(Core*)
11	then	(S(ADVP-TMP*)	12	*	(Non-core*)
12	**pollinate**	(VP*	10	*	(V*)
13	the	(NP*	-	*	(Core*
14	male-sterile	*	-	*	*
15	plants	*)))))	12	*)	*)
16	.	*)	-	*	*

Table 4.4: Core and non-core arguments for *sow* and *pollinate*

4.1.3 Argument classification

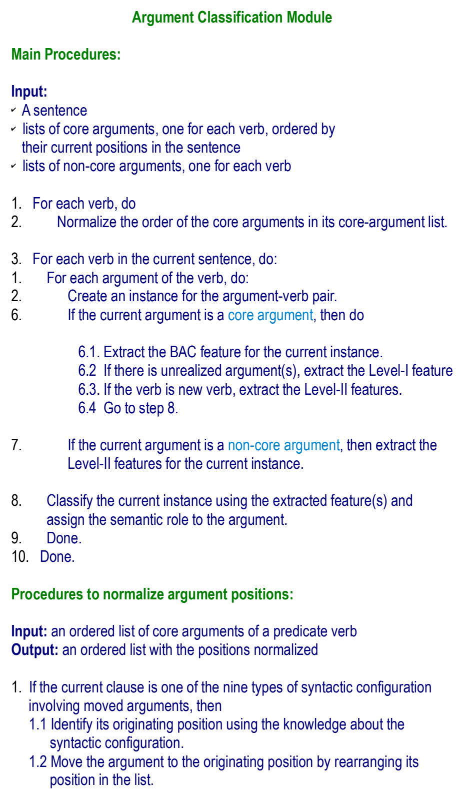
The argument identifier (see figure 4.2) assigns a list of core arguments and a list of non-core arguments for each predicate verb in an input sentence. Then, the argument classifier proceeds as follows.

For each of the non-core arguments of a verb, the argument classifier creates a non-core argument-verb instance and extracts the Level-II features for the instance.

Before the argument classifier creates instances and extracts features for the core arguments, it normalizes the positions of the arguments in the core-argument list for each verb as follows.

Given the ordered list of core arguments of a verb, the argument classifier checks if any of the arguments is a moved argument, a displaced argument, a shared argument in a co-ordinated structure, or the referent of a relative clause (see section 4.3). If the argument is involved in one of these structures, the argument classifier reconstructs its originating position based on the coded knowledge of these structures and inserts the argument to the relevant position in the core argument list. If none of the arguments is involved in any of the above four situations, the core argument list is kept as is.

Once the core argument list of the verb is normalized. The argument classifier creates a core argument-verb instance for each core argument and extracts the base argument configuration feature for the instance. Recall that a base argument configuration (BAC) represents an argument structure of the verb. When the arguments are normalized, the cur-



```
Argument Classification Module

Main Procedures:

Input:
✓ A sentence
✓ lists of core arguments, one for each verb, ordered by
  their current positions in the sentence
✓ lists of non-core arguments, one for each verb

1.  For each verb, do
2.      Normalize the order of the core arguments in its core-argument list.

3.  For each verb in the current sentence, do:
1.     For each argument of the verb, do:
2.          Create an instance for the argument-verb pair.
6.          If the current argument is a core argument, then do

               6.1. Extract the BAC feature for the current instance.
               6.2  If there is unrealized argument(s), extract the Level-I feature
               6.3. If the verb is new verb, extract the Level-II features.
               6.4  Go to step 8.

7.          If the current argument is a non-core argument, then extract the
            Level-II features for the current instance.

8.     Classify the current instance using the extracted feature(s) and
       assign the semantic role to the argument.
9.     Done.
10.  Done.

Procedures to normalize argument positions:

Input: an ordered list of core arguments of a predicate verb
Output: an ordered list with the positions normalized

1.  If the current clause is one of the nine types of syntactic configuration
    involving moved arguments, then
    1.1 Identify its originating position using the knowledge about the
        syntactic configuration.
    1.2 Move the argument to the originating position by rearranging its
        position in the list.
```

Figure 4.2: The Argument Classification module

rent project considers the resulting argument list and the verb's position correspond to one of the argument structures of the verb. Hence, the resulting feature that lists the arguments and the verb positions is called the base argument configuration feature. The next section gives some examples of BAC features. While the BAC feature represents an argument structure of a verb, it also represents the context dependence among the semantic roles of the arguments in the list.

After the argument classifier extracts the BAC feature for an instance, it also extracts the Level-I feature for the instance. If the verb has not been seen before in training, Level-II features will be extracted for the instance.

When the argument classifier finishes extracting features for all the instances, it classifies the instances and assigns each a semantic role label.

The above procedures followed by the argument classifier is summarized in figure 4.2.

4.1.4 Classification examples

The preceding section depicts the procedures of the argument classifier. This section illustrates the procedures with a sentence without any moved or displaced core argument and with a passive sentence with a moved argument.

(55) is a sentence where no moved or displaced argument occurs. (55).1 shows the core argument list that has been identified by the argument identifier. When the argument classifier gets the core argument list, it checks if any of the argument has been moved or displaced. If this case, none has. So, the argument classifier keeps the order of the arguments as is. Next, it creates an instance for each argument, as shown in (55).2 and (55).3. Since there is no moved or displaced argument, when the argument classifier extracts the BAC feature for *Dana*, it reads the phrase type of the argument list, marks the position of the argument in question, and inserts the verb in the correct position. So, *Data* gets the BAC feature of {**curNP**-open-NP}.

(55) [$_{Core}$Dana] opened [$_{Core}$ the door].

(1) **Core argument list**: {NP_Dana, NP_the door}

(2) **Instance and BAC feature:** *Dana_open* - {**curNP**-open-NP}

(3) **Instance and BAC feature**: *the door_open* {NP-open-**curNP**}

On the other hand, (56) is a passive sentence. According the Treebank annotation, the subject is a moved argument. And the final NP is in fact a displaced core argument. Both arguments are correctly identified as core arguments by the argument identifier. Then, in (56).4, the argument classifier normalizes the positions of the core arguments by reconstructing their positions according to base argument configuration of the verb. The argument classifier encodes the knowledge about the passive structures so that is able to reconstruct the moved and displaced arguments. Then in (56).5 and in (56).6 the BAC features are extract for each argument.

(56) [$_{NP\_thing-discovered}$ The problems$_i$] were uncovered *trace$_i$* ... by [$_{NP\_discoverer}$ the government].

(1) **Core argument list:** { NP_ the problems, NP_the government}

(2) **Moved argument:** { NP_ the problems }

(3) **Displaced argument:** { NP_the government}

(4) **Normalized core argument list:** { NP_the government, NP_ the problems}

(5) **Instance and BAC feature:** { *The government_uncover* - curNP_uncover_NP}

(6) **Instance and BAC feature:** { *the problems_uncover* - NP_uncover_curNP}

4.2 Base argument configuration and BAC features

As section 2.4.1 illustrates, an argument structure of a given verb specifies the positions of its subject, object, and other types of complements of with respect to the verb. A predicate verb may appear in one or more argument structures while there is no argument in each argument structure has moved or been displaced. Then, section 2.4.2 presents the definition of **base argument configuration**. A base argument configuration with respect to a given verb consists of the arguments in an argument structure of the same verb except that the argument types of subject, object, etc are replaced with their corresponding syntactic categories and the lemma of the verb is also included. A base argument configuration feature then is just a listing of the elements in a base argument configuration with its current position marked by the string *cur*.

Extracting the BAC features for the core arguments of a predicate verb that is not in any of the structures involving moved or displaced arguments is straight forward. That is, to extract the BAC feature for a core argument of this verb, one needs only to list the syntactic categories of the arguments and the lemma of the verb in the order as they appear in the clause the verb appears in because these core arguments are assumed to overlap with the arguments in an argument structure of the verb. In addition to the examples in sections 2.4 and 4.1, example (57) helps to illustrate this point. Because there is no moved argument in (57).1, the current base argument configuration with respect to verb *finger*, shown in (57).3, is considered as a sequential listing of the syntactic categories of the two arguments, shown in (57).2, from the parse tree with the lemma of the verb indicated. Then, the BAC features for the two arguments can be easily extracted as shown in (57).4 and (57).5.

(57) (1) To date, $[_{NP}$ scientists] have fingered $[_{NP}$ two of these cancer-suppressors].

(2) **Core argument list:** (subject_NP, object_NP)

(3) **Base argument configuration** (NP, finger, NP)

(4) **BAC feature for *scientists*** (curNP-finger-NP)

(5) **BAC feature for *two...cancer-suppressors*** (NP-finger-curNP)

However, if any of the arguments of the verb has moved or been displaced, the resulting arguments' positions do not match with those in any argument structure of the verb. There will be no BAC features for these arguments unless special knowledge and operations are needed to reconstruct a base argument configuration for the verb. The next section discusses the knowledge and operations needed to extract the BAC features for the arguments of the verbs involved in twelve types of structures with moved or displaced arguments.

4.3 Handling configurations different from BAC

Sections 2.4 and 4.1 show that extracting the BAC features for the arguments of the predicates involved in the structures where moved or displaced arguments occur requires reconstructing a base argument configuration for the predicate. This section first describes the twelve types of structures observed in the Penn Treebank where moved or displaced arguments occur. These structures are grouped into four categories, including nine types of moved arguments, one type of displaced arguments, one type of shared arguments in coordinated structures, and one type of extra arguments as the antecedents of relative clauses. At the same time, the procedures to extract the BAC features for these arguments are illustrated.

Please note that the twelve types of syntactic structures involving moved or displaced arguments are observed in the Peen Treebank with noticeably high frequencies. However, there potentially exist other types syntactic structures involving moved or displaced arguments that have not been included here because that they either have not been observed or do not occur in Treebank at all.

4.3.1 Structures involving movements

This section starts with illustrating the nine types of moved arguments. Then, the procedures to create the BAC feature for these arguments are described.

4.3.1.1 Structures involving movements

Nine types of syntactic configurations involving movements of the argument(s) that are found in the training and development data sets, including Treebank sections 2 through 21 and 24 are handled in current project. Table 4.5 lists such configurations. Following the Treebank notation, the moved arguments and their trace are co-indexed in examples (58) through (67). Note that the actually Treebank tag for traces is *-NONE-* and *trace* is used in the current presentation only for the sake of illustration.

Syntactic structure	Example
subject of passive verb	subject *the problems* of *uncover* in 58
subject of relative clause	subject *which* of *widen* in 59
object of relative clause	object of *which* of *define* in 60
subject control	subject *John* of *pull* through subject control verb *try* in 61
object control	subject *the commission* of *audit* through object control verb *order* in 62
subject of raising verb	subject of the raising verb *seem* also the subject of the verb *share* in 63
subject of V-ing clause	subject *Yasser Arafat* of *ask* in 64
subject of what-clause	subject *what* of *amount* in (66)
object of what-clause	object *what* of *hope* in (67)

Table 4.5: Structures involving movements and examples

(58) $[_{NP\_thing-discovered}$ The problems$_i$] were uncovered *trace$_i$* ... by $[_{NP\_discoverer}$ the government].

(59) ... $[_{NP\_casualagent}$ the U.S. trade deficit$_i$] , $[_{WHNP\_casualagent}$ which$_i$] *trace$_i$* widened $[_{PP\_extent}$ by 31%]...

(60) ...$[_{NP\_thing-defined}$ its cash flow$_i$], $[_{WHNP\_thing-defined}$ which $_i$] $[_{NP\_describer}$ the company] defined *trace$_i$* $[_{PP\_secondaryattribute}$ as earnings] ...

(61) $[_{NP\_puller}$ John$_i$] tried *trace$_i$* to pull $[_{NP\_thingpulled}$ the wool] over Mary's eyes...

(62) The Illinois Supreme Court ordered $[_{NP\_auditor}$ the commission$_i$] *trace$_i$* to audit $[_{NP\_audited}$ Commonwealth Edison's construction expenses] ...

(63) ...$[_{NP_sharer}$nations$_i$] seem *trace$_i$* to share $[_{NP_thing_shared}$his views of America]...

(64) $[_{NP\_asker}$ Yasser Arafat$_i$] has written to the chairman..., *trace$_i$* asking $[_{NP\_hearer}$ him] $[_{VP\_favor}$ to back a Palestinian bid to join]...

(65) $[_{NP\_institution}$ Bell$_i$], based *trace$_i$* in Los Angeles, makes and distributes ...

(66) ... program traders received $[_{WHNP\_focus}$ what$_i$ *trace$_i$* amounted $[_{PP\_ground}$ to an exemption from the uptick rule in certain situations]...

(67) Sales were well short of $[_{WHNP\_thinghopedfor}$ what$_i$ $[_{NP\_hoper}$ they] had hoped *trace$_i$*.

4.3.1.2 Reconstructing BAC features for moved arguments

To construct the base argument configuration features for each type of configuration in the preceding section, the project relies on two kinds of knowledge. First, the feature extraction module of the current project encodes the knowledge about the argument positions of each syntactic structure and how the positions are related to those in the corresponding positions in an argument structure of the verb, or an base argument configuration of the verb. Table 4.6 lists the knowledge encoded for the nine types of structures involving movements.

Syntactic structure	Encoded grammatical knowledge
subject of passive verb	position of the subject in passive voice
subject of relative clause	position of the subject in relative clause
object of relative clause	position of the object in relative clause
subject control	subject control verbs learned from training data position of subject of controlled predicate
object control	subject control verbs learned from training data position of subject of controlled predicate
subject of raising verb	raising verbs learned from training data position of the subject of the infinitive
subject of present participle clause	head noun modified by the V-ing clause
subject of what-clause	position of subject in what-clause
object of what-clause	position of object in what-clause

Table 4.6: Encoded knowledge about each structure

Second, the feature extractor also relies on the grammatical relations defined in dependency grammar to reconstruct the base argument configurations because the relations such as subject-verb, verb-object, and modifier-head are essential to identify the nine types of syntactic structures involving moved arguments as shown in table 4.5. Although the Penn Treebank provides for some grammatical relations such as the subject relation in the annotations, they cannot be used because SRL systems cannot assume gold parses are always available and automatic constituency grammar parses, such as Charniank and Collins parsers. The current project instead derives the needed grammatical relations, such as subject-verb, verb-object, and modifier-head relations, using the word-to-head relations read from the parse tree. The word-to-head relations are provided for by the software in Yamada (2008) . Table 4.7 lists the parse of the first half of a sentence along with the head-word index of each word.

The columns in table 4.7 correspond to the index, word form, POS, current phrase type, and head word index of each word respectively. The head word indices show the dependencies between a word and its head word. The modal verb *would* is the highest node in the dependencies and does not have a head in the tree. The head index for the

Index	Word Form	POS	Phrase Type	Head Index
1	The	DT	(NP	3
2	proposed	VBN	-	3
3	changes	NNS	)	5
4	also	RB	(ADVP)	5
5	would	MD	(VP	-1
6	allow	VB	(VP	5
7	executives	NNS	(NP)	9
8	to	TO	(VP	9
...				

Table 4.7: Sample parse from the Charniak parser

highest node is -1. Based on the word-to-head information, the feature extractor extracts the modifier-head relation between *proposed* and its head *changes*. Similarly, the NP *the proposed changes* is extracted as the subject of *allow* because the head index of the NP is *would*.

With the above two kinds of knowledge, the current systems extracts the BAC features for each moved argument in examples (58) through (67) in table 4.8.

In	Argument	Surface structure	BAC feature
(58) uncover	*the problems*	curNP uncover *trace* NP	NP uncover curNP
(59) widen	*which*	NP curWHNP *trace* widen PP	curWHNP widen PP
(60) define	*which*	NP curWHNP NP define *trace* PP	NP define curWHNP PP
(61) pull	*John*	curNP try *trace* to pull NP	curNP pull NP
(62) audit	*the commission*	curNP *trace* audit NP	curNP audit NP
(63) share	*nations*	curNP seem *trace* share NP	curNP share NP
(64) ask	*Yasser*	curNP write *trace* ask NP toVP	curNP ask NP toVP
(65) base	*Bell*	curNP, base *trace*	base curNP
(66) amount	*what*	curWHNP *trace* amount PP	curWHNP amount PP
(67) hope	*what*	curWHNP NP hope *trace*	NP hope curWHNP

Table 4.8: BAC features of moved arguments in examples (58) through (67)

Please note that rather than being used in the procedures to reconstruct base argument configurations, the grammatical relations themselves are not directly built into the features in the current project. Therefore, none of the features in table 4.8 includes grammatical relations as part of the feature. Also note that the first NP arguments of verbs *widen* and *define* in their surface argument lists shown in table 4.8 are both dropped from the corresponding features }*curWHNP widen PP* }and { *NP define curWHNP PP* }. Section 4.3.4 discusses how extra arguments are handled in some relative clauses which include the preceding two cases.

4.3.2 Structures involving displaced arguments and BAC feature

The preceding section illustrates the syntactic configurations where arguments have undergone movement so that they do not correspond with the argument positions in any argument structure of the verb. This section describes one syntactic structure where the NP subject argument is placed after the predicate verb which grammatically functions as a V-ing modifier of the NP. Table 4.9 lists the structure and the relevant example shown in (68).

Syntactic structure	Example
V-ing modifying a noun	*members* is the subject of the V-ing modifier *remaining* in (68)

Table 4.9: Structures involving displaced arguments and FAC features

(68) ... the remaining [$_{NP_thing_remaining}$members]...
- **surface structure of *members***: *remain curNP*
- **BAC feature for *members***: *curNP remain*

In example (68), the subject NP *members* of the V-ing predicate *remaining*, which is actually a modifier of the subject NP, cannot be found at the position preceding the predicate at the position of its subject.

To derive the BAC feature in the V-ing modified NP structure, one can just move the argument in front of the verb. So, after moving the phrase of NP of*members* in front of the modifying verb *remain*, one gets the BAC feature *NP remain* as shown in (68).

4.3.3 Structures involving shared arguments and BAC features

Predicate verbs connected by a conjunction, such as *and* or *or*, share a grammatical subject or a grammatical object, or both. Table 4.10 lists the *coordinated verb structure* that shares the arguments shown in (69) and (70).

Syntactic structure	Example
coordinated verb	*Bell* shown as the subject of *makes* in (69) *building products* shown as the object of *makes* in (69) *Bell* shown as the subject of *distributes* in (70) *building products* shown as the object of *distributes* in (70)

Table 4.10: Structures involving shared arguments

Although the verbs *make* and *distribute* share the same grammatical subject *Bell* and object *building products*, (69) and (70) show that the semantic role sets are { *creator, created* } and { *distributor, thing distributed* } respectively.

(69) [$_{S}$ [$_{NP\_creator}$ Bell] [$_{VP}$ [$_{V}$ makes] [$_{CC}$ and] [$_{V}$*distributes*] [$_{NP\_created}$ building products]]].
- **surface structure of *Bell***: *curNP make NP*
- **BAC feature for *building products***: *NP make curNP*

(70) [$_{S}$ [$_{NP\_distributor}$ Bell] [$_{VP}$ [$_{V}$ makes] [$_{CC}$ and] [$_{V}$*distributes*] [$_{NP_thing_distributed}$building products]]].
- **surface structure of *Bell***: *curNP distribute NP*
- **BAC feature for *building products***: *NP distribute curNP*

To convert the surface structure of the arguments of the verbs in the conjunction structure, one can follow the steps as follows:
1. Identify the core arguments of each verb accordingly.
2. Check if any argument of a verb has undergone movement. If so, use the approach from section 4.3.1.2 to construct the BAC features for the moved argument(s). If not, construct the BAC feature for each core argument based on their current positions.

Since none of the arguments in (69) and (70) has moved, the BAC features for each core argument are create based on their current positions as shown in (69) and (70) .

4.3.4 Structures involving extra arguments and BAC features

In relative clauses where the relative pronoun is the subject of the predicate verb in the relative clause, the predicate takes both the antecedent/referent of the relative clause and the relative pronoun as its arguments which share an identical semantic role. Thus, the predicate has an extra argument. For example, in (71) the predicate verb *widen* in the relative clause has the relative pronoun *which* and the referent of the relative clause *the U.S. trade deficit* as its arguments that take the same semantic role of *casual_agent*. The relative pronoun *which* is the grammatical subject of *widen*. Similarly, the verb *define* in the relative clause in (72) has the relative pronoun *which* and the referent *the cash flow* as its arguments, both of which have the same semantic role of *thing_defined*. The relative pronoun *which* is the grammatical subject of *define*. However, neither *widen* nor *define* takes two arguments of the same role in their argument structures.

To convert the surface argument structure of *which* to a BAC feature, the feature extractor simply drops the phrase type of *the U.S. trade deficit*. If one applies the same approach the other three forementioned arguments, one gets the BAC features listed in (71) and (72).

(71) ... [$_{NP\_casualagent}$ the U.S. trade deficit$_i$] , [$_{WHNP\_casualagent}$ which$_i$] *trace$_i$* widened [$_{PP\_extent}$ by 31%]...
- **surface structure of *the U.S. trade deficit***: *curNP WHNP *trace* widen PP*
- **BAC feature for *the U.S. trade deficit***: *curNP widen PP*

- **surface argument structure of *which***: *NP curWHNP *trace* widen PP*
- **BAC feature for *which***: *curWHNP widen PP*

(72) ...$[_{NP\_thing-defined}$ its cash flow$_i$], $[_{WHNP\_thing-defined}$ which $_i]$ $[_{NP\_describer}$ the company] defined *trace$_i$* $[_{PP\_secondaryattribute}$ as earnings] ...
- **surface structure of *its cash flow***: *curNP WHNP NP define *trace* PP*
- **BAC feature for *its cash flow***: *NP define curWHNP PP*
- **surface structure of *which***: *NP curWHNP NP define *trace* PP*
- **BAC feature for *which***: *NP define curNP PP*

4.4 Back-off features for core arguments in two situations

The previous two sections discuss the BAC feature design that generalizes from different syntactic structures of a verb. While the BAC features generalize across different syntactic configurations of the verb, they are also specific to the verb because the verb form is built into the BAC features. However, there are two situations that call for features more general than the BAC features because no base argument configuration can be reconstructed for the predicate in these situations. The first situation includes the cases where a core argument of the verb is not realized. In the second situation, the predicate verb itself has not been seen by the statistical model during training. This rest of the section starts with describing the syntactic structures where some core arguments are not realized. Then the Level-I and Level-II back-off features are described in turn.

4.4.1 Unrealized core arguments

Unrealized core arguments are the arguments that appear in the base argument configuration but are missing in the surface syntactic structure of a given verb. Table 4.11 lists the syntactic structures where some core arguments may not be realized. Since the base argument configuration feature requires all core arguments be present, when there is unrealized argument the BAC feature cannot be extracted for the realized argument. Examples (73) through (79) illustrate this case.

Syntactic structure	Example
passive voice	(74)
past participle modifying a noun	(76)
some core PP arguments	(78 & (79))

Table 4.11: Structures involving missing core arguments

The core argument *the government* with the semantic role of *discover* is realized in (73), and the BAC feature *NP_uncover_curNP* for the core argument *the problems* can be extracted because there is no argument missing from the base argument configuration. However in (74) because the core argument *the government* is not realized the BAC feature cannot be extracted for *the problems*. The sequence *unrealized_NP_uncover_ curNP* is not a valid BAC feature for *the problems* because the argument *the government* is not present.

The predicate verb *propose* is the main verb in (75). The BAC feature *NP_propose_curNP* is extracted for the core argument *the changes*. However in (76), the predicate verb *propose* is a VBN modifier of the core argument *changes* and the subject NP argument is not realized. Thus, the sequence *unrealized_NP_propose_curNP* is not a valid BAC feature for *changes*.

Similarly, because the *toPP* argument *to* 3.28 *billion* is not realized in (78) as it is in (77), the sequence *NP_rise_curNP_unrealized_toPP_fromPP* is not a valid BAC feature for the core argument 4%. And, because the *fromPP* argument *from* 3.16 *billion* is not realized in (79) as it is in (77), the sequence *NP_rise_curNP_toPP_unrealized_fromPP* is not a valid BAC feature for the core argument 4%.

(73) [$_{NP\_thing-discovered}$ The problems] were uncovered by [$_{NP\_discoverer}$ the government].
- **BAC feature for *the problems***: *NP uncover curNP*

(74) [$_{NP\_thing-discovered}$ The problems] were uncovered.
- **? BAC feature for *the government***: *unrealized_NP uncover curNP*

(75) [$_{NP_entity_proposing}$The government] *proposed* the [$_{NP_thing_proposed}$changes] ...
- **BAC feature for *changes***: *NP propose curNP*

(76) the *proposed* [$_{NP_thing_proposed}$changes] ...
- **? BAC feature for *changes***: *unrealized_NP_entity_proposing propose curNP*

(77) [$_{NP_thing_rising}$Sales] *rose* [$_{NP\_amount\_risen}$ 4%] [$_{toPP_start_point}$to 3.28 billion] [$_{fromPP\_end\_point}$ from 3.16 billion].
- **BAC feature for** 4%: *NP rise curNP toPP fromPP*

(78) [$_{NP_thing_rising}$Sales] *rose* [$_{NP\_amount\_risen}$ 4%] [$_{fromPP\_end\_point}$ from 3.16 billion].
- **? BAC feature for** 4%: *NP rise curNP unrealized_toPP fromPP*

(79) [$_{NP_thing_rising}$Sales] *rose* [$_{NP\_amount\_risen}$ 4%] [$_{toPP_start_point}$to 3.28 billion] [$_{fromPP\_end\_point}$·
- **? BAC feature for** 4%: *NP rise curNP toPP unrealized_fromPP*

The next section describes the Level-I back-off features to handle the syntactic structures illustrated in this section.

4.4.2 Level-I back-off features

The preceding section illustrates how base argument configuration features cannot be extracted for the realized arguments in three syntactic structures where there are unrealized arguments. This section presents the Level-I back-off features based on the fact that the unrealized core arguments do not change the grammaticality of the syntactic structures from which they are absent. That is, although there are unrealized core arguments in examples (74), (76), (78), and (79), they are still grammatical utterances. This fact allows the system to extract the features for the realized arguments with the unrealized arguments missing from the features, while the realized arguments are returned to their positions in the base argument configuration.

(80) through (83) are Level-I back-off features corresponding with (74) through (79).

(80) $[_{NP\_thing-discovered}$ The problems$]$ were uncovered.
- **Level-I back-off feature for *the government***: *uncover curNP*

(81) the *proposed* $[_{NP_thing_proposed}$changes$]$ ...
- **Level-I back-off feature for *changes***: *propose curNP*

(82) $[_{NP_thing_rising}$Sales$]$ *rose* $[_{NP\_amount\_risen}$ 4% $]$ $[_{fromPP\_end\_point}$ from 3.16 billion$]$.
- **Level-I back-off feature for** 4%: *NP rise curNP fromPP*

(83) $[_{NP_thing_rising}$Sales$]$ *rose* $[_{NP\_amount\_risen}$ 4% $]$ $[_{toPP_start_point}$to 3.28 billion$]$ $[_{fromPP\_end\_point}$.
- **Level-I back-off feature for** 4%: *NP rise curNP toPP*

When extracting the BAC features and the Level-I back-off features during testing, the feature extractor assumes that the current verb has been seen before. But if the verb has not been seen before, it relies the Level-II back-off features which are based on a set of generic features commonly used in literature.

4.4.3 Level-II backoff features for unseen verbs and all realized core arguments

Section 4.2 presents the BAC features that generalize across various syntactic configurations that the same verb appears in. Section 4.4.2 illustrates the Level-I backoff features that handle the syntactic structures where some core argument is not realized. However, neither the BAC features nor the Level-I features can resolve the unseen verbs during testing because both types of features are dependent on verbs that exist in the training data. Thus, this section presents eleven commonly used features in literature as Level-II backoff features for the current project, which generalize across different verbs (Gildea and Palmer, 2002;

Pradhan et al., 2004; Punyakanok et al., 2005; Toutanova et al., 2005). Features listed in (84) through (93) capture the contextual information surrounding a constituent. Although the **predicate** is also listed as a feature in (94), it does not impose a strong constraint on the rest of the features as the predicate verb does for the BAC and Level-I backoff features because when in a statistical model the weights from features (84) through (93) may balance or even cancel out that from the predicate feature. Due to this reason, the Level-II features can be applied to not only unseen verbs but also **any realized core argument**.

(84) **phrase type** - the syntactic category of the constituent

(85) **path** - the path from the constituent to the predicate

(86) **voice** - the voice of the clause where the constituent is in

(87) **position** - position with respect to the predicate

(88) **head word POS** - POS of the head word of the constituent

(89) **first word/POS** - the first word/POS of the constituent

(90) **last word/POS** - the last word/POS of the constituent

(91) **parent head word/POS** - the head word of the constituent's parent node and its POS

(92) **right sibling phrase type/head/POS** - the syntactic category of the constituent's right sibling node, its head word, the head's POS

(93) **left sibling phrase type/head/POS** - the syntactic category of the constituent's left sibling node, its POS and head word

(94) **predicate** - the predicate verb itself

4.5 Features for non-core argument

Section 2.2.2 defines the eleven types of non-core semantic roles in PropBank. This section illustrates each non-core semantic role with several examples, summarizes the distinctive features for each corresponding non-core argument, and associates the distinctive features with Level-II features defined in section 4.4.3 whenever it can.

(95) **Type: LOC - location**

Example role: Bell, based [$_{PP_ArgM-LOC}$in Los Angeles], makes and distributes electronic, computer and building products.

Distinctive feature(s): based in (verb + preposition)

Example role: Companies would be compelled to publish [$_{PP_ArgM-LOC}$in annual proxy statements] the names of insiders who fail to file reports on time.

Distinctive feature(s): publish-in-statement (verb + preposition + embedded NP head)

Location semantic roles are assigned to PPs modifying a noun or verb. Positions of the location roles vary as the examples above show. Two feature types are summarized from the above examples.

- verb+preposition; sample feature value, *based_in*
- verb + preposition+head of embedded NP, e.g., *publish_in_statement*

The above distinctive features for the LOC role can be represented as the Level-II features as follows:

- verb+preposition $\rightarrow$ {parent head word, head word}
- verb + preposition+head of embedded NP $\rightarrow$ {parent head word, head word, right sibling head word}

(96) **Type: TMP - time**

Example role: Revenue edged up 3.4% to $904 million from $874 million [$_{PP_ArgM-TMP}$ in last years third quarter].

Distinctive feature(s): in-quarter (preposition + NP-head)

Example role: The luxury auto maker [$_{NP_ArgM-TMP}$last year] sold 1,214 cars in the U.S.

Distinctive feature(s): year (head of NP)

Example role: [$_{ADVP_ArgM-TMP}$Currently], to report ...

Distinctive feature(s): currently (head of ADVP)

Constituents of types of PP, NP, ADVP, etc., with a verb as their head, can bear the TMP semantic role. Positions of TMP arguments vary. The TMP role is identified with the meaning/semantic category of the lexical head of a candidate phrase. In cases of PPs, the head preposition also contributes to the identification. Two feature types can be concluded from the examples above.

- preposition+head of the embedded NP, e.g., *in_quarter*

- head word of a temporal phrase, e.g., *year, currently*

The distinctive features for the TMP role can be written as Level-II feature as follows:
- preposition+head of the embedded NP $\rightarrow$ { head word, head word of right sibling }
- head word of a temporal phrase $\rightarrow$ {head word}

(97) **Type: EXT - extent**

Example role: Jaguar **shares** closed [$_{PP_ArgM-EXT}$at 869 pence], up 122 pence, on hefty turnover of 9.7 million shares.

Distinctive feature(s): close-at-pence (verb + preposition + head of the embedded NP)

Example role: But the kids with [$_{RB_ArgM-EXT}$highly] educated parents did 68% less housework than those in less-educated families .

Distinctive feature(s): highly-educated (adverb + adjective)

In the example about the Jaguar shares, the subject *shares* and the verb *close* together select the EXT semantic role of the PP *at 869 pence*. In this example, the argument bearing the EXT role is an NP. In the example about children's homework load in different families, the adjective *educated* modified by the adverb *highly* is gradable. The argument bearing the EXT role in the latter example is an adverb. The following two feature types can be used to identify the EXT roles.

- verb + preposition + head of the embedded NP, e.g., *share_close_at*
- adverb+adjective, e.g., *highly_educated*

The above features for the EXT role can be summarized as the following Level-II features.
- verb + preposition + head of the embedded NP $\rightarrow$ { parent head word, head word, head word of right sibling }
- adverb+adjective $\rightarrow$ { head word, parent head word }

(98) **Type: DIS - discourse connectives**

Example role: [$_{PP_ArgM-DIS}$In addition], further packaging ...have reduced the effects of prepayment risk ...

Distinctive feature(s): in addition-head being main verb (PP+head being the verb in question)

Example role: [$_{CC_ArgM-DIS}$And] the Kennedy amendment would invade not only federal but state sentencings, in two important ways.

Distinctive feature(s): and + external head being main verb (lexical item + head being main verb)

Specific phrases and lexical items whose head is the main verb help to identify the DIS roles. The following feature type may be used for the DIS role.

- preposition + noun head, e.g., *in addition*
- conjunction if its external head word is the verb in question, e.g., *and*

These features can be expressed as a Level-II feature as follows.

- preposition + noun head → { head word, right sibling of head word }
- conjunction → { and }

(99) **Type: CAU - cause**

Example role: All clauses headed by *because*, *because of*, *in light of*, *due to*, etc. take the CAU role.

Distinctive feature(s): preceding lexical items and phrases as heads of clauses that have a causal relationship with the main clause

Head words themselves can be the features indicating a CAU role. The feature type below can be utilized to identify CAU roles.

- head of a clause, e.g., *because*
- idiom , e.g., *due to*

Expressed as Level-II feature, the above features become

- head of a clause → { head word}, and
- idiom → { first word of constituent | head word, head of right sibling}.

(100) **Type: PNC - purpose**

Example role: All clauses headed by *in order to*, *in exchange for*, *to*, *in favor of*, etc. bear the PNC role.

Distinctive feature(s): preceding lexical items and phrases as heads of clauses that are semantically a purpose for the event in the main clause

The distinctive features for the PNC role can be summarized as the Level-II features as follows:

- { head word } for *to*
- { head word, right sibling of head word} for *in order to, in exchange for , etc.*

(101) **Type: MNR - manner**

Example role: Yutaka Kume, who took the helm [$_{PP_ArgM-MNR}$as Nissan's president] in June 1985 , added simply ...

Distinctive feature(s): as-president (preposition+head of embedded NP). More similar examples: grad-attention-with etc.

Example role: Nissan handled the die-hards [$_{PP_ArgM-MNR}$in a typically Japanese fashion] ...

Distinctive feature(s): in-fashion (preposition + head of embedded NP). More similar examples: with-tactics

In both cases, the preposition and the head of the embedded NP identify the MNR role. Two feature types can then be used.

- head preposition of PP + head noun of embedded NP, e.g., *as_president* and *in_fashion*

Expressed as the Level-II features, these are

- head preposition of PP + head noun of embedded NP $\rightarrow$ { head word, head of embedded NP }

(102) **Type: DIR - direction**

Example role: While volatility won't go [$_{ADVP_ArgM-DIR}$away], he said ...

Distinctive feature(s): lexical features: adverbs indicating directions such as *back*, *away*, *upward*, *around*, *aside*, etc.

Example role: The energy ... streamed [$_{ADVP_ArgM-DIR}$into Tokyu Group shares], pushing prices of its companies up across the board .

Distinctive feature(s): stream + into (verb + head of PP)

Direction adverbs can be used as lexical features. If direction adverbs do not appear in a sentence, then use *verb stem+head of PP* as the feature. Some of the sample values are *away* and *stream_into*.

These features can be summarized as the Level-II features as follows.

- adverb $\rightarrow$ { head word }
- verb stem+head of PP $\rightarrow$ { head word of parent, head word}

(103) **Type: NEG - negation marker**

Example role: All negation adverbs, such as *not*, *never*, etc.

Distinctive feature(s): lexical features - negation adverbs

The negation adverbs themselves can be used as lexical features for the role of NEG, for example, *not* and *never*. The Level-II feature of *head word* can be used for the negation adverbs.

(104) **Example role:** All modal verbs, such as *could*, *would*, etc.

Distinctive feature(s): lexical features - modal verbs

Modal verbs themselves can be used as lexical features for the MOD role, for example, *could* and *would*. The Level-II feature of *head word* can be used for the modal verbs.

(105) **Type: ADV - general purpose**

Example role: I [$_{ADVP_ADV}$actually] want to go from Ontario to Chicago

Distinctive feature(s): head word of ADVP

Example role: [$_{SBAR_ADV}$As part of what a Recognition spokeswoman termed an "amiable agreement"], Prospect Group will wind up with control of top management posts $\cdots$

Distinctive feature(s): head word of SBAR

These features can be summarized as one Level-II feature as follows.

- head word of ADVP/SBAR $\rightarrow$ { head word }

The above examples show that the features representing the non-core arguments can be expressed as the Level-II features presented in section 4.4.3. However, no assumption in these features is made about the dependency of non-core arguments on other arguments, unlike the base argument configuration or Level-I backoff features that impose the dependencies among core arguments.

4.6 Features for argument identification

Sections 4.2 through 4.5 illustrate the features for classifying the core and non-core arguments with the correct semantic labels. However, the process of identifying the arguments precedes the classification task, and the process itself requires appropriate features. The BAC and Level-I backoff features require the feature extractor to be able to distinguish between core and non-core arguments. Therefore, the current system uses the Level-II features to identify the core and non-core arguments for each predicate verb. For comparison purposes, the current system also uses the Level-II features to identify all arguments and non-arguments, which all existing practice.

4.7 Chapter summary

This chapter lays out the feature extraction plans for identifying and classifying the arguments. The illustrated plans reveal the three areas in which the current system approaches the SRL task differently from other systems. To sum up, the current system explicitly impose context dependence on core arguments of a predicate verb. Using the base argument configuration features, the current project generalizes across different syntactic configurations of the predicate. To construct the base argument configuration of a given verb, the current project utilizes the concepts from both constituency grammar and dependency grammar. To implement these steps, the current system identifies core and non-core arguments of a given predicate verb while existing systems only identifies arguments and non-arguments. The next chapter incorporates the feature design into a statistical learning framework.

Chapter 5
System Description

Chapter 4 summarizes the three main directions in feature engineering found in previous SRL systems based on constituency grammar representation and identifies three less researched areas that the current project focuses on investigating, areas including how to generalize across different syntactic structures/alternations of the same verb, how to handle the core semantic and non-core semantic arguments separately by imposing the context dependency constraint on the core arguments, and how to utilize dependency grammar relations while using the constituent grammar representation. Chapter 4 presents three levels of feature engineering to address these areas and to handle unrealized core arguments and unseen verbs. The current chapter spells out how the preceding feature engineering for argument identification and classification is integrated into the statistical learning framework.

To achieve this goal, section 5.1 recapitulates the main areas the current project intends to address and gives an overview of the system architecture designed to address these issues. The sections that follow presents the main components of the system. Specifically, section 5.2 describes the argument identification module that builds a 2-way identifier simulating traditional argument identifiers and builds a 3-way identifier specifically for the current project. Section 5.3 presents the argument classification module. Section 5.4 illustrates the statistical models/classifiers for the argument identification and classification tasks. Section 5.5 illustrates how the statistical classifiers are incorporated into the main components of the current system.

5.1 Current SRL system

Chapter 4 illustrates the three less explored areas in previous SRL systems and the corresponding solutions by the current project. These areas and solutions are recapitulated as follows. In the first area, the current project investigates how to generalize across different syntactic configurations of a verb by assigning the realized core semantic arguments to a **base argument configuration** which corresponds to one of the verb's argument structures. The investigation in the first area requires distinguishing core arguments from non-core arguments because only core arguments are the participants in a base argument configuration

1. Argument identification

- 1.1 Training
  - 1.1.1 Build a statistical classifier to classify candidate arguments into core, non-core, or non arguments
- 1.2 Identifying arguments in test data
  - 1.2.1 Parse test data using the Charniak parser
  - 1.2.2 Identify the candidate argument(s) for each verb
  - 1.2.3 Classify the candidate(s) as a core, non-core, or non argument

2. Argument classification

- 2.1 Training
  - 2.1.1 Build a statistical classifier to assign a semantic role label to the identified arguments in step 1.2
- 2.2 Predicting the semantic role labels in test data
  - 2.2.1 Handling core arguments
    - 2.2.1.1 Extract the BAC feature for a core argument
    - 2.2.1.2 Back-off to Level-I feature(s) if necessary
    - 2.2.1.3 Back-off to Level-II feature(s) if necessary
  - 2.2.2 Handling non-core arguments
    - 2.2.2.1 Extract Level-II features for non-core arguments
  - 2.2.3 Assign semantic role labels to the arguments using the classifier Trained in 2.1.1

Figure 5.1: System Architecture

of a given verb. Thus, the second area that the current project investigates is whether or not imposing context dependence on the core arguments but not on the non-core arguments is a feasible approach. Since it requires the knowledge of dependency/grammatical relations, such as subject-verb, verb-object, modifier-head, etc., to form a base argument configuration with respect to the verb but such relations are not used as features explicitly, the third area investigates whether such implicit usage of the dependency relations is effective. The investigations in the three areas are unified into the BAC features when all the core arguments of a predicate verb are present. In addition to handling the three less-researched areas with the BAC features, the current project extracts the Level-I back-off features to handle the situation where one or more of the verb's core arguments are missing. To deal with the unseen verbs, the most general Level-II features are utilized. Chapter 4 analyzes the features associated with the non-core semantic arguments and chooses the Level-II features for the non-core arguments. The above feature design and solutions are incorporated into the system illustrated in figure 5.1.

The two main modules of the current systems are the **argument identification** and the **argument classification** modules. The next two sections illustrate the two modules in turn.

Argument Identification Module

Top level procedures:

```
Input:
  A parsed sentence
  All predicate verbs in the sentence
  A list of all the constituents in the sentence

Output:
  An ordered list of core arguments for each verb
  A list of non-core arguments for each verb

1. For each verb, do
2.    For each constituent, do
3.      Determine if this constituent is a candidate argument of the  verb.
            3.1 if yes, then go to step 4
            3.2 if no, then go to step 2
4.       Create an instance for the candidate argument-verb pair.
5.       Extract the Level-II features for this instance.
6.       Classify the instance based on the features.
7.       Assign a core, non-core, or non-argument label to the candidate argument.
8.      If core argument is assigned, then add this argument to the core-argument list
          of the current verb.
9.      If non-core argument, then assign the argument to the non-core argument list
          of the current verb.
10.    Done.
11. Done.
```

Procedures to determine candidate arguments:

```
Input:
  A constituent
  Current verb

1 If the parent of the current constituent is the verb, then return yes
2 Otherwise,check if the constituent is a moved argument, displaced argument,
   antecedent of relative clause, or in a co-ordinated structure
2.1 If yes, then use heuristics to determine if the constituent is an argument of
     the verb. If yes, then return yes.
3. Return no.
```

Figure 5.2: The Argument Identification module

5.2 Argument identification

Like all previous SRL systems, the argument identification module starts with selecting the appropriate syntactic constituents to be the candidate semantic arguments for a given predicate verb. The current system selects four types of syntactic constituents to be candidate semantic arguments. The first type of syntactic constituents include the syntactic arguments, adjuncts, and discourse markers whose head is the predicate verb. Modal verbs that are the head of the predicate verb belong to the second type of grammatical constituents. The negation adverbs that share the same syntactic head with the predicate verb are the third type of constituents. The fourth type of syntactic constituents are the NPs pre-modified by a *-ing* verb or a past participle, that is, the predicate verb itself.

However, when the preceding four types of candidate semantic arguments are selected, **unlike** previous systems that classify them as *arguments* or *non-arguments*, a 3-way classification, the current identification module classifies each candidate into a **core argument**, **non-core argument**, or **non-argument**. This 3-way classification separates the core arguments from the non-core arguments so that the context dependence constraint can be explicitly imposed on the core arguments but not on the non-core arguments during the classification phase described in the next section.

The foregoing procedures of the argument identification module are shown in figure 4.1 and are repeated in figure 5.2.

The features used to train the identification classifier are the Level-II features described in (84) through (94), repeated in (106) through (116) below.

(106) **phrase type** - the syntactic category of the constituent

(107) **path** - the path from the constituent to the predicate

(108) **voice** - the voice of the clause where the constituent is in

(109) **position** - position with respect to the predicate

(110) **head word/POS** - the head/POS of the constituent

(111) **first word/POS** - the first word/POS of the constituent

(112) **last word/POS** - the last word/POS of the constituent

(113) **parent head word/POS** - the head word of the constituent's parent node and its POS

(114) **right sibling phrase type/head/POS** - the syntactic category of the constituent's right sibling node, its head word, the head's POS

(115) **left sibling phrase type/head/POS** - the syntactic category of the constituent's left sibling node, its POS and head word

(116) **predicate** - the predicate verb itself

(106) through (93) are the features commonly used for the argument identification task in literature. Using these features, the current argument identifier trains a regularized logistic regression classifier for the identification task. The statistical classifier is discussed in section 5.4.

Examples (117) through (119) illustrate the identification process. (117) is an unannotated input sentence. In (118), the bracketed syntactic argument *The economic* $\cdots$ *nations*, the modal verb *will*, and the adjuncts *in Australia*, *next week*, and *to discuss* $\cdots$ *telecommunications* are selected as the candidate semantic arguments of verb *meet*. And finally, the classifier identifies the candidates as core, non-core, or non argument accordingly, the result of which is shown in (119).

(117) The economic and foreign ministers of 12 Asian and Pacific nations will meet in Australia next week to discuss global trade as well as regional matters such as transportation and telecommunications.

(118) $[_{NP}$The economic and foreign ministers of 12 Asian and Pacific nations$]$ $[_{Model}$will$]$ meet $[_{PP}$ in Australia$]$ $[_{NP}$ next week$]$ $[_{To\_VP}$ to discuss global trade as well as regional matters such as transportation and telecommunications$]$.

(119) $[_{core}$The economic and foreign ministers of 12 Asian and Pacific nations$]$ $[_{non-core}$will$]$ meet $[_{non-core}$ in Australia$]$ $[_{non-core}$ next week$]$ $[_{non-core}$ to discuss global trade as well as regional matters such as transportation and telecommunications$]$.

In (119), the syntactic argument NP *The economic and foreign ministers of 12 Asian and Pacific nations* is detected as a core argument for the predicate verb *meet*. The following features may be used by the argument identifier to identify this NP as a core argument:

phrase type *NP*

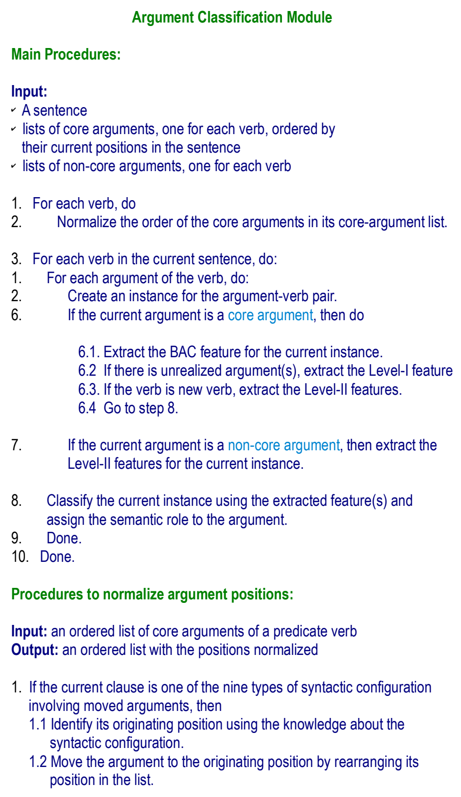
head/POS *ministers/NNS*

parent head/POS *will/MOD*

predicate *meet*

...

The current system implements a 2-way argument identifier and a 3-way argument identifier. The statistical models will be described in section 5.5 after such models are described in section 5.4.



```
Argument Classification Module

Main Procedures:

Input:
✓ A sentence
✓ lists of core arguments, one for each verb, ordered by
  their current positions in the sentence
✓ lists of non-core arguments, one for each verb

1.  For each verb, do
2.       Normalize the order of the core arguments in its core-argument list.

3.  For each verb in the current sentence, do:
1.     For each argument of the verb, do:
2.          Create an instance for the argument-verb pair.
6.          If the current argument is a core argument, then do

                6.1. Extract the BAC feature for the current instance.
                6.2  If there is unrealized argument(s), extract the Level-I feature
                6.3. If the verb is new verb, extract the Level-II features.
                6.4  Go to step 8.

7.          If the current argument is a non-core argument, then extract the
            Level-II features for the current instance.

8.     Classify the current instance using the extracted feature(s) and
       assign the semantic role to the argument.
9.     Done.
10.  Done.

Procedures to normalize argument positions:

Input: an ordered list of core arguments of a predicate verb
Output: an ordered list with the positions normalized

1.  If the current clause is one of the nine types of syntactic configuration
    involving moved arguments, then
    1.1 Identify its originating position using the knowledge about the
        syntactic configuration.
    1.2 Move the argument to the originating position by rearranging its
        position in the list.
```

Figure 5.3: The Argument Classification module

5.3 Argument classification

The argument classification module runs after the identification module to assign a semantic role label to each identified core or non-core argument. The classifier begins with extracting feature(s) for each core and non-core argument. As outlined in figure (5.1), for a core argument, the argument classifier extracts the BAC feature and two levels of back-off features when necessary. For a non-core argument, the argument classifier extracts only the Level-II features. Separating the feature representations of the core and non-core arguments differs the current system from all previous SRL systems.

Specifically, the three levels of feature extraction are intended to achieve three goals. Firstly, the BAC feature generalizes across different syntactic structures/alternations of the same predicate verb. This generalization helps when one syntactic structure/alternation has never been seen in the test data but can still be handled by using the BAC feature generated from a different syntactic structure of the same predicate verb. The BAC feature for a core argument imposes the constraint of context dependence on this argument. Section 4.3 illustrates the generalization process. Secondly, the Level-I features handle the situation where some core argument(s) that the current core argument are dependent on is(are) missing.

Section 4.4.1 illustrates this case. Thirdly, the Level-II features are extracted because they generalize across different verbs when the current predicate verb has not been seen before and BAC and Level-II features are thus not available.

After the feature extractor extracts the corresponding features for the core and non-core arguments, the argument classifier assigns a semantic role label to the core and non-core arguments. The procedures of the argument classifier are summarized in figure 4.2 and are repeated in figure 5.3. For example, the core and non-core arguments identified in (119) are assigned with the semantic role labels in (120).

(120) [$_{MEETER}$The economic and foreign ministers of 12 Asian and Pacific nations] [$_{Model}$will] meet [$_{LOCATION}$ in Australia] [$_{TEMPORAL}$ next week] [$_{PURPOSE}$ to discuss global trade as well as regional matters such as transportation and telecommunications].

The current system builds four argument classifiers with four combinations of the BAC, Level-I, and Level-II features. These classifiers are described in section 5.5 after the statistical models are introduced in section 5.4.

The statistical models are described in the next section.

5.4 Statistical models and classifiers

This section introduces the statistical methods the current project constructs for the argument identification and classification tasks. Regularized logistic regression models are built for both the identification and classification tasks (section 5.4.2). KNN classifiers (5.4.3) and Naive Bayes classifiers (section 5.4.4) are also built to compare against the logistic regression models. The next section describes how these statistical models are integrated into various identification and classification components of the current system.

Table 5.1 summarizes the notations used in this project.

5.4.1 The feature function

The current project defines the indicator function in (5.1) to integrate the features for both argument identification and classification tasks.

$$f_i(y,c) = \begin{cases} 1 & \text{if some condition(s) is/are satisfied} \\ & \text{for the } i_{th} \text{ feature of } c \\ & \text{with respect class label } y \text{ ;} \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

a	a realized argument corresponding to a syntactic argument an adjunct, a modal verb, a discourse marker, or a negation adverb
r	a semantic role label assigned to a
R	a set of all possible semantic role labels $\{r_0, \cdots r_l,\}$
$f_i(r,a)$	a feature function for the argument a with role r
$f(r,a)$	a set of feature functions corresponding to the set of features defined for argument a
t	a semantic argument type
T	the set of all possible argument types $T = \{r \mid r \in \{core, non-core, null\}\}$
y	a generic class/type label
c	a syntactic constituent
C	a set of grammatical units defined in PropBank that are realized semantic arguments $C = \{c \mid c \in \{syntacticargument, adjunct, modelverb, discoursemarker, negationadverb\}\}$
$\mathscr{T}$	the set of all training instances

Table 5.1: Notations

The following are several example feature functions for both identification and classification tasks.

For example, the feature function in (5.3) corresponds to the semantic argument *Dana* realized as the subject NP syntactic argument in (122).

(121) $[_{NP\_core}$**Dana**$]$ opened $[_{NP\_core}$ the door$]$.

(122) $[_{NP\_Agent}$**Dana**$]$ opened $[_{NP\_thing\_open}$ the door$]$.

$$f_i(t = core, c = Dana) = \begin{cases} 1 & \text{if } headofparent = open \text{ and } predicate = open \\ & \text{and } t = core \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

$$f_i(r = Agent, a = Dana) = \begin{cases} 1 & \text{if base argument configuration} = curNP\_open\_NP \\ & \text{and } r = Agent \\ 0 & \text{otherwise.} \end{cases} \tag{5.3}$$

Similarly, the feature function in (5.5) corresponds to the semantic argument *last year* realized as an NP adjunct in (124).

(123) The luxury auto maker [$_{NP\_non-core}$ **last year**] sold 1,214 cars in the U.S.

(124) The luxury auto maker [$_{NP\_TEMPORAL}$ **last year**] sold 1,214 cars in the U.S.

$$f_i(t = noncore, c = lastyear) = \begin{cases} 1 & \text{if last word of constituent} = year \\ & \text{and } t = non-core \\ 0 & \text{otherwise.} \end{cases} \tag{5.4}$$

$$f_i(r = Temporal, a = lastyear) = \begin{cases} 1 & \text{if last word of constituent} = year \\ & \text{and } r = Temporal \\ 0 & \text{otherwise.} \end{cases} \tag{5.5}$$

5.4.2 Modeling with regularized logistic regression

The current project incorporates the feature function defined in (5.1) into the the multi-class logistic regression model shown in equation 5.6.

$$p(y \mid c) = \frac{\exp\{\sum_{i=0}^{k} w_{ri} f_i(y,c)\}}{\sum_{y' \in Y} \exp\{_{i=0}^{k} w_{y'i} f_i(y',c)\}} \tag{5.6}$$

The optimal model weights are found through equation (5.7)

$$\hat{w} = \arg\max_{w} \sum_{i} \log P(y^{(i)} \mid c^{(i)}) - \alpha \sum_{j=0}^{k} w_j^2 \tag{5.7}$$

The current project relies on Java version of the LIBLINEAR package implementation (Lin, 2008) of regularized logistic regression to construct the models to identify the core and non-core arguments and to predict their semantic labels.

5.4.3 K-nearest neighbor classifier

Following Aha and Kibler (1991), the K-nearest neighbor classifier used in the current project consists of three components, a similarity function, a concept description building module, and a classification function.

The similarity between two constituents, c_i and c_j, is defined as the negation of their Euclidean distance. Equation (5.8) shows the calculation of the similarity,

$$Similarity = -\sqrt{\sum_{k=1}^{n} s(c_{ik}, c_{jk})} \tag{5.8}$$

where c_{ik} is the k-th feature of c_i and the indicator function $s(c_{ik}, c_{jk})$ is defined in (5.9).

$$s((c_{ik}, c_{jk})) = \begin{cases} 1 & \text{if } c_{ik} = c_{ik} \\ 0 & \text{otherwise.} \end{cases} \tag{5.9}$$

The concept description building process is the training process, and the results that are the correctly classified training instances along with their class labels are saved in the **concept description** $\mathscr{C}$. This process is listed in table 5.2.

`KNN Training`	
`Input`	training instances $\mathscr{T}$
`Output`	Concept description $\mathscr{C}$ - the set of correctly classified training instances
`Initialize`	$\mathscr{C} \leftarrow$ K-instances for each class from $\mathscr{T}$
`Loop`	for each instance $x \in \mathscr{T}$ 1. Find the K-nearest neighbors $(n_1 \cdots n_k)$ to x in $\mathscr{C}$, using Similarity(x, n_i) 2. **If** $(n_1 \cdots n_k)$ classify x correctly by majority voting and assign label y to x **then** $\mathscr{C} \leftarrow \mathscr{C} \cup (x, y)$ **else** discard x
`Done`	

Table 5.2: Procedures for training a K-nearest neighbor classifier

Given a new instance and the concept description $\mathscr{C}$ learned from the training process in table 5.2, the classification can be performed by the majority voting process shown in table 5.3

`KNN Classification`	
`Input`	Concept description $\mathscr{C}$, a new instance c
`Output`	a class label y for x
`Do`	
	1. Find the K nearest neighbors $(n_1 \cdots n_k)$ for x in $\mathscr{C}$, using Similarity(x, n_i) 2. Assign a class label y to x by majority voting among $(n_1 \cdots n_k)$

Table 5.3: Classification by majority voting

The Weka (Witten and Frank, 2005) implementation of the KNN classifier is integrated into the current system.

5.4.4 Naive Bayes classifier

Given a grammatical unit c of v and the feature vector $\vec{f}_c$ for c, the probability of a class label y given $\vec{f}_c$ is calculated using Equation (5.10).

$$\begin{aligned} P(y|\vec{f}_c) &= \frac{P(\vec{f}_c|y)P(y)}{P(\vec{f}_c)} \\ &\approx \frac{\prod_{i=1}^{n} P(f_i|y)P(y)}{P(\vec{f}_c)} \end{aligned} \tag{5.10}$$

The naive Bayes classifier determines the class label of a grammatical unit c using Equation (5.11).

$$\begin{aligned} \hat{y} &= \arg_{y\in Y} \max P(y|\vec{f}_c) \\ &= \arg_{y\in Y} \max \frac{\prod_{i=1}^{n} P(f_i|y)P(y)}{P(\vec{f}_c)} \\ &= \arg_{y\in Y} \max \prod_{i=1}^{n} P(f_i|y)P(y) \end{aligned} \tag{5.11}$$

$$P(f_i|y) = \frac{counts(f_i, y) + 1}{counts(y) + counts(f_i)} \tag{5.12}$$

$$P(y) = \frac{counts(y)}{\sum_{y'\in Y} counts(y')} \tag{5.13}$$

The Laplace smoothing is used in equation (5.12). The current project integrates the Weka (Witten and Frank, 2005) implementation of the naive Bayes classifier into the system.

5.5 Classifiers and components

In order to implement the system described in the preceding three sections, the current project builds seven classifiers/components, including a baseline argument identifier, a baseline argument classifier, a system argument identifier, and four fundamental argument classifiers. Table 5.4 lists these components.

Classifiers & Components	Procedures/Description
Baseline arg. identifier	Identify semantic arguments from parse trees; 2-way classification: argument/non-argument.
Baseline arg. classifier	Classify semantic arguments from the argument identifier, treating all core and non-core arguments detected by the argument identifier as arguments using only the Level-II features.
System argument identifier	Identify semantic arguments from parse trees; 3-way classification: core, non-core, and non arguments.
Four fundamental argument classifiers	**A**. Classify core args. w/ BAC and non-core args. w/ Level-II features **B**. Classify core args. w/ BAC + Level-I back-off features and non-core args. with Level-II features **C**. Classify core args. w/ BAC + Level-I+ Level-II back-off features, and non-core args. with Level-II features **D**. Classify core args. w/ BAC + Level-I &, and use voting w/ Level-II back-off features if BAC and Level-I features not available, and non-core args. with Level-II features

Table 5.4: System components

The baseline argument identifier is a logistic regression (LR) 2-way classifier that classifies the candidate arguments into argument/non arguments. The baseline argument classifier is an LR or KNN classifier. Both the baseline identifier and classifier are trained on Level-II features.

The system argument identifier is an LR 3-way classifier that classifies the candidate arguments into core, non-core, or non arguments.

The four fundamental argument classifiers perform the classification tasks based on the features designed to address the fore-mentioned less-researched areas. Specifically, the BAC features are intended to generalize across different syntactic configurations of a predicate verb. The BAC features are at the same time constraints imposed on the core arguments of a predicate verb so that their positions in their actual sentences are brought to conform with those in a base argument configuration of the verb. In order to derive the BAC features, as shown in section 4.3 the knowledge from both the constituency grammar and dependency grammar are needed to correctly analyze all involved grammatical relations between the arguments and the predicate verb.

The first fundamental classifier classifies the core arguments based only on the BAC features. The performance of this classifier indicates how effectively the BAC features represent the data and more specifically how well the BAC features generalize across different syntactic structures/alternations of the predicate verbs. The second fundamental classifier classifies the core arguments based on the BAC features, and backs off to the Level-I back-off features in cases where some argument of the verb is not realized. The performance of this classifier then measures how well the Level-I back-off features handle the situations where there are missing arguments. The third fundamental classifier tries to classify the core arguments using only the BAC and Level-I features and backs off to the most general

Level-II features when the first two types of features are unavailable. The performance of this classifier indicates how well the system handles the cases where the predicate verb has never been seen during training. The fourth fundamental classifier starts with the BAC and Level-I features. When neither is available, it backs off to the Level-II features. Unlike the third fundamental classifier, when using the Level-II features, it makes the prediction based on the majority voting from three different algorithms, including a logistic regression model, a nearest-neighbor classifier, and a naive Bayes classifier. All four fundamental classifiers rely on the Level-II features to classify the non-core arguments.

The preceding sections illustrate how the feature design described in chapter 4 is incorporated into the current SRL system. The next chapter tests the effectiveness of the current feature and system design.

Chapter 6
Experiments, Results, and Discussion

Chapter 4 proposes the base argument configuration features that unify the current system's novel approaches to investigating the three less explored areas in the field of semantic role labeling, including how to develop features that generalize across the syntactic variations that a verb appears in, how to utilize the context dependence among the semantic roles of the core semantic arguments, and how to utilize both the constituency and dependency grammar concepts in the same system. In addition to inquiring into the preceding areas, chapter 4 proposes the Level-I back-off features to handle the unrealized arguments and the Level-II back-off features to deal with the unseen verbs. Chapter 5 incorporates the BAC, Level-I, and Level-II features into an argument classifier and combines it and an argument identifier into a complete SRL system. The current chapter evaluates how effectively the novel approaches to the three areas solve the classification task through the BAC features, how adequately Level-I and Level-II features handle the unrealized arguments and unseen verbs respectively, and how the argument identifier affects the argument classifier in the complete system. The evaluations are mainly conducted through three sets of experiments and through comparisons with the baseline systems, which are implementations of the standard approaches to the argument identification and classification tasks. The comparisons with the baseline systems show that the current feature engineering and system design are effective for the argument classification task, and are hence effective for solving the three intended less-explored areas in the field of SRL. At the same, the comparisons with the state-of-the-art performance demonstrates the necessity to improve the current argument identifier. A sketch of the plan for the evaluations follows.

Following a brief review of the three less-explored areas, section 6.1 summarizes the novel approach to each area as well as the the complete system as a new approach to SRL task. Section 6.2 lists the training and test data and lays out the plan for five sets of experiments. Section 6.3 reports on the experiments with the 2-way and 3-way argument identifiers. The experiments with the baseline system, including the baseline argument identifier and classifier, are described in section 6.4.

In section 6.5, the first set of experiments perform the argument classification task on the pre-annotated arguments with four different settings of the BAC and the two levels of back-off features. A baseline argument classifier approximating the common argument classification method found in literature is also built and performs the same classification task. The results from the current argument classifier and the baseline classifier are compared. The results from the current argument classifier are also compared with two state-of-the-art argument classifiers. The comparisons show that the current approaches to the role labeling task is effective.

In section 6.6, the second set of experiments examine the current SRL system as a complete system. First, the argument identifier identifies the core, non-core, and non arguments from the gold syntactic parse trees in WSJ section 23. Then, the argument classifier assigns the semantic role labels to the identified arguments. The results are compared against a baseline system that simulates a standard basic SRL system. Although the results from the complete system are above those of the baseline system, the classification results are affected by the identification results. At the same time, the argument classifier itself remains as effective.

In section 6.7, the same set of experiments are repeated, with the input to the argument identifier changed to automatically parsed data using the Charniak parser. This set of experiments yield similar results to the last set of experiments. However, the overall F-measure of 70.9% is below the state-of-the-art on WSJ section 23. This becomes the motivation to immediately improve the performance of the argument identifier because the argument classifier itself still remains effective.

6.7.1 summarizes the experiments and discusses the necessity to improve the argument identification module.

6.1 The less-explored areas and novel solutions

This section reviews the three less-explored areas and summarizes the current new solutions to each area. Then, the complete SRL system of the current project is summarized.

As illustrated in section 3.2, the research issue of how to generalize across different syntactic variations of a predicate verb arises in the following situation: new syntactic configurations regarding the verb are encountered in the test data. But the original feature set tuned for the training and development data is not designed to cover such configurations. More feature types have to be designed to cover these newly discovered configurations. As a result, the design of the new configuration-specific features leads to increased feature space. And yet, other unseen syntactic configurations regarding the same verb showing up in the test data will still not be handled. Consider verb *v*. If both active and passive forms of *v* are found in the training data, one can design an active feature and a passive feature to cover the two configurations respectively. But, if it is possible to contrive a feature to cover both cases, one could achieve a smaller feature space. At the same time, if the system has only seen the passive configurations of *v* in the training data, then it would not be able to generalize over the active configurations in the test data. To be able to generalize across different syntactic configurations of the same verb and, as a result, to minimize the feature space, it is necessary to find the common ground spanning across all syntactic configurations of the verb.

Section 3.2 also shows that syntactic frame features involving the listing of all the arguments of a predicate verb have been used by a number of systems, such as Gildea and Jurafsky (2000), Xue and Palmer (2004), and Toutanova et al. (2005), assuming context dependence among the arguments, without distinguishing between core and non-core arguments because none of the systems paid close attention to the **theory of argument realization**, the updated **linking theory**, which implies that the context dependence exists only between the core semantic roles of the core semantic arguments. Therefore, the research issue of how to impose the context dependence constraint on only the core semantic arguments arises.

Sections 3.1 and 3.2 extensively describe the features employed in the SRL systems based upon constituency grammar. Section 3.3.2 juxtaposes the features utilized in the systems based on dependency grammar. Just as the concept of constituency or phrase structures are not found in dependency grammar, constituents and phrase types are absent from the dependency grammar-based features, which are only composed of the word-to-head grammatical relations illustrated in section 3.3.2. On the other hand, the constituency grammar-based features do not explicitly contain grammatical relations between the heads, although such relations can be derived from the constituents. Therefore, the two families of features are mutually exclusive. While the SRL researchers have been investigating whether the dependency grammar-based features are more effective than the constituency features and hence can eventually replace the latter ones for the SRL task since CoNL-2008 and continuing on to CoNLL-2009, the third research issue of how one can incorporate both types of grammars into one system and benefit from the combination arises.

Now that the three open areas that current projects focuses on addressing are summarized. The novel solution for each area is reviewed in the following.

To represent the common ground across different syntactic structures that a verb appears in, the current system utilizes the positions/positional configurations of the arguments in one of the verb's argument structures. Such positional configurations make up the base argument configuration features in the current system and provide the solution to the first open research question. The context dependence among the semantic roles of the core arguments of the verb specifies that the positional configurations of the core arguments in relation to the verb are associated with a specific list of semantic roles. Imposing this specification as a strong constraint, the system requires that only core semantic roles can be entered in the base argument configurations, which gives the solution to the research issue of how to impose the context dependence on the core semantic arguments. The solution to the research issue of how to combine the constituency and dependency grammars unfolds naturally with the procedures to construct the base argument configurations for the syntactic structures that involve movements. This is so because, as shown in section 4.3.1, constructing the base argument positions for the moved arguments requires the knowledge of both the constituency of the arguments and the grammatical relations between the semantic arguments and the predicate verb. Such grammatical relations overlap with those defined in dependency grammar. Therefore, the procedures to derive the base argument configurations from the syntactic structures involving transformations incorporate both dependency and constituency grammars, although the dependency relations defined in dependency grammar do not show up in the base argument configuration features.

Together with other components of the system, the solutions above distinguish the current system from previous SRL systems in terms of feature design and system architecture. First, the base argument configuration features unify the solutions to the three open research issues, generalize across different syntactic configurations that a given verb may appear in, and lead to a limited feature space. Second, the current system identifies core and non-core semantic arguments in order to precisely describe the core arguments involved in a base argument configuration of a given predicate verb as well as to precisely represent the context dependence that exist only among the core semantic arguments. Third, the system is designed in a way so that the BAC features can be conveniently combined with the Level-I and Level-II back-off features that handle the missing arguments and the unseen verbs in the test data.

Now that the open research issues and the solutions have been presented and the resulting differences in the current feature and system designs are shown, it is time to demonstrate how effectively the designs handle the intended scenarios. The next section describes the data and experiments for this purpose.

6.2 Data and experimental designs

This section starts with a summary of the training, development, and test data for the experiments. And the experimental plan follows.

6.2.1 The data sets

Table 6.1 lists training and test data for the current project. All data sets are identical to those for the shared task of SRL on CoNLL-2005.

Data	Description
Training data	Penn Treebank sections 2 through 22 with PropBank annotations
Development data	Penn Treebank section 24 with PropBank annotations
Test data set #1	Penn Treebank section 23 with PropBank annotations

Table 6.1: Experimental data

6.2.2 The experimental plan

The present section briefly describes the five sets of experiments to evaluate the current feature and system designs. The plan per se is summarized in table 6.2. Descriptions of the component classifiers can be found in section 5.5.

Set	Experiment	Procedures/description
1	Identifying arguments	traditional 2-way identification current 3-way identification
2	Constructing the baseline system	approximating standard SRL through 2-way identification and baseline classification
3	Classifying gold-identified arguments	Run fundamental classifiers on the data with pre-annotated core and non-core arguments. Identify the best classifier.
4	Identification + classification on gold parses	Identify core & non-core args. in gold parses. Then run the best fundamental arg. classifier.
5	Identification + classification on automatic parses	Identify core & non-core args. in automatically parsed trees. Then run the best fundamental arg. classifier.

Table 6.2: Experimental Plan

In the first set of experiments, a 2-way argument identifier simulating the standard traditional argument identification is built and evaluated. That is, the 2-way argument identifier only classifies the candidate grammatical constituents into arguments or non-arguments of a given verb. A 3-way argument identifier unique to the current project is also built and evaluated. The 3-way identifier classifies the candidate arguments into core and non-core arguments, and non-arguments of a given verb.

The second set of experiments build and evaluate the baseline systems, simulating the standard two-step identification and classification approach.

The third set of experiments run the fundamental argument classifiers over the gold-identified arguments, using the different combinations the BAC, Level-I, and Level-II features. The fundamental argument classifiers are described in section .

The fourth set of experiments start with identifying the semantic arguments from the gold parses of the WSJ section 23 using the 3-way argument identifier and classify the arguments using the fundamental argument classifiers.

In the fifth set of experiments, the 3-way argument identifier first detects the core and non-core arguments from the automatically parsed WSJ section 23 and then classify the arguments with the fundamental argument classifiers.

The experiments are discussed in the next five sections.

6.3 Experiments with the argument identifiers

In this set of experiments, a two-way argument identifier and a three-way argument identifier were built and tested separately. The two-way argument identifier following the traditional identification method to classify the candidate arguments into arguments or non arguments was built. The three-way argument identifier classified the candidate arguments into core, non-core, and non arguments. Both identifiers were trained with the logistic regression model. Both identifiers relied on the Level-II features and were used to detect the arguments from the gold parses of WSJ section 23 and automatically parsed results of sectin 23 using Charniak parser. The precision, recall, and F-measures from the identification experiments are shown in table 6.3.

Identifier	Data	Classifier	Precision	Recall	F-measure
2-way	gold parses	Logistic regression	92.16	89.47	90.80
2-way	Charniak parses	Logistic regression	82.51	80.25	81.36
3-way	gold parses	Logistic regression	90.31	86.42	88.32
3-way	Charniak parses	Logistic regression	80.88	77.36	79.08

Table 6.3: Argument identification results

Several observations can be made from the identification experiments. First, the F-measures from the identification by both the 2-way and 3-way identifiers on gold parses are higher than those from the Charniak parses. This makes sense because there are fewer constituents in the Charniak parses corresponding to the Propbank argument boundaries than those in gold parses. Second, the identification results by the 3-way identification results are lower than those by the 2-way identification results. This is because the 3-way classification task is slightly more challenging than the 2-way classification task.

The SRL systems since CoNLL-2005 do not usually report on the identification performance. Hence, there would be no direct comparison with the identification results by the state-of-the-art SRL systems on WSJ section 23 since CoNLL-2005. However, several state-of-the-art systems tested on the CoNLL-2004 version of the WSJ section 23 did report on the identification performance. For example, Pradhan et al. (2004) obtained an F-measure around 93% on the gold parses and around 86% on the automatic parses. These scores are not comparable with the identification results by the 2-way identifier shown in table 6.3 because there are fewer arguments annotated for the CoNLL-2004 version of section 23 than for the CoNLOL-2005 version.

6.4 Experiments with the baseline system

The baseline SRL system shown in table 6.2 consists of a baseline argument identifier and a baseline argument classifier. The identifier is the 2-way identifier described in the preceding section. The baseline argument classifier utilizes the set of features commonly found in literature, such as Pradhan et al. (2004), Pradhan et al. (2005), Toutanova et al. (2005), and Surdeanu et al. (2007), which are the Level-II features in the current project. The two-step approach is the standard approach found in literature. More sophisticated techniques, such as re-ranking or joint-inferencing, are not used in the baseline argument classifier.

The baseline system was applied to both the gold parses and automatic parses. Section 6.5.1 reports that the argument classifier of this baseline system achieves the performance close to the re-ranking-based state-of-the-art argument classifier. The results from the baseline argument classifier and the overall baseline results are compared with several systems in the next three sections.

6.5 Experiments with argument classification

The argument classification module shown in figure 5.1 takes the core and non-core arguments identified by the identification module and assigns a semantic role label to each argument. In order to test the design and effectiveness of the classification module independently of the identification step, the current project builds an argument classifier to predict the semantic role labels of the pre-annotated core and non-core arguments from PropBank. The performance of the argument classifier on the pre-annotated arguments forms the performance upper bound of the current system. Because the argument classifier runs on the pre-annotated arguments, it provides for the ideal situations for testing the effectiveness of the current feature engineering and system design. In the next two sections, the results and findings from running the logistic regression and KNN argument classifiers are reported respectively.

6.5.1 Results from the logistic regression classifier

Section 5.4.2 describes the regularized logistic regression classifier the current project builds. This set of experiments applied the logistic regression classifier to WSJ section 23 with pre-annotated core and non-core arguments and predicts the semantic roles of the arguments. The classification results for five experimental settings are shown in table 6.4, along with those of the baseline argument logistic regression classifier.

Experiments	Baseline	1	2	3	4
Features for core arguments	Level-II features only	BAC features	BAC + Level-1 features	BAC + Level-I + maj. voting w Level-II	BAC + Level-I + Level-II features
Ftrs for non-core	Level-II	Level-II	Level-II	Level-II	Level-II
Precision	82.69	91.83	91.54	91.27	88.73
Recall	83.15	84.25	84.94	85.74	89.06
F-measure	82.92	87.88	88.12	88.41	88.89

Table 6.4: Classification results from logistic regression models

Table 6.4 shows that the baseline classifier achieved the precision, recall, and F scores of 82.69%, 83.15%, and 82.92%. The set of scores come quite close to that of the state-of-the-art re-ranking argument classifier in Surdeanu et al. (2007) shown in table 6.6 of section 6.5.4, with the precision, recall, and F-measure scores of 88.08%, 82.84%, and 85.38%. The baseline classifier is then shown to be a quite plausible representation of the standard classification approach.

Columns numbered 1 through 4 correspond to four settings of the current system. Although all four settings extract Level-II features for non-core arguments, they differ in what features they extract for the core arguments.

The first setting extracts only the base argument configuration (BAC) features for the core arguments and achieves the precision, recall, and F-measure of 91.83%, 84.25%, and 87.88% respectively.

If a BAC feature cannot be extracted due to unrealized core-arguments, the second setting adds the Level-I back-off features. This setting slightly increases the recall to 84.94%, indicating that the Level-I features improve the coverage of the features. At the same time, the Level-I features also causes the precision to drop from 91.83% to 91.54%. However, the overall F-measure increases from 87.88% to 88.12%, indicating that the increase in recall overcomes the drop in precision.

The preceding trends in the changes in precision, recall, and F-measure are preserved in the fourth experimental setting, where the most general Level-II back-off features are extracted where the BAC and Level-I features are unavailable. As illustrated in section 4.4.3, the Level-II back-off features are the most general set of features that generalize across unseen verbs during training. This leads to the increase from 84.94% to 89.06% in recall and the increase from 88.12% to 88.89%, with the trade-off of the decrease from 91.54% to 88.73% in precision.

In order to overcome the decrease in precision, when the Level-II features are extracted for the core arguments for which the BAC and Level-I features cannot be extracted, the third setting performs a majority voting among three classifiers trained on the Level-II features for the core arguments. If two or more classifiers agree on the semantic role label they assign to a core argument, then the label is assigned to the core argument as its final semantic role label. The three classifiers include a logistic regression classifier, a naive Bayes classifier, and a *K*-nearest neighbor classifier with *K* set to 3. This approach keeps the precision above 91% while slightly increases the recall and F-measure from the first and second experiments.

The systems with the settings in columns 1 through 4 all obtained a recall at least 2% higher than that of the baseline. The precision and F-measure scores are at least 4% higher than that of the baseline.

6.5.2 Results from the nearest-neighbor classifiers

The same set of experiments in the preceding section were repeated. But this time the nearest-neighbor classifiers were used to classify the pre-annotated arguments. The experimental settings and results are shown in table 6.5. In these experiments, one observes the same trends in the scores as the experiments using the logistic regression models. First of all, all four experiments based on the current feature and system designs achieved more than 9% of improvement over the baseline system in terms of F-measure. Second, the features that are more general added in each of the settings from settings 1 through 3 increased the recall and F-measure scores. And at the same time the precision dropped slightly from each previous setting. Third, the fourth experiment which involved voting among the classifiers over the Level-II features maintained a similar precision as the first experiment and yet increased both the recall and F-measure.

In addition to these observations, one can also see that the nearest-neighbor classifier does a slightly better job than the logistic regression model in the classification task under the current system design. The F-measures of the first four experiments shown in table 6.5 are all higher than those in table 6.4. But, the nearest-neighbor baseline classifier achieved a lower performance than the logistic regression (LR) baseline model, which is also shown in table 6.5.

Experiments	KNN Baseline	Best LR Baseline	1	2	3	4
Features for core arguments	Level-II features only	Level-II features only	BAC features	BAC + Level-1 features	BAC + Level-I + maj. voting w Level-II	BAC + Level-I + Level-II features
Ftrs for non-core	Level-II	Level-II	Level-II	Level-II	Level-II	Level-II
Precision	78.70	82.69	92.42	92.10	91.82	89.24
Recall	79.28	83.15	84.79	85.45	86.25	89.57
F-measure	78.99	82.92	88.44	88.65	88.95	89.41

Table 6.5: Classification results from nearest-neighbor classifiers and the best logistic regression baseline

6.5.3 Discussion: effectively handling three issues

Table 6.4 shows that the experiment with the logistic regression (LR) model using the BAC features alone achieved the precision, recall, and F-measure of 91.83%, 84.25%, and 87.88% respectively. And table 6.5 shows that the experiment with the KNN classifier using the BAC features alone achieved the precision, recall, and F-measure of 92.42%, 84.79%, and 88.44%. The two sets of scores both exceeded the best LR baseline scores with the precision, recall, and F-measure of 82.69%, 83.15%, and 82.92%.

The fact that the KNN and LR models using the BAC features alone reached a precision above 90% indicates that the BAC features quite accurately capture an argument structure of a given verb. At the same time, the fact that both models yielded a recall above 84% indicates that the BAC features have a relatively wide coverage of the data. More specifically, since the BAC features unify the solutions to the three less-explored issues and have been shown effective, the solutions themselves must be effective. That is, the argument positions of a base argument configuration with respect to a specific verb effectively normalize over different syntactic configurations that the verb appears in. The context dependence constraint ensures that only core arguments enter this base argument configuration and that a list of fixed semantic roles are associated with the core arguments in this base argument configuration. Finally, combining the concepts from constituency and dependency grammars are helpful for identifying the moved argument(s) and reconstructing the its/their positions in this base argument configuration.

As the Level-I features were added to the BAC features, the precision scores for both LR and KNN argument classifiers dropped slightly, but the recall scores rose at the same time resulting to the overall increase in the F-measures. This indicates that the Level-I features were able to cover unrealized arguments without sacrificing the overall system performance. Moreover, as Level-II features were added to the BAC and Level-I features, although the precision scores for both classifiers dropped slightly, the recall scores again rose at the same time resulting to the overall increase in the F-measures. This demonstrates that the Level-II features were able to generalize across unseen verbs to a degree without sacrificing the overall system performance.

6.5.4 Comparisons with two other SRL systems

This section compares the results from the current system against those of two systems reported in Surdeanu et al. (2007) on the classification task of the pre-annotated arguments in WSJ section 23. The two systems include a re-implementation of the re-ranking system from Toutanova et al. (2005) and a system by Surdeanu et al. that performs the sequential labeling of the arguments. The rest of the section describes that at least the comparisons with the re-ranking system validate the current feature and system designs centered on the BAC features and the solutions to the three less explored issues that are incorporated into the BAC features.

The systems are chosen for comparison purposes for several reasons. First, these two systems are the only ones for which the performance on the pre-annotated arguments from the CoNLL-2005 test set is reported. Second, the re-ranking system by Toutanova et al. (2005) was the state-of-the-art between 2005 and 2007, with an overall F-measure of 78.5% on on WSJ section 23. Third, Surdeanu et al. (2007) reported the new state-of-the-art performance since 2007, with an overall F-measure of 80% on WSJ section 23, and is the one of the few latest systems utilizing constituency grammar-based representation in literature since CoNLL-2005.

Table 6.6 lists two oracle systems with the lowest and highest performance from the current project. Listed in the table are also the oracle re-ranking-based system and the oracle system that combines multiple strategies presented in Surdeanu et al. (2007).

System	Lowest LR	Highest KNN	Re-ranking system	Combined strategies
Features	BAC only	BAC+Level-I+ Level-II	similar to Level-II + additional ftrs	similar to Level-II +additional ftrs
`Precision`	91.83	89.24	88.08	99.12
`Recall`	84.25	89.57	82.84	85.22
`F-measure`	87.88	89.41	85.38	91.64

Table 6.6: Oracle classification results. **LR**: logistic regression

Table 6.6 shows that both the logistic regression model which yields the lowest F-measure and the KNN classifier which yields the highest F-measure in the current project have some edge over the re-ranking system in terms of precision, recall, and F-measure. Thus, the comparisons with the re-ranking system further verify the BAC-centered approach to the argument classification task of the current system.

At the same time, the system by Surdeanu et al. that utilizes combined strategies achieves an almost perfect precision, while the best precision the current project achieves is 92.42% (see table (6.5)) by the KNN system with the BAC features. However, the best recall score the current project could reach is 89.57% with the logistic regression model using BAC, Level-I, and Level-II features, which is about 4% higher than the 85.22% of the combined strategies by Surdeanu et al. (2007). The highest F-measure that the current project achieves is 89.41%, which is about 2% lower than the system by Surdeanu et al. (2007). It is expected that the system the with combined strategies would have a higher performance because Surdeanu et al. apply sophisticated strategies to three complete SRL systems to obtain the state-of-the-art performance (see section 3.2.4.2). However, each of the three SRL systems does not yield the state-of-the-art performance. Surdeanu et al. (2007) do not report an oracle performance for each of the three systems rather than that for the combined strategies. But, they do report each system's performance on the automatically parsed data, each of which is $2-3\%$ lower than the final score of 80.56% in F-measure (see section 3.2.4.2). Judging by this difference, the individual oracle system's F-measure must be lower than the by 91.64% the combined strategies. Therefore, the results of the current argument classifier are comparable to both state-of-the-art systems.

6.6 Identification and classification with gold parses

In this set of experiments, the baseline system was first run to identify the arguments from the gold parses using the logistic regression identifier. Then the classified the detected arguments using the logistic regression baseline argument classifier shown in table 6.4.

With the current system, an argument identifier was first applied on the Penn Treebank parse trees, the gold parses, to identify the core and non-core arguments. The identifier itself is built upon a logistic regression classifier. Then, the best argument classifier, which is the nearest-neighbor classifier, with the BAC, Level-I, and Level-II features, was used to classify the core and non-core arguments into one of the semantic role labels.

Table 6.7 lists the results from this set of experiments.

Experiment	Precision	Recall	F-measure
`Baseline system`			
Arg. Identification arg or non-argument	92.16	89.47	90.80
Arg. Id. & Classification	73.22	71.90	72.55
`Current system`			
Arg. Identification core, non-core, non args.	90.31	86.42	88.32
Arg. Id. & Classification	81.16	78.23	79.67
% of classification correctness from identification	89.87	90.52	90.20

Table 6.7: Experimental results with the gold parses

Table 6.7 shows that the F-measure of the argument identifier for the current system is 88.32%, while that of the argument identifier for the baseline is 90.70%. Since the current system is the only system that identifies core and non-core semantic arguments, there is no direct comparison with any other system in terms of the performance of argument identification.

Table 6.7 also shows that KNN/IBK classifier with the design of the current project outperforms the baseline system by 7% in F-measure.

In addition to the above results, one can make two additional observations.

First, the argument classifier performs consistently as it does in the oracle systems. This is so because the percentage of correctness in precision, recall, and F-measure that the argument classifier obtains on the identified arguments match with those of oracle systems. Table 6.7 shows that argument classifier gains the precision, recall, and F-measure of 81.16%, 78.23%, and 79.67% out of the corresponding precision, recall, and F-measure of 90.31%, 86.42%, and 88.32% by the argument identifier. This suggests that the argument classifier maintains the percentage correctness of 89.87%, 90.52% and 90.20%, which match with the precision, recall, and F-measure of 89.24%, 89.57%, and 89.41% by the KNN oracle argument classifier shown in table 6.6. Thus, the argument classifier centered on the BAC features and augmented with the two level of back-off features performs consistently. As a result, this further confirms the verified Hypotheses I, II, and III.

Second, the final system performance is dependent on the performance of the argument identifier. Although the argument classifier maintains its performance on getting about 90% of the correctly identified arguments right, the final system performance was affected by the result of the argument identifier.

6.7 Identification and classification with automatic parses

In this last set of experiments, the settings were essentially identical to the previous set of experiments, except that the input to the argument identifier were the automatically parsed data using the Charnaik parser (McClosky et al., 2006). Table 6.8 lists the results from this set of experiments.

One can make the following observations from the results.

The overall F-measure from this set of experiments is about 5% above the baseline F-measure, which is consistent with the results from the experiments on the gold parses.

The argument identification module on the automatically parsed data yields an F-measure of 79.08%, which is 9.24% lower than the F-measure of 88.32% from the ar-

Experiment	Precision	Recall	F-measure
`Baseline system`			
Arg. Identification arg or non-argument	82.51	80.25	81.36
Arg. Id. & Classification	66.13	65.04	65.58
`Current system`			
Arg. Identification core, non-core, non args.	80.88	77.36	79.08
Arg. Id. & Classification	72.20	69.57	70.86
% of classification correctness from identification	89.26	89.93	89.60

Table 6.8: Experimental results with the automatically parsed input

gument identification task on the gold parses. As a result, the argument classifier gives only an F-measure of 70.86%, which is about an 8% drop from the 79.67% from the classification results on the gold parses. Both results show that the system is affected by the parsing results.

However, the performance of the argument classifier still remains consistent in this set of experiments. The reasoning here is identical to that in the preceding section. That is, the percentage of correctness in precision, recall, and F-measure that the argument classifier obtains on the identified arguments match with those of oracle systems, as well as with those on the gold parses. Table 6.7 shows that argument classifier gains the precision, recall, and F-measure of 72.20%, 69.57%, and 70.86% out of the corresponding precision, recall, and F-measure of 80.88%, 77.36%, and 79.08% by the argument identifier. This suggests that the argument classifier maintains the percentage correctness of 89.26%, 89.93%, and 89.60%, which match with the precision, recall, and F-measure of 89.24%, 89.57%, and 89.41% by the KNN oracle argument classifier shown in table 6.6. Thus, the argument classifier centered on the BAC features and augmented with the two level of back-off features performs consistently.

6.7.1 Discussion

The consistency of the argument classifier throughout the three sets of experiments shows that the design behind the argument classifier is effective. However, at the same time, the overall system performance is affected by the that of the argument identification module. As a matter of fact, the overall F-measure of 70.86% on the automatically parsed data is below state-of-the-art performance, which is around 79%. This is an important motivation for improving the argument identification module.

Chapter 7
Conclusion

While the present dissertation tackles the semantic role labeling task in general, it focuses on discovering the right syntactic/semantic knowledge to create features that generalize across the syntactic variations that a verb appears in and involve argument movement or displacement.

Incorporating the knowledge of context dependence among the semantic roles of the core arguments, that of the syntactic structures involving argument movement or displacement, that of the relations between constituents defined in dependency grammar, and that of verb argument structure, the current dissertation designs the base argument configuration features that effectively generalize across different syntactic structures that a verbs appears in. Working together with the two levels of features that handle the syntactic structures involving unrealized core arguments and unknown verbs respectively, the BAC features effectively deal with the argument classification task. Due to this feature design, the argument classification module performs consistently regardless of the performance of the argument identification module.

While the current dissertation discovers the right knowledge and feature design to tackle the argument classification task, the overall system performance is affected by the argument identification task. The argument identification module identifies the semantic argument(s) of a given verb based on a single parse tree of the Charniak parser. The module then classifies the semantic arguments of the verb into core, non-core, and non arguments based on a set of generic features. The experiments show that this approach does not yield the optimal identification results and hence affects the overall system performance.

The immediate future task would be to improve argument identification by incorporating multiple parse trees of the same sentence from the Charniak parser or by combining multiple parses from different automatic parsers. Integrating multiple parses will allow for global optimization and alleviates the limitation of single parses.

Bibliography

Aha, D. and D. Kibler. 1991. Instance-based learning algorithms, *Machine Learning*, 6, 37–66.

Carnie, Andrew. 2002. *Syntax, A Generative Approach*, Blackwell Publishing.

Carreras, Xavier and Lluis Marquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling, in *Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL)*, Ann Arbor, Michigan, 152164.

Carter, R. J. 1988. *Some Linking Regularities*.

Chen, John and Owen Rambow. 2003. Use of deep linguistic features for the recognition and labeling of semantic arguments, in *Proceedings of EMNLP-2003*.

Chomsky, Noam. 1986. *Knowledge of Language: Its Nature, Origin and Use*, Praeger.

Collins, Michael and Terry Koo. 2005. Discriminative reranking for natural language parsing, *Computational Linguistics*, 31(1), 25–69.

Culicover, Peter W. and Ray Jackendoff. 2005. *Simpler syntax*, Oxford University Press.

Dowty, David. 1987. Thematic proto-roles, subject selection, and lexical semantic defaults, Linguistic Society of America Colloquium.

Fellbaum, Christiane. 1998. *WordNet, An Electronic Lexical Database*, MIT Press.

Fillmore, Charles J. 1968. *Universals in Linguistic Theory*, Holt, Rinehart and Winston, chap. The Case of Case, 1–88.

Fillmore, Charles J. 1970. *Readings in English Transformational Grammar*, Ginn and Company, chap. The Grammar of Hitting and Breaking, 120–133.

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*.

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)*, Hong Kong, 512520.

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 (ACL-02)*, Philadelphia, PA, 239246.

Gordon, Andrew and Reid Swanson. June 23-30, 2007. Generalizing semantic role annotations across syntactically similar verbs, in *Proceedings of the 2007 meeting of the Association for Computational Linguistic*, Prague, Czech Republic.

Haghighi, Aria, Kristina Toutanova, and Christopher Manning. 2005. A joint model for semantic role labeling, in *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, Ann Arbor, Michigan, 173–176.

Hajič, Jan. 2009. Conll-2009 shared task description, URL `http://ufal.mff.cuni.cz/conll2009-st/task-description.html`.

Jackendoff, Ray. 1990. *Semantic Structures*, MIT Press.

Johansson, Richard and Pierre Nugues. 2007a. Extended constituent-to-dependency conversion for english, in *Proceedings of NODALIDA 2007*.

Johansson, Richard and Pierre Nugues. 2007b. Lth: Semantic structure extraction using nonprojective dependency trees, in *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, Prague, 227230.

Johansson, Richard and Pierre Nugues. 2008. Dependency-based semantic role labeling of propbank and nombank, in *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*.

Levin, Beth and Malka Rappaport Hovav. 1996. From lexical semantics to argument realization, *Manuscript*, 1, 1–8.

Levin, Beth and Malka Rappaport Hovav. 2005. *Argument Realization*, Cambridge, URL `http://www.mitpressjournals.org/doi/pdfplus/10.1162/coli.2006.32.3.447?cookieSet=1`.

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

M'arquez, Llus, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. 2008. Semantic role labeling: An introduction to the special issue, *Computational Linguistics*, 34(2), 146–159.

McClosky, David, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing, in *Proceedings of the North American Conference on Computational Linguistics*.

Nivre, Joakim. 2007. *Inductive Dependency Parsing*, Springer.

Palmer, Martha, Dan Gildea, and Paul Kingsbury. 2003. The proposition bank: An annotated corpus of semantic roles, *Computational Linguistics*, 1, 10–12.

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

Perlmutter, David M. and Paul M. Postal. 1984. *Studies in relational grammar*, Chicago, IL: University of Chicago Press.

Pesetsky, David Michael. 1982. *Paths and Categories*, Ph.D. thesis, MIT.

Pinker, Steven. 1989. *Learnability and Cognition: The Acqusition of Argument STructure*, MIT Press.

Pradhan, Sameer, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005. Support vector learning for semantic argument classification, *Machine Learning*, 60(1), 11–39.

Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James H. Martin, and Daniel Jurafsky. 2004. Shallow semantic parsing using support vector machines, in *Association for Computational Linguistics annual meeting (HLT/NAACL-2004)*.

Punyakanok, Vasin, Dan Roth, and Wen tau Yih. 2005. The necessity of syntactic parsing for semantic role labeling, in *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Pustejovsky, James. 1995. *The Generative Lexicon*, MIT Press.

Quirk, Randolph, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1985. *A Comprehensive Grammar of the English Language*, Longman.

Radford, Andrew. 1997. *Syntactic theory and the structure of English*, Cambridge.

Ruppenhofer, Josef, Michael Ellsworth, Miriam Petruck, Christopher Johnson, and Jan Scheffzyk. 2006. *FrameNet II: Extended Theory and Practice*, FrameNet Project. Available with the FrameNet database data set.

Saeed, John. 2003. *Semantics*, Blackwell Publishing.

Stowell, Timothy. 1981. *Origins of Phrase Structure*, Ph.D. thesis, MIT.

Surdeanu, Mihai, Richard Johansson, Lluis Marquez, Adam Meyers, and Joakim Nivre. 2008a. Conll-2008 shared task description, online.

Surdeanu, Mihai, Luis Marquez, 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 Lluis Marquez. 2008b. Analysis of joint inference strategies for the semantic role labeling of spanish and catalan, in *Proceedings of the 9th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*.

Surdeanu, Mihai and Jordi Turmo. 2005. Semantic role labeling using complete syntactic analysis, in *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*.

Thompson, Cynthia A., Roger Levy, and Christopher D. Manning. 2003. A generative model for semantic role labeling, in *ECML 2003*, pp. 397–408.

Toutanova, Kristina, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning imrpoves semantic role labeling, in *Proceedings of ACL 2005*, Ann Arbor, MI.

Vickrey, David and Daphne Koller. 2008. Sentence simplification for semantic role labeling, in *Proceedings of ACL-08: HLT*.

Witten, Ian H. and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, 2 edn.

Xue, Nianwen and Martha Palmer. 2004. Calibrating features for semantic role labeling, in *Proceedings of EMNLP*, Barcelona, Spain.

Yamada, Hiroyasu. 2008. Converter for penn treebank corpus, URL `http://www.jaist.ac.jp/~h-yamada/`.